99
2EJ

UNIVERSIDAD NACIONAL AUTONÓMA DE MÉXICO

FACULTAD DE INGENIERÍA

KIOSCOS DE INFORMACIÓN

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

P R E S E N T A N:

SÁNCHEZ LARIOS CARLOS

TEAPILA CUATECO RICARDO

VALERIANO DE JESUS MARÍA DEL PILAR

Director de Tesis: Ing. Domingo Palao Muñoz

México, D.F. Agosto, 1995

Con eterno agradecimiento ...

A nuestra máxima casa de estudios, la Universidad Nacional Autónoma de México, por habernos albergado, por dejarnos ser parte de ella y por formarnos íntegramente como profesionales y personas.

A la Facultad de Ingeniería y todos sus profesores que nos dieron las bases y supieron motivarnos a seguir adelante.

A la Dirección de Cómputo para la Administración Académica por darnos la oportunidad de dejarnos plasmarle algo de nosotros, por todas las facilidades, por dejarnos tener acceso a los recursos de nuestra universidad, por confiar en nosotros y por otorgarnos la experiencia que es tan valiosa para nuestro futuro profesional.

Agradecimientos especiales a nuestro director de tesis :

Ing. Domigo Palao Muñoz

por su ayuda y orientaciones

así como a :

Act. Ma. Teresa Molina Charpenel
Ing. Fernando Baz García
Sr. Octavio López Chávez
Lic. Lorea San Martín Tejedo
Ing. Ana Cecilia Padilla Chávez
Ing. Edgar Valencia Figueroa
Sr. Víctor Javier Arrieta Rosas
Ing. Gustavo de la Cruz Tovar
Al equipo de Multimedia de la Coordinación Técnica

y en general a todas aquellas personas que no hemos listado pero que han influido para culminar esta etapa que es tan importante para nosotros.

Atte.

Pilar, Ricardo y Carlos

A mis padres, mi hermana y toda mi familia...

Por ser el apoyo, el estímulo, la confianza y por prepararme en la vida. Con gratitud permanente, cariño, emoción y respeto les dedico este gran esfuerzo.

A mis Amigos...

Que me brindan su apoyo y consejos en los momentos difíciles, por alentarme a seguir adelante, gracias por compartir conmigo sus experiencias, por depositar en mí su confianza y por ayudarme a encontrar este valioso sentimiento llamado amistad.

A todos mis compañeros de la DCAA...

Gracias por tantas cosas...

A ustedes, esas personas que siempre me acompañan y caminan junto a mí, porque a pesar de no estar físicamente se encuentran tan presentes.

A mis compañeros Ricardo y Carlos...

Gracias por haberme brindado la oportunidad de trabajar con ustedes, por dejarme compartir algo tan importante, por la enorme paciencia, dedicación y confianza que han tenido en mí en todo este tiempo.

Pilar

A mis padres:

Como una muestra de mi cariño y agradecimiento por tantos años de apoyo recibido, lo cual constituye la herencia más valiosa que pudiera recibir. Gracias por su confianza y comprensión, eso me ayudó a iniciar un largo camino de superación que es lo que ustedes esperaban. Los llevaré siempre en mi corazón.

A mis hermanos:

Yola, Güicho, Lolita y mi sobrinito Adrián. Que este primer esfuerzo de terminar una carrera profesional sirva de ejemplo y estímulo para ustedes.

A mis compañeros de la DCAA:

A todos los becarios con quienes he convivido momentos tan agradables, tanto en el trabajo como fuera de él.

A mis grandes amigos:

Edgar, Ceci, Iván y Fernando. Gracias por todo el apoyo recibido y dejarme ser parte de ustedes.

Por último, a mis compañeros de tesis:

Pilar y Carlos. Gracias por confiar en mí y haberme soportado todo este tiempo.

Ricardo

Finalmente, ha llegado uno de los momentos más esperados de mi vida... el de haber podido culminar satisfactoriamente mis estudios profesionales. Y es justamente con el trabajo más difícil que al final representó para mí hablando en el ámbito escolar como es mi tesis profesional, en donde quiero manifestar y dedicar con todo cariño este trabajo a todos mis seres queridos. Empezando por mis padres, que me concedieron la oportunidad de abrirme las puertas al Universo del conocimiento, siempre con todo el amor y la mejor de las voluntades, día con día y siempre, empezando porque también me dieron la oportunidad de estudiar inglés, sin el cual no hubiera podido traducir y comprender muchos aspectos de esta tesis profesional. Gracias miles, aunque estoy seguro que la mejor manera de dárselas será a lo largo de la vida.

De la misma manera quiero agradecerle a mis hermanos Martha y Jose Luis, a los cuales, cada día que pasa los quiero más. Dedico también este trabajo a mis cuñados: Lucy y Angel a los cuales quiero y respeto mucho. Igualmente a todos mis sobrinos habidos y por haber, especialmente a Mariana, Nayeli y mi sobrino favorito Eric.

A todos mis tíos y tías que son definitivamente como unos segundos padres y madres respectivamente. Ya que ahora tienen un sobrino Ingeniero, dispuesto siempre a ayudarlos. En especial a mis tías Esther, Lourdes, Lupita, Herminda, Clara, Ceci, Amparo, Rosa, Bertha y Estela. Y mis tíos Gonzalo, Ramón, Arturo, Rogelio, Guillermo, Jose Luis, Felipe, Octavio y Roberto.

A todos mis primos que sinceramente de querer nombrarlos tendría que utilizar muchas cuartillas, pero sólo por mencionar a algunos: Esther, Gabriela, Mauricio, Ivonne, Nicolás, Ricardo (ojalá y termine su escuela) y muchos más, aunque no lo crean siempre me estoy acordando de ustedes.

A mis padrinos por aceptarme incondicionalmente y tener fé en mí.

A todos mis amigos y personas que directa o indirectamente fueron un aliciente para que pudiera llegar hasta este momento tan importante para mí, en especial a: Alfredo (el amigo más antiguo que tengo), Iván, Cesar, Victor, Lorena, Verónica y Xathya. Y a todas las amigas de mi mamá y amigos de mi papá, como lo son: Lidia, Ceci, Celia, Concepción, Isabel, Ing. Licea e Ing. Sergio. También a la mamá de Xathya. A mis maestros y director de Tesis Ing. Domingo Palao M. A todos ellos gracias por su comprensión, apoyo y confianza.

A mi esposa, que aunque ahorita no tengo, anticipadamente se la quiero dedicar.

Un apartado especial para todos mis compañeros de la Universidad porque ellos comprenderán mejor que nadie cuanto trabajo, tiempo, dedicación y entrega se necesitan para poder llegar a ser Ingeniero en la máxima casa de estudios.

Y por último a mis compañeros de tesis, los cuales tuvieron que vivir también todas aquellas horas, días festivos y fines de semana ininterrumpidos para poder finalizar satisfactoriamente este trabajo. Empezando por Ricardo, el cual me dió una muestra de profesionalismo, entrega, Calidad y amistad. Gracias por mostrarme el camino de la excelencia y por corregirme esos errores que cometía en el desarrollo del trabajo, y por enseñarme tantas cosas relativas a la computación. Y a Pilar, porque siempre que le preguntaba algo me lo respondía y me ayudaba. Siempre tendrás mi amistad.

Muchas gracias a todos, aunque esto es sólo el principio.

Atentamente: Carlos Sánchez Larios.

Agosto de 1995

ÍNDICE

Capítulo 1. Introducción 1

Capítulo 2. Antecedentes 7

2.1 Las GUI's (Graphical User Interface) 8
2.1.1 El sistema de ventanas X-Window

2.2 Arquitectura cliente/servidor 18
2.2.1 Características del cliente
2.2.2 Características del servidor
2.2.3 Ventajas y desventajas sobre otros modelos
2.2.4 Ejemplos

2.3 Multimedia 29
2.3.1 Definición de Multimedia
2.3.2 Anatomía
2.3.3 Aplicaciones en Kioscos de Información

2.4 Sistemas Distribuidos 44
2.4.1 Sistemas Abiertos
2.4.2 Arquitectura de los Sistemas Distribuidos

Capítulo 3. Recursos Disponibles 53

3.1 La Dirección de Cómputo para la Administración Académica 54
3.1.1 Proyectos desarrollados
3.1.2 Equipo disponible

3.2 Tecnologías de Redes locales 64
3.2.1 Redes tipo Ethernet
3.2.2 Token Ring
3.2.3 FDDI

3.3 RedUnam 75

3.4 La Red Internet 79
3.4.1¿Cómo trabaja Internet?
3.4.2 Servicios disponibles

Capítulo 4. Mosaic para sistema X-Window 105

4.1 Descripción 106
4.2 Requerimientos 110
4.3 Cliente 112
4.4 Servidor 113
4.4.1 El protocolo HTTP (HyperText Transfer Protocol)

4.5 Los URL's (Uniform Resource Locator) 115
4.5.1 Tipos de URL's

4.6 El Lenguaje HTML (HyperText Markup Language) 121
4.6.1 Formas
4.6.2 Los CGI's (Common Gateway Interface)

4.7 Especificación de Recursos 140

Capítulo 5. Propuesta del Proyecto 147

5.1 Sistema de Información integrado en un Kiosco 147
5.2 Características del equipo 149
5.3 Desarrollo 179
5.3.1 Instalación y Configuración del cliente Mosaic
5.3.2 Instalación y configuración del servidor http
5.3.3 Generación de documentos Hipertexto

5.4 Integración del Kiosco 218
5.5 Pruebas realizadas 224

Capítulo 6. Caso de Aplicación: Sistema de Inscripciones a cursos de la DCAA 231

6.1 Introducción 232
6.1.1 Importancia de las Bases de Datos
6.1.2 Planteamiento del problema

6.2 Conceptos Generales de Bases de Datos 237
6.2.1 SYBASE como manejador de bases de datos
6.2.2 Clientes en SYBASE

6.3 Sistema de inscripciones a cursos implantado en Kioscos de Información 276
6.3.1 Descripción del proceso
6.3.2 Diseño del sistema

6.4 Desarrollo 286
6.4.1 Creación de las tablas en SYBASE
6.4.2 Generación de formas en HTML
6.4.3 Programación de las interfaces entre Mosaic y clientes en SYBASE a través de CGI's
6.4.4 Implantación del sistema de inscripciones
6.4.5 Pruebas finales

Capítulo 7. Perspectivas de desarrollo 347

Capítulo 8. Conclusiones 353

Apéndice A 357

Apéndice B 361

Apéndice C 365

Apéndice D 383

Apéndice E 387

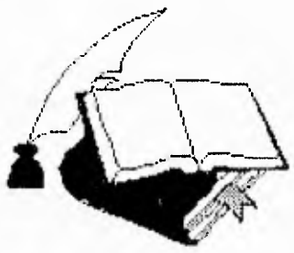
Apéndice F. Manual de Usuario 389

Glosario 403

Bibliografía 415

Capítulo 1

Introducción



La Universidad Nacional Autónoma de México es la institución educativa más importante del país, se ha caracterizado por estar a la vanguardia en equipo y tecnología, además de ser la portadora más importante en la difusión de la cultura informática en todas las áreas del conocimiento. Para lograr este objetivo, cuenta con dependencias como la Dirección General de Servicios de Cómputo Académico (DGSCA) y la Dirección de Cómputo para la Administración Académica (DCAA) por mencionar algunas, donde se han desarrollado proyectos basados en modernas tecnologías para el manejo y distribución de la información.

Kioscos de Información es un nuevo concepto que involucra aplicaciones desarrolladas en multimedia, redes de computadoras, arquitectura cliente/servidor, bases de datos, GUI's (Graphical User Interface) y otros elementos importantes en materia de cómputo que hacen posible la integración de los conceptos antes mencionados en una aplicación como la que desarrollamos en el presente trabajo.

Siempre hemos tenido la necesidad de buscar nuevos medios de comunicación para difundir información, entre los cuales podemos destacar: la radio, la televisión, etc. El avance vertiginoso de la tecnología digital ha ayudado a crear novedosas aplicaciones en todos los campos de la informática y son reflejados actualmente, por ejemplo; en los modernos equipos de digitalización de imágenes y sonido, ambientes gráficos, etc., donde el ser humano enfrenta y conoce nuevas técnicas para el manejo de información.

En los últimos años ha crecido la importancia de presentar aplicaciones de cómputo, en un ambiente amigable y agradable para el usuario, tal es el caso del ambiente gráfico, mediante el cual con ayuda del *ratón* e íconos, el usuario especifica una orden y la máquina genera un resultado. Desde sus inicios, este tipo de aplicaciones han requerido de una gran cantidad de recursos de computadora tanto de hardware como de software (memoria, almacenamiento masivo, dispositivos periféricos e inclusive programación robusta), lo cual significa una limitante para tales aplicaciones en cuanto a su implantación.

Por otra parte, la evolución que han tenido los procesadores en los últimos años, hacen posible tener aplicaciones más rápidas y por lo tanto, podemos ahora incluir elementos de multimedia para mejorar presentaciones hacia el usuario. Tenemos el caso de las Estaciones de Trabajo, las cuales son veloces y tienen la facilidad de procesar grandes volúmenes de información en forma rápida, por tal motivo algunas de las aplicaciones actuales desarrolladas para este tipo de arquitecturas incluyen ya un ambiente gráfico.

Desde que las computadoras de la UNAM están conectadas a la Red Internet (la red de computadoras más grande del mundo), es posible utilizar de manera efectiva uno de los instrumentos más poderosos que la tecnología de comunicaciones ha puesto al servicio de la comunidad científica en las universidades y centros de investigación del mundo entero. Al emplear las redes de transferencia de datos, es posible intercambiar no únicamente mensajes, notas o cartas (lo que se conoce como *correo electrónico*) sino grandes cantidades de información en diferentes formatos a una velocidad nunca antes vista. Como un ejemplo tenemos a la RedUnam, donde se tiene una infraestructura de comunicación que involucra desde computadoras personales hasta la supercomputadora Cray conectadas a través de un anillo de fibra óptica del cual hablaremos más adelante.

Otro aspecto también importante digno de mencionar, es la importancia que tiene la Red Internet en todas las áreas del conocimiento, debido a que en esta red se puede encontrar software e información sin costo alguno, así como contactar con los autores de los mismos para intercambiar ideas, fomentando una mayor difusión de la cultura informática entre todas las entidades gubernamentales, educativas y comerciales que se encuentran en diferentes partes del planeta.

Lo anterior nos lleva a pensar en el desarrollo de una aplicación que incluya los conceptos mencionados anteriormente aprovechando que cada una de las partes que la integrarán ya existen individualmente.

La inquietud surge a raíz de actividades desarrolladas dentro de la Coordinación Técnica de Redes e Interoperabilidad de la Dirección de Cómputo para la Administración Académica (DCAA), las cuales en los últimos meses han involucrado el uso de nuevas tecnologías tanto en hardware como en software, así que podemos aprovechar los recursos con los que actualmente cuenta la Universidad en materia de cómputo y sistemas desarrollados por algunas Universidades de otros países. En su conjunto tenemos disponibles herramientas que podemos aplicar a un sistema específico como es el caso del servicio de información que estará a disposición de la comunidad universitaria.

El proyecto que presentamos como tema de tesis, es el resultado de largos meses de investigación, durante los cuales tuvimos la oportunidad de conocer a través de novedosas fuentes de información (correo electrónico, archies, gophers, servidores de noticias y otros), la existencia de herramientas de software que forman parte del *dominio público*[1] en la Red Internet, tal es el caso de **Mosaic** que por clasificarse dentro de este tipo de software, no tiene costo alguno.

El sistema que proponemos tiene como objetivo fundamental, proporcionar información a la comunidad universitaria y público en general. Consiste en la

[1] Software de dominio publico, es aquel que se encuentra disponible en la Red Internet y que se puede usar, pero no se puede vender. Regularmente se encuentra en servidores FTP anónimos.

instalación de estaciones de trabajo (Workstations) en lugares estratégicos dentro del campus universitario, de tal forma que utilizando un ambiente gráfico en una pantalla *touchscreen*, el usuario pueda interactuar con el mismo, observando imágenes, videos e incluso escuchar sonidos.

El proyecto se divide en dos fases, la primera es la realización del Kiosco en sí y en la segunda presentamos un caso en el cual se puede aplicar el proyecto que proponemos. No nos limitamos únicamente a *presentar información*, sino que resaltamos también la forma en como podemos aprovechar otras herramientas e integrarlas en nuestro sistema, por ejemplo, los manejadores de Bases de Datos. Veremos como es posible aprovechar toda una infraestructura de red (por ejemplo RedUnam) para compartir diferentes recursos de cómputo y ofrecer servicios a la comunidad, los cuales podrían solucionar diversos problemas no sólo dentro de la Universidad, sino que podemos considerar a la iniciativa privada y entidades gubernamentales.

La presente tesis está organizada en 8 capítulos los cuales describiremos brevemente a continuación.

Capítulo 1 Introducción. Está dedicado a definir el objetivo de la realización del proyecto.

Capítulo 2 Antecedentes. Incluye conceptos teóricos básicos para el desarrollo de esta tesis, conceptos simples pero que tienen un gran significado. Para algunos son, palabras nuevas pero que incluyen años de dedicación y desarrollo por parte de sus creadores. Incluimos los conceptos más importantes del proyecto tales

como: Interfaz gráfica del usuario (GUI), arquitectura cliente/servidor, Multimedia y Cómputo Distribuido.

Capítulo 3 Recursos Disponibles. En esta parte, destacamos algunas de las tecnologías de cómputo disponibles en la Dirección de Cómputo para la Administración Académica las cuales utilizamos en la realización del proyecto. Nos referimos a los aspectos más importantes tales como la infraestructura de red con la que cuenta la Universidad Nacional Autónoma de México (RedUnam), así como a la red más grande del mundo (Red Internet), detallando de esta última, los servicios que proporciona, la información que podemos encontrar, su importancia, etc.

Capítulo 4 Mosaic para sistema X-Window. Este capítulo lo dedicamos a presentar detalles de Mosaic, el software que utilizamos para desarrollar la aplicación y cómo es posible interactuar con otras herramientas que forman parte del dominio público, así como la interacción con otras máquinas. También consideramos el diseño de una interfaz entre Mosaic y Bases de Datos bajo el concepto cliente/servidor.

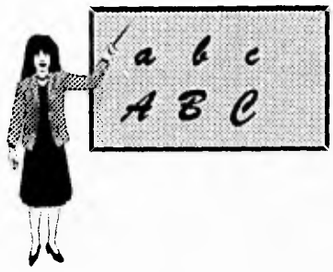
Capítulo 5 Propuesta del Proyecto. Este apartado incluye la propuesta del proyecto como tal, detallamos en que consiste, que se necesita para ponerlo en marcha y como se desarrollaron las pruebas que aplicamos, etc. A partir de aquí, analizamos la propuesta de un caso de aplicación del proyecto que presentamos, la cual será descrita en el capítulo 6.

Capítulo 6 Caso de Aplicación: Sistema de Inscripciones a cursos de la DCAA (Dirección de Cómputo para la Administración Académica). Lo dedicamos a analizar un caso de aplicación de un *Kiosco de Información*. Este

capítulo marca la segunda fase del trabajo, es importante mencionar que nuestro proyecto es aplicable a resolver un gran número de problemas, y eso es precisamente lo que hacemos; resolver uno de ellos. Tomamos como base un sistema de inscripciones para los cursos de la Dirección de Cómputo para la Administración Académica.

Capítulo 7 Perspectivas de Desarrollo. Este capítulo lo dedicamos exclusivamente para las perspectivas de desarrollo que tiene este proyecto, destacamos la importancia y utilidad no sólo dentro de la Universidad como un medio informativo, sino como un producto comercial para la iniciativa privada.

Capítulo 8 Conclusiones. Por último, tenemos las conclusiones al trabajo y proyecto desarrollado como tema de tesis.



Capítulo 2

Antecedentes

Antes de comenzar con el desarrollo de nuestro proyecto, vamos a explicar algunos conceptos importantes relacionados con el mismo. Es necesario tenerlos en cuenta debido a que los usamos en capítulos posteriores.

La historia de la evolución de las computadoras es realmente interesante, ha abarcado todas las áreas del conocimiento. Actualmente, tenemos una mayor cantidad de usuarios que cada día crece con el uso de las nuevas tecnologías, y a su vez demandan nuevos servicios y aplicaciones. Con el surgimiento de los ambientes gráficos, los sistemas de cómputo se hacen más fáciles de manejar; es sorprendente saber que con un simple click del *ratón*, se ejecutan una gran cantidad de instrucciones que el usuario ya no necesita teclear.

Otro concepto interesante que tomamos en cuenta en este capítulo es la arquitectura cliente/servidor; mencionamos sus características principales e incluimos algunos ejemplos. El lector se dará cuenta que el concepto es muy sencillo pero, hay que comprender como funciona en sistemas prácticos basados en él.

Multimedia es otro de los elementos que se utilizará en el proyecto pero en este caso en un ambiente de red.

Por último, las redes de computadoras nos permiten actualmente compartir recursos entre máquinas. Esto nos lleva a definir términos que nos permitan explicar como se realizan este tipo de operaciones. Particularmente, hablamos de los sistemas distribuidos, la tendencia que tienen y sus características. En nuestro caso, el proyecto se basa precisamente en este concepto, más adelante veremos como es

posible acceder a recursos de otras máquinas utilizando Mosaic y aprovecharlos en el desarrollo de aplicaciones que cada día se vuelven más importantes.

2.1 Las GUI's (Graphical User Interface)

El hombre siempre ha buscado la forma de comunicarse con el mundo que lo rodea. Es para todos importante que los demás nos entiendan, por ello no es necesario usar palabras tan rebuscadas o medios muy complicados para lograrlo. Esto se refleja mucho en el mundo de la computación, a medida que evoluciona, los programadores y creadores de sistemas buscan que sus trabajos sean comprendidos por los demás. Ya no es tan funcional que se logren crear funciones poderosas para ejecutar algo, sino que ahora es necesario que éstas estén disponibles para cualquier tipo de usuario, y no sólo para los que lograban ser expertos. No se quiere decir con esto que todo puede estar visible para nosotros y que ahora todos seamos expertos, ya que la experiencia y dedicación nos conducirán a lograrlo. Pero al punto donde se quiere llegar es: si logramos que este camino sea más fácil, todos estaremos aptos para lograr cualquier cosa.

La computación puede resultar un poco complicada para algunas personas, pero tal vez el problema es que no han interactuado con un sistema amigable que les permita realizar sus funciones.

Actualmente los nuevos sistemas y la mayoría de ellos incluyen interfaces gráficas donde se pretende que sea más fácil para el usuario captar ese mundo que en ocasiones le puede resultar complicado. Ya no es necesario que tecleé toda una línea para ejecutar un comando, sino que ahora se le presenta un dibujo que representa la ejecución de una tarea que él quiere se realice. Es por eso que surgen conceptos y herramientas que ayuden al usuario, tales como las GUI's. Una GUI

(Graphical User Interface por sus siglas en inglés) es en forma básica un menú, como su nombre lo indica es una *Interfaz de Usuario Gráfica*, ya se había comentado la importancia de los gráficos en los sistemas. Pero, ¿qué es una interfaz de usuario? Es la parte del sistema informático que define la forma como el usuario interactúa con el sistema, de cómo se introducen los mandatos y de cómo se despliega la información.

La GUI permite al usuario seleccionar un programa usando el apuntador del *ratón*. Estos programas son desplegados generalmente en forma de íconos. Un ícono es un pequeño símbolo o dibujo sobre la pantalla que representa un programa específico. Por ejemplo, un ícono con un dibujo de un pincel puede representar un paquete para dibujar. Para cargar el programa, el usuario apunta sobre el pincel usando el *ratón* y hace un click (o doble click) sobre el botón del *ratón*. De este modo el programa podrá ser ejecutado.

El mundo de las GUI's apareció formalmente y de manera atractiva cuando Apple introdujo al mercado la Macintosh, pero en realidad sus comienzos se encuentran en Xerox, cuando el centro de investigaciones de Palo Alto de la misma compañía comenzó con un proyecto llamado Xerox Star.

El ambiente de Windows vendría a ser precisamente el primo de la Mac y así como estos dos ambientes, existen muchos otros como X Window, NextStep, etc. Con Macintosh nos enfrentamos a la definición más clara de una GUI, la Mac define las partes que están relacionadas con una GUI como las siguientes:

- un dispositivo apuntador, generalmente un *ratón*.
- menús sobre pantalla que pueden aparecer o desaparecer bajo el control de algún dispositivo apuntador.
- ventanas que gráficamente despliegan los procesos que realiza la computadora para ejecutar alguna tarea.

- íconos que representan archivos, directorios y cajas de diálogo, botones, bordes, cajas de verificación, etc.

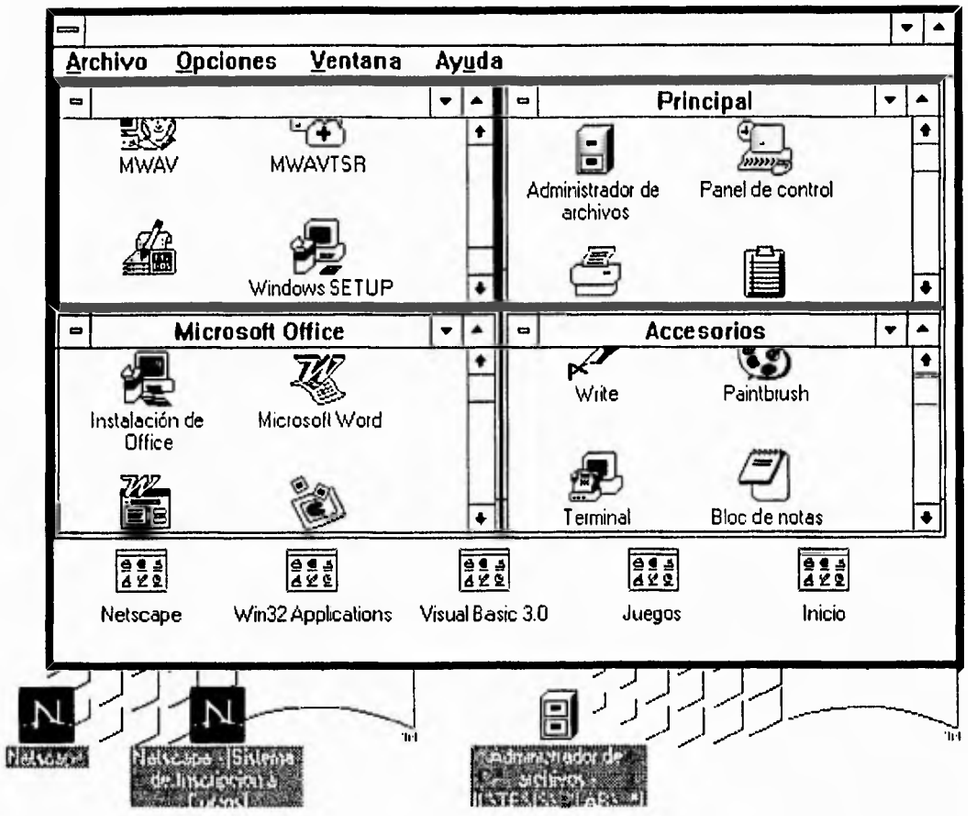


Figura 2.1 Ejemplo del ambiente gráfico Windows como una GUI

Es seguro que las GUI's de hoy tienen muchas variantes, algunas por ejemplo, no usan íconos, en otras son opcionales o pueden aparecer solo algunas veces. Otras GUI's requieren un *ratón* mientras otras permiten trabajar y manipular todo desde el teclado. Pero en general, todas son iguales superficialmente, aunque existen algunas "híbridas"; la mayoría consisten de tres componentes principales: un sistema de ventanas, un modelo de imagen y una *interfaz de programas de*

aplicación (API). Un sistema de ventanas es un grupo de herramientas de programación y comandos que construyen las ventanas, menús y cajas de diálogo que aparecen sobre la pantalla. Con todo esto se controla la creación de ventanas, su tamaño y el movimiento de una a otra donde el usuario realiza otras funciones. Un ejemplo de un sistema de ventanas es X-Window.

El modelo de imagen define cómo las letras (fonts) y gráficos son creados sobre la pantalla. Por ejemplo, la presentación y tamaño de un texto en un procesador de palabra o un programa de publicidad especificado a través de un modelo de imagen, así como las líneas o curvas del programa de CAD. El PostScript es el modelo de imagen mejor conocido, familiar de las impresoras láser, el Display PostScript es una versión de pantalla del modelo de imagen del PostScript. El modelo de imagen de la Macintosh es el QuickDraw, y el PM de Microsoft para OS/2 usa un modelo de imagen llamado GPI (Interfaz de Programación Gráfica).

Una API es un conjunto de llamadas a funciones de un lenguaje de programación, son programas específicos que se encargan de las ventanas, menús, barras de desplazamiento e íconos que aparecen sobre la pantalla. Windows tiene su propia API. Las ventanas de DEC usan una API llamada XUI (X User Interface), la cual incluye llamadas a funciones para el sistema X Window. Open Look es una nueva API para sistemas operativos de Sun. NextStep usa su propia API (definido por bibliotecas y objetos llamados *kits*) y su propio sistema de ventanas llamado *windows server*.

Algunas GUI's proveen el acceso a interfaces de comandos en línea convencionales para el usuario. Por ejemplo, el paso de argumentos para aplicaciones o ver el texto de un archivo sin usar el *ratón*, menús e íconos. Con todas estas variantes es fácil desglosar las diferentes familias de GUI's.

Cerca de los años 80's fue muy común que muchas aplicaciones tuvieran su propia y única interfaz de usuario, definiendo los comandos básicos y dejándolos visibles mientras se usaba una aplicación en particular. Pero todo esto podría ser totalmente transparente para el usuario. Es decir, si por ejemplo, se quería mover un conjunto de archivos de un lugar a otro utilizando un ícono, no era tan necesario mostrarle como lo estaba haciendo el sistema. Pero fue un buen intento para lograr utilizar gráficos.

Las GUI's se conocen aproximadamente desde 1982 cuando las primeras computadoras personales Macintosh fueron lanzadas al mercado, ya que fueron las primeras en usar íconos. Macintosh introdujo el concepto de ***interfaz común de usuario***[2] , con la que logró una gran audiencia. La interfaz común del usuario que define la GUI usa menús desplegables y cajas de diálogo, cumple con sistemas de ayuda, habilita al usuario a explorar la aplicación y literalmente él descubre comandos que le ofrecen nuevas y más complicadas funciones con sólo leer alguna documentación en un menú. Además, todas las aplicaciones tienen el mismo cuerpo o estructura; después de aprender a usarlo la primera vez, el usuario estará apto para ejecutar después cualquier aplicación.

Con todas estas ventajas que pueden ofrecernos las GUI's nos enfrentamos ahora a un nuevo reto; los sistemas no deben cumplir solamente con una muy buena y atractiva presentación para el usuario, sino que la pregunta es: ¿cuántas de estas aplicaciones se pueden ejecutar al mismo tiempo?. Todo esto llega a ser particularmente crucial cuando las aplicaciones en algunas ocasiones son tardadas y es necesario que se ejecuten otras. Es por lo tanto muy importante que el sistema permita manejar la multitarea donde se permite correr la misma aplicación en

[2] Una interfaz es el medio que permite la comunicación entre dos entidades. El concepto de interfaz común de usuario es un estándar que han adoptado varios fabricantes en su software, al tratar de tener el mismo principio para la creación de interfaces con el usuario.

diferentes ventanas y al mismo tiempo, o ejecutar dos aplicaciones diferentes en ventanas diferentes.

Se puede ver a continuación cómo en los últimos años, han ocurrido dos principales cambios en la evolución de la comunicación con las computadoras y de lo que ellas pueden hacer:

- las interfaces gráficas de usuario estándar han proporcionado mejores ambientes, en los cuales se han descubierto nuevos caminos para la interacción con el hardware, y
- los programadores quienes construyen aplicaciones han desarrollado y entendido de cómo los elementos de las GUI's pueden ser usados para desarrollar nuevo software que interactúe con el usuario en caminos diferentes.

La mayoría de los programadores piensan que el término GUI es un pequeño conjunto de elementos integrados para realizar el reconocimiento de algún comando. El *ratón*, menús desplegables, ventanas y cajas de diálogo forman una GUI. Agregando color y figuras, el ambiente se hace más atractivo para el usuario. Una interfaz gráfica de usuario está basada en los siguientes cuatro puntos:

1. Una interfaz común para el usuario.
2. Alta resolución y despliegue de color.
3. Lo que tu ves es lo que tu obtienes (What You See Is What You Get, WYSIWYG), este punto es muy importante ya que el usuario espera que se ejecute lo que él está viendo.
4. Manipulación directa.

Para manipular o navegar en el mundo de una GUI se requieren de tres cosas:

1. El hardware que hace posible la realización de los cuatro puntos anteriores.
2. Un sistema de software que haga fácil el desarrollo de aplicaciones para así también construir aplicaciones gráficas.
3. La alta disponibilidad de aplicaciones escritas para esto.

Hoy en día, las GUI's son aceptadas como estándar en los tres ambientes más populares del mercado: las PC's con Windows y OS/2, las MAC con su propio sistema operativo, y UNIX con su propio sistema en las estaciones de trabajo.

Algunos Pro's de las GUI´s son:

- Proveen de una interfaz estándar para los usuarios. Cada programa usa el mismo método de selección de opciones (apuntar y hacer click).
- Los programas trabajan unos con otros. Esto es fácil para mover una figura que, por ejemplo, representa un paquete de dibujo a una ventana de un procesador de palabra; deben por eso estar en contacto para saber de donde se parte y a donde se llega, la liga entre las funciones es muy importante.
- Solamente las GUI's necesitan tener un dispositivo instalado. Por ejemplo, una impresora, pantalla de video y un driver de *ratón* son necesarios para manipular la GUI, todos los programas pueden acceder a estos dispositivos.
- Las GUI's permiten utilizar el servicio de multitarea.

Algunos Contras de las GUI's son:

- Requiere de hardware relativamente rápido para manejar despliegues y actualización de gráficos.

- Grandes dispositivos de almacenamiento, ya que la cantidad de información que se maneja es muy grande.
- Monitores de alta resolución (preferentemente a color) para el despliegue de íconos.

2.1.1 El sistema de ventanas X-Window

Hasta hace muy poco tiempo, todo usuario de una computadora necesitaba aprender un sinfín de mandatos en línea para comunicarse con ella. Sin embargo, ahora estamos frente a un cambio en el mundo de la comunicación usuario-computadora: en el campo del software tenemos el surgimiento de interfaces gráficas entre el usuario y la computadora y, en el campo del hardware, tenemos el surgimiento de las estaciones de trabajo y de las terminales gráficas, llamadas también terminales X.

Otra gran tendencia, en el aspecto del software y hardware, es hacia la conectividad de equipos, es decir, hacia la integración de redes de computadoras.

En un ambiente de redes de computadoras tenemos a los usuarios finales ubicados en alguna parte de la red, equipados con estaciones de trabajo, y en otra parte están los equipos principales o mainframes, impresoras láser, lectores de documentos (scanners) y todos los periféricos a compartir entre todos los usuarios. En cada una de las estaciones de trabajo se tiene la posibilidad de contar con una Interfaz de Usuario Gráfica (GUI) como medio de comunicación entre la máquina y el usuario.

En este contexto, el usuario final puede trabajar en su estación de trabajo con diversas ventanas a la vez, las cuales, inclusive, pueden estar controladas por diversas máquinas de la red e intercambiar información proveniente de una máquina remota con otra máquina también remota.

Por ejemplo, en una ventana puede tenerse el despliegue del resultado de cálculo realizado por una aplicación corriendo en una máquina VMS; en otra ventana, el resultado de una aplicación corriendo en IBM RS/6000 y, en otra, el resultado de los cálculos locales en una SUN. Es decir, podemos tener una conectividad de aplicaciones.

Actualmente, esta configuración es posible con el concepto llamado: *Sistemas Abiertos*, tendencia que trata de realizar un ambiente de trabajo donde los diferentes componentes de software de un sistema informático puedan proporcionarse por diferentes proveedores, y funcionen en diferentes plataformas.

En el componente de los sistemas operativos se está proponiendo el UNIX y en sistemas de ventanas, el sistema *X-Window* con sus principales tipos de GUI, MOTIF y OPEN LOOK.

Un poco de Historia

Desde la aparición en los revolucionarios estudios en la XEROX, que dieron las bases de la interfaz Macintosh, hasta la época actual de las terminales gráficas, la comunicación con el sistema operativo se ha simplificado enormemente. Ahora la comunicación con el equipo se realiza por medio de acciones con el *ratón*, con sus botones y su movimiento. Una acción se realiza seleccionando primero un objeto en la pantalla, el cual está representado por un ícono.

La selección se realiza moviendo el ratón y ubicando el ícono del ratón, generalmente una flecha, sobre el objeto a seleccionar. Después, la acción se ejecuta cuando oprimimos un botón del ratón un cierto número de veces. Esta comunicación gráfica está evolucionando al mundo del software.

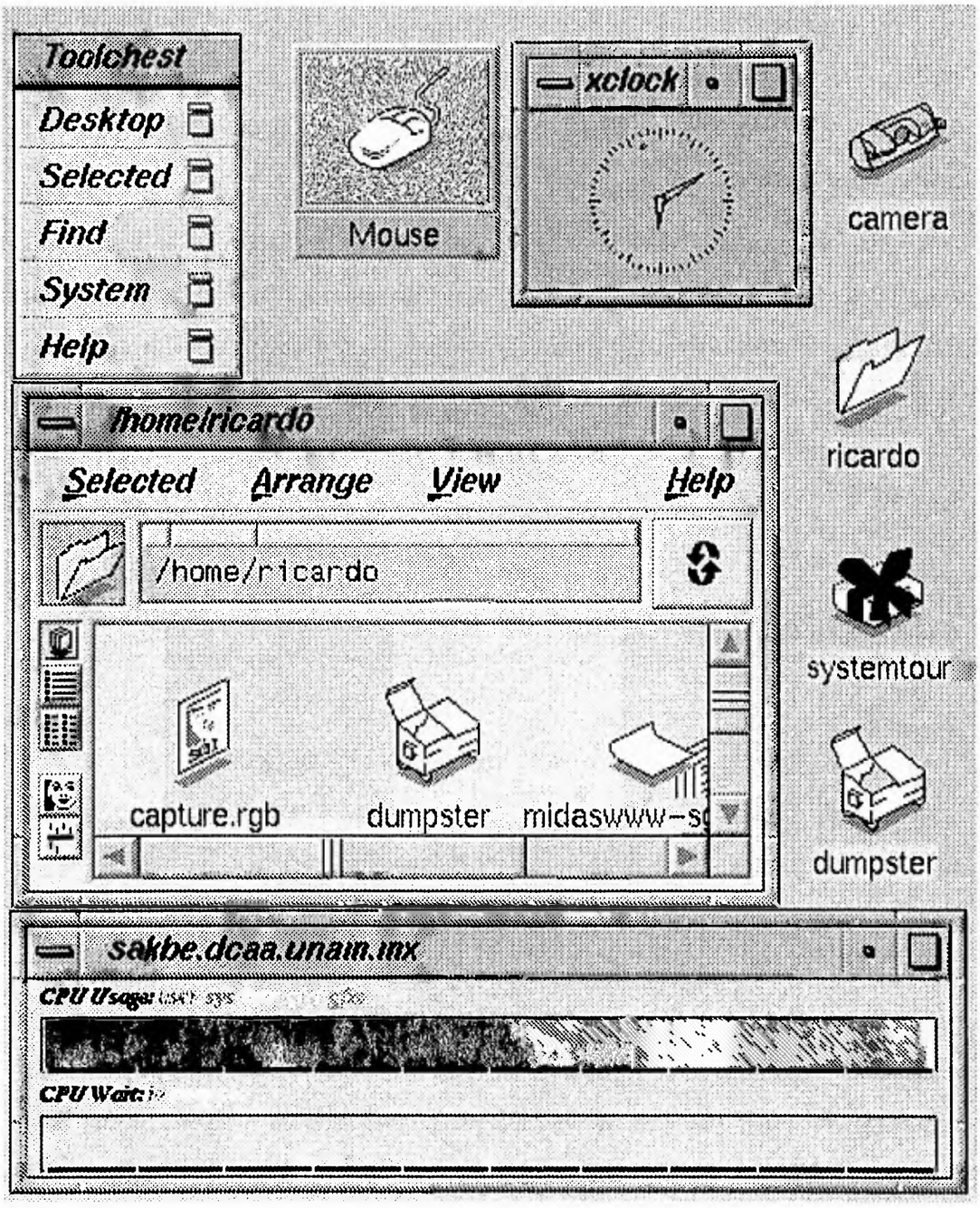


Figura 2.2 El sistema de ventanas X-Window

En el mundo de lo sistemas operativos más utilizados actualmente (MS-DOS para las computadoras personales y UNIX para casi todo tipo de computadora), ahora se cuenta con interfaces gráficas: el programa Windows de Microsoft (para MS-DOS) y X-Window en UNIX. Estos programas están cambiando la forma de trabajar con la computadora.

En el mundo UNIX, al sistema de ventanas también se le conoce como sistema X. *X-Window* (ver Figura 2.2) es un sistema de comunicación con la máquina que consiste de un sistema de ventanas en red que permite la construcción de redes en sistemas heterogéneos. El sistema X debe pensarse con un servicio de red que utiliza la tecnología cliente-servidor. Este servicio es parte de una interfaz de usuario.

2.2 Arquitectura cliente/servidor

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, que 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.

Definición de 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.

Un *servidor* es un administrador de recursos. Un *recurso* puede ser identificado como algo físico (por ejemplo, una impresora) o abstracto, para el caso de una base de datos. Un cliente es un usuario de los recursos que administra el servidor.

Bajo este esquema, se reparte el proceso de una aplicación entre un *front end* (el cliente, que puede ser una PC o una estación de trabajo) y un *back end* (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 seguirá 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 servidor o puede manipular sus propios datos.

2.2.1 Características del cliente

En un sistema cliente/servidor, un cliente es un proceso que interactúa con el usuario, observando las siguientes características:

a) Presenta la interfaz al usuario (UI).

Esta interfaz permite al usuario introducir sus consultas para recuperación y análisis de datos, así como recibir los resultados de dichas consultas, típicamente en un ambiente gráfico (GUI).

b) Forma consultas o comandos en un lenguaje predefinido, para su presentación al servidor.

El cliente y el servidor pueden usar un lenguaje estándar, como el SQL (Structured Query Language), o un lenguaje propietario predefinido.

c) Se comunica con el servidor por medio de una metodología de comunicación de procesos determinada y transmite consultas o comandos al servidor.

Un cliente ideal hace que esta comunicación con el servidor sea transparente al usuario.

d) Realiza análisis de datos sobre los resultados de la consulta o comando que regresan del servidor y lo presenta al usuario.

La naturaleza y el grado de análisis que se ejecuta en el cliente puede variar de un sistema a otro.

Las características (b) y (d) marcan la diferencia entre cliente y las terminales tontas que se encuentran conectadas a un host, dado que los clientes, como ya vimos, deben poseer capacidades de procesamiento.

Por otra parte, la última característica no debe ser confundida con el tipo de procesamiento que se da en una red local, donde se toman del servidor de archivos todos los elementos necesarios para ser procesados localmente y responder al requerimiento del usuario.

Existen además elementos clave que deben ser considerados en relación al cliente:

- *Sistema Operativo de la estación de trabajo.*

  Se deben diseñar sistemas cliente/servidor que soporten diversos sistemas operativos en las estaciones de trabajo, como lo es Unix.

- *Consideraciones respecto al hardware.*

  El cliente debe ser tan pequeño como sea posible para que pueda ser soportado por estaciones de trabajo.

- *Consideraciones respecto a la conectividad.*

  Estas consideraciones influyen también en el diseño del software del cliente. Un diseño modular asegurará que las aplicaciones en los clientes sean portables a diversas plataformas tanto de software como de hardware.

- *GUI.*

  La interfaz al usuario es definida, principalmente, por el objetivo del sistema cliente/servidor y por el sistema operativo de la estación de trabajo. En general, una interfaz al usuario debe ser lo más modular posible para que el cliente pueda pasar de un sistema operativo a otro.

2.2.2 Características del Servidor

En un sistema cliente/servidor, un servidor es un proceso o conjunto de procesos que deben existir en un equipo que da servicio a uno o más clientes.

Tiene las siguientes características:

1. Un servidor da servicio al cliente.

La naturaleza y el grado de este servicio es definido por el objetivo del sistema cliente/servidor.

Asimismo, un servicio puede requerir un mínimo de computación en el servidor, como es el caso de los servidores de archivos o de los servidores de

impresión, o necesitar de procesamiento intensivo, como se da en los servidores de bases de datos.

2. Un servidor solamente responde a las consultas o comandos de los clientes.

Esto es, ningún servidor inicia la conversación con un cliente, tampoco atiende directamente interfaces con el usuario final. Simplemente actúa como repositorio de datos (servidor de archivos), o de conocimiento (servidores de bases de datos) o como prestador de servicios de impresión.

Sin embargo, un servidor sí puede iniciar la conversación con otro servidor, solicitándole un servicio que a su vez le permitirá atender el requerimiento de un cliente. Esto, por supuesto, debe ser transparente al usuario.

3. Un servidor ideal hace transparente todo el esquema cliente/servidor al cliente y al usuario.

Un cliente que se comunica con un servidor no tiene porque estar enterado de la plataforma de hardware y software que intenta acceder, así como de la tecnología de comunicación (hardware y software) que hace posible ese enlace.

Para ello, es deseable y recomendable que en un ambiente de servidores múltiples, los servidores se comunican entre sí para proporcionar un servicio al cliente sin que éste conozca de esta múltiple existencia, ni de la comunicación entre servidores.

Así, una arquitectura cliente/servidor divide a la aplicación en procesos separados que corren en distintas máquinas enlazadas por una red. Es por ello que el

diseñador de aplicaciones divide las tareas en subtareas a ser llevadas a cabo ya sea por el cliente o por los servidores, teniendo como únicas limitantes las facilidades que le ofrezca el sistema operativo de red, así como las reglas de la organización.

Entre más avanzado sea un sistema operativo de red, las aplicaciones serán más pequeñas y fáciles de desarrollar en menos tiempo. Además, el hecho de que en el servidor resida la información de la organización permite incrementar la seguridad, pues se establecen mejores controles de acceso a la misma. Existen además elementos clave que deben ser considerados en relación con el servidor:

- *Escalabilidad.*

  Un servidor debe ser escalable, es decir, debe soportar números crecientes de clientes. Por ejemplo, en el caso de los servidores de bases de datos, el diseño debe permitir el crecimiento a un sistema de bases de datos distribuidas.

- *Interfaz con el servidor.*

  Es importante que las características del servidor (tanto en software como en hardware), sean transparentes a los clientes, excepto en lo que se refiere a la interfaz estándar de acceso a sus servicios, como puede ser SQL, lo que además protege a los clientes de cambios en la tecnología.

2.2.3 Ventajas y desventajas sobre otros modelos

Un sistema cliente/servidor ofrece soluciones a las desventajas de los clientes centralizados tanto en mainframes como en servidores locales, descritos anteriormente, al tener las siguientes características:

- *Es inteligente a nivel de equipo de escritorio, ya que el cliente es el responsable de la interfaz con el usuario.*

  El cliente transforma las consultas o comandos del usuario a un lenguaje predefinido que es comprendido por el servidor y presenta los resultados que el servidor le envía como respuesta, con lo que se obtiene mayor capacidad de proceso a un menor costo.

- *Permite compartir los recursos del servidor de manera óptima.*

  Estos recursos pueden ser tanto el procesador, como almacenamiento secundario y periféricos. Un cliente puede pedir al servidor que realice procesos grandes, o puede pedirle que corra grandes aplicaciones (servidores de bases de datos), y como fruto obtener solamente los resultados de ese proceso.

- *Utilización óptima de la red.*

  Dado que los clientes se comunican con el servidor a través de un lenguaje predefinido, como puede ser SQL, y el servidor sólo remite al cliente los resultados de la consulta o comando, reduciendo el tráfico en la red de los sistemas centralizados que transfieren los archivos de datos en su totalidad.

- *Permite cierta independencia sobre el sistema operativo y los protocolos de comunicación.*

  Esto facilita el mantenimiento de las aplicaciones y asegura su portabilidad.

- *Permite desarrollos más flexibles a un menor costo.*

  Hoy, la mayoría de los componentes de un sistema cliente/servidor, están disponibles comercialmente, provenientes de diversos proveedores, lo que da

a las organizaciones libertad de elección, sin por ello exentarlas de los desarrollos internos.

Con el crecimiento de sistemas cliente/servidor, los proveedores se dirigen por el camino de los sistemas abiertos, que son fáciles de integrar y que requieren de menor trabajo para ser incluidos al sistema de la organización.

Entre los puntos débiles de una arquitectura cliente/servidor, tenemos que existe poca experiencia con ella, parte del software disponible se encuentra en su versión *beta* y el ya liberado tiene algunos errores (bugs); además, las facilidades para la administración de los datos y herramientas para la seguridad de los mismos deben ser mejoradas con el fin de ofrecer sistemas confiables.

Sistemas cliente/servidor vs. sistemas de tiempo compartido

El procesamiento en tiempo compartido utilizando los grandes sistemas de cómputo (mainframe por ejemplo), no es siempre la mejor solución para todas las aplicaciones. Un buen ejemplo, se presenta con el advenimiento de los sistemas con interfaces gráficas, que demandan una respuesta instantánea del subsistema gráfico, requiriendo un procesador dedicado y muy posiblemente personal.

El poder de las nuevas estaciones de trabajo, que comúnmente cuentan con un monitor a color de alta resolución y que opcionalmente cuentan con capacidades de multimedia, hacen que los conceptos y las tendencias del cómputo se modifiquen.

Un sistema de tiempo compartido, provee a los usuarios un ambiente en el que se comparten los recursos, tales como el espacio en disco, las impresoras, los programas y datos. Para que se compartan estos recursos en distintas estaciones de trabajo, éstas tienen que estar unidas a través de una red de cómputo. Por ejemplo, en la UNAM se tiene una red con un gran número de estaciones de trabajo, en la que

los usuarios tienen que aprender a diferenciar entre lo local y remoto, saber en que máquina se tienen tales y cuales recursos y programas disponibles. Esto nos lleva a encontrarnos con una serie de problemas.

El problema de administración de la red y de equipos conectados a ella, se convierte en un problema enorme. En el ambiente de estaciones de trabajo, cada usuario debe ser operador, administrador y programador de sistemas. Esto debido obviamente, a que ni los administradores de sistemas, ni los operadores pueden hacerse cargo de 100 máquinas al mismo tiempo. En un sistema de tiempo compartido (mainframe por ejemplo) se ha llegado a un grado de madurez en estos aspectos: un equipo de trabajo se dedica a la operación del sistema, otro a la administración y en otro equipo se desarrollan los programas de aplicación necesarios para las labores de los usuarios finales.

Se han implantado a la fecha algunas soluciones a estos problemas, pero no existe una solución tan adecuada como a la que se llegó en un tiempo con los sistemas de tiempo compartido. Se tienen ya comandos para copias de archivos en una red, y aún mejor, existen posibilidades de manejar archivos remotos. Desafortunadamente en la mayoría de los casos, el usuario debe estar consciente de la diferencia entre un sistema local y uno remoto. El problema radica en que los sistemas operativos de hoy no se diseñaron para ser *sistemas operativos distribuidos*.

2.2.4 Ejemplos

Servicio de datos estructurado

Uno de los mejores ejemplos de la arquitectura cliente/servidor, está dada en un ambiente de base de datos, en donde el cliente es responsable de la interacción humana y formulación de consultas y actualizaciones a la base de datos. SQL

(lenguaje de consulta estructurado) será el lenguaje que interactuará entre esas consultas y actualizaciones con el servidor.

X Window. Sistema de clientes y servidores

Como se discutió en la sección 2.1.1, el papel del sistema X Window es el administrar los recursos físicos en la máquina del usuario, por ejemplo los subsistemas gráficos y los dispositivos de entrada. El sistema corriendo el servidor X, recibe como entrada estándar el teclado, *ratón* o cualquier otro dispositivo, la cual es producida en las formas especificadas por el protocolo X y pasadas al proceso del cliente apropiado. El cliente puede tener un papel general, por ejemplo, como administrador de ventanas que pueda controlar el tamaño, localización y visibilidad de las mismas, o una función muy específica, como la copia de un archivo de un disco a un servidor X..

Servicios del Deamon Internet (inetd)

Dentro de la familia de protocolos de Internet, existen cientos de ellos que ofrecen servicios, partiendo de los más fundamentales hasta los generalmente utilizados como el protocolo de transferencia de archivos FTP.

Sin *inetd*, un sistema podría prácticamente soportar solo un pequeño subconjunto de estos servicios (esto es cierto porque la mayoría de ellos necesitan estar corriendo para responder las peticiones de los clientes). Con inetd, existe un archivo de configuración, /etc/services, que es una lista de todos los servicios soportados por el sistema, los números de puertos de TCP para establecer la comunicación cliente/servidor, y el programa asociado a este servicio.

2.3 Multimedia

En el mundo dinámico de nuestros días, el volumen y la complejidad de la información que se necesita transportar, comprender y utilizar está en una etapa muy importante de crecimiento. Los recursos y medios tradicionales tales como transparencias, videos y películas, no son ya suficientes para transmitir información e ideas completamente.

Con la llegada de las computadoras personales y tecnologías de multimedia, tenemos disponibles nuevos métodos para transmitir ideas. Esas soluciones integran texto, gráficas, animación, audio y video para transmitir y proveer información.

Desde la década pasada, hemos sido testigos de los grandes cambios tecnológicos en las siguientes áreas de la tecnología de información: computación y comunicaciones. Presenciamos la llegada de las computadoras personales, el mundo de las redes, medios de almacenamiento masivo, tecnologías de video interactivas, tecnologías de imágenes y gráficas, así como de digitalización de voz y video y el incremento masivo de bases de datos tanto públicas como privadas. Las actuales tecnologías de información tienen capacidad para proveer a investigadores, estudiantes y público en general, un sistema dinámico para el acceso inmediato a la información.

Gracias a las presentaciones de multimedia, los expositores interactúan con su auditorio en formas que eran hasta hace poco tiempo inimaginables. Por ejemplo, si el público está interesado en un tema que sale de la estructura de la presentación, el expositor oprime un botón y automáticamente pasa a otro segmento de la misma. Si un cliente pregunta sobre un tema que abordó antes, el expositor puede regresar casi instantáneamente al punto de interés sin tener que revisar meticulosamente 30 diapositivas diferentes antes de encontrar la adecuada.

Las presentaciones en multimedia no sólo sirven para que el expositor dé respuestas claras a las preguntas que plantea el auditorio, sino que gracias al audio, el video y animaciones, el público se compenetra más en el tema de la presentación.

Es un hecho que en los últimos años, el uso de las computadoras personales ha gravitado hacia el área de la comunicación. Por su parte, la industria informática ha vuelto los ojos hacia la conectividad y nuevas formas de comunicación entre la gente, tanto en el trabajo como en el hogar.

Si hace un año cualquier usuario deseaba saltar a la plataforma de multimedia (donde se usa video, audio y computación), la inversión mínima para lograrlo sumaba varios miles de dólares, lo cual traducido a equipo significaba incorporar a una PC avanzada, tarjetas de video, audio y fax-modem, bocinas y finalmente esperar a que el conjunto de todos estos elementos fueran compatibles entre sí y con la PC.

2.3.1 Definición de Multimedia

Algunos etimologistas querrán llegar siempre hasta la raíz de lo que significa la palabra mutimedia, la cual se deriva de *multi* (muchos) y *media* (medios), sus primeros usos como concepto surgieron en la industria de la computación cuando se comenzó a describir una integración de sonido y animación en las Macintosh y PCs. Hoy en día la palabra multimedia ha pasado a formar parte de nuestro vocabulario habitual.

Cuando se habla de la palabra *multimedia*, el primer concepto que viene a la mente de las personas es el de software. Sin embargo, de poco sirve un buen paquete de multimedia si no contamos con el hardware adecuado.

En esencia, multimedia es la tendencia de mezclar diferentes tecnologías para difundir información, el recurrir a varios sentidos a la vez para lograr un efecto

Las presentaciones en multimedia no sólo sirven para que el expositor dé respuestas claras a las preguntas que plantea el auditorio, sino que gracias al audio, el video y animaciones, el público se compenetra más en el tema de la presentación.

Es un hecho que en los últimos años, el uso de las computadoras personales ha gravitado hacia el área de la comunicación. Por su parte, la industria informática ha vuelto los ojos hacia la conectividad y nuevas formas de comunicación entre la gente, tanto en el trabajo como en el hogar.

Si hace un año cualquier usuario deseaba saltar a la plataforma de multimedia (donde se usa video, audio y computación), la inversión mínima para lograrlo sumaba varios miles de dólares, lo cual traducido a equipo significaba incorporar a una PC avanzada, tarjetas de video, audio y fax-modem, bocinas y finalmente esperar a que el conjunto de todos estos elementos fueran compatibles entre sí y con la PC.

2.3.1 Definición de Multimedia

Algunos etimologistas querrán llegar siempre hasta la raíz de lo que significa la palabra mutimedia, la cual se deriva de *multi* (muchos) y *media* (medios), sus primeros usos como concepto surgieron en la industria de la computación cuando se comenzó a describir una integración de sonido y animación en las Macintosh y PCs. Hoy en día la palabra multimedia ha pasado a formar parte de nuestro vocabulario habitual.

Cuando se habla de la palabra *multimedia*, el primer concepto que viene a la mente de las personas es el de software. Sin embargo, de poco sirve un buen paquete de multimedia si no contamos con el hardware adecuado.

En esencia, multimedia es la tendencia de mezclar diferentes tecnologías para difundir información, el recurrir a varios sentidos a la vez para lograr un efecto

mayor en la comprensión del mensaje. Bajo esta definición, la televisión es un dispositivo de multimedia porque incluye video, audio, texto y animaciones en un sólo sistema de presentación. Pero al trasladar este concepto al mundo de la informática, la palabra multimedia significa la transmisión de mensajes a través de una computadora. Para que esta última o una aplicación (programa) sean consideradas multimedia, deberán integrar por lo menos tres de los siguientes cinco tipos de datos: texto, gráficas, imagen fija, imagen en movimiento y audio.

El principal punto de interés de este concepto se centra en las especificaciones técnicas de la computadora, refiriéndonos al hardware y al software, actualmente integran elementos de texto, gráficas, animación y sonidos desarrollados independientemente por varias fuentes en una sola interfaz para el usuario.

Es posible clasificar dos tipos de sistemas en multimedia: lineales e interactivos.

Los lineales -algunas veces conocidos como pasivos- son las presentaciones de multimedia en donde el usuario simplemente recibe información, instrucciones o se trata de algún programa de entretenimiento, sin que éste tenga control alguno sobre el contenido de la presentación. El ejemplo más claro de este tipo de sistemas es la televisión.

Los sistemas de multimedia interactivos, son aquellos que permiten al usuario participar activamente en la aplicación, regularmente este tipo de sistemas se desarrollan con ayuda de una GUI (Graphical User Interface).

No debemos olvidar que con toda esta tecnología de la cual hablaremos más adelante, se podrá obtener información así como establecer nuevos medios de comunicación en formas nunca antes logradas.

2.3.2 Tecnologías

El primer paso para integrar una plataforma multimedia es la necesidad de integrar componentes de hardware compatibles que producirán los resultados deseados en una computadora.

La plataforma básica de multimedia debe ser capaz de manejar texto, animación gráfica, imágenes, video y sonido. Esos elementos pueden ser integrados y manipulados en varias condiciones sobre pantallas individuales.

Algunas computadoras pueden manejar los elementos antes mencionados por medio de un software especializado. Sin embargo, las fotografías, video y audio deben ser digitalizadas antes de que estos elementos puedan ser procesados y manipulados en un proceso de cómputo. Esto plantea dos problemas básicos de integración :

1) El video y audio existen electrónicamente en forma de señales analógicas o digitales que difieren de los estándares fijados por la industria de computadoras; y

2) Las imágenes en movimiento requieren de medios de almacenamiento masivo y deben ser procesados a grandes velocidades para generar efectos comparables a los de la televisión.

Estos problemas son resueltos por varios dispositivos que:

1) Digitalizan elementos de multimedia para hacerlos compatibles con los estándares fijados por las computadoras.

2) Utilizan técnicas de compresión y descompresión acorde a los requerimientos de velocidad. Esta solución incluye microchips especializados que pueden encargarse de esta tarea.

Acerca de la compresión de video

La compresión en el video reduce la cantidad de datos requeridos para reproducir las imágenes. Minimiza el espacio en disco, incrementa la velocidad de acceso, y permite guardar una imagen digital en movimiento con una calidad aceptable. Todas las técnicas de compresión explotan la redundancia digital de un video, donde la misma información de la imagen es transmitida repetidamente. Por ejemplo, puede haber redundancia entre pixeles sobre una pantalla donde los pixeles adyacentes tienen el mismo color los cuales pueden ser registrados vertical y horizontalmente, hay redundancia entre líneas adyacentes, cuando el movimiento sólo ocurre en una pequeña porción de la pantalla, hay también redundancia considerable entre esos movimientos sobre el tiempo.

La investigación sobre la tecnología de compresión en video ha estado bajo estudio desde hace 25 años aproximadamente, y muchas técnicas se han llevado a la práctica. Estas técnicas de compresión son medidas por cuatro criterios básicos: radio de compresión, calidad de la imagen, velocidad de compresión y descompresión, así como los recursos de hardware y software requeridos para llevarse a cabo.

El procesamiento de una sola imagen en un segundo o menos es generalmente aceptable, mientras que el video requiere por lo menos 15 marcos por segundo para producir el efecto de movimiento.

Acerca del audio digital

En este caso, el audio primero debe ser convertido de forma analógica a digital usando algún convertidor A/D. El proceso de digitalización, similar a las señales de video analógicas, involucra muestreo y cuantización en intervalos de tiempos específicos. Varias técnicas de compresión han sido desarrolladas para reducir este tamaño. Sin embargo, es un poco difícil debido a que este tipo de señales no tienen una estructura (señales aleatorias), y no se tiene la misma oportunidad de usar el concepto de redundancia para llevar al cabo la compresión.

Integrando todos los recursos

El hardware que se utiliza en las presentaciones multimedia se agrupa en dos categorías: desarrollo y reproducción. Al elaborar presentaciones de multimedia, el equipo que se necesita depende del tipo de plataforma en que se va a trabajar. Para las presentaciones multimedia se requiere de mucho espacio de almacenamiento. Por ejemplo, unos cuantos minutos de video absorben algunos megabytes de espacio en disco.

El hardware necesario para reproducir la presentación en multimedia depende del lugar en que el usuario vaya a hacer dicha presentación.

Aunque el objetivo es proveer aplicaciones para usuarios finales con una plataforma interactiva de multimedia, o el de manejar un equipo para desarrollar tales aplicaciones, es recomendable establecer una estrategia de multimedia adecuada, porque los diferentes dispositivos tanto de entrada como de salida deben ser integrados, probados e implantados para trabajar simultáneamente. De otra manera será difícil obtener el máximo funcionamiento de estas tecnologías, es

recomendable tener en cuenta dicha estrategia para ser implantada en el proyecto adecuado.

Una estrategia de multimedia ya establecida, también atrae a otros grupos corporativos cuyas actividades pueden ser cruciales en la difusión de información. Esos grupos pueden estar involucrados en el desarrollo de presentaciones de mercado, entrenamiento corporativo, videos para anuncios comerciales y promocionales, procesamiento de imágenes, sistemas expertos e inclusive desarrollo de películas. Todas estas entidades pertenecen a los diferentes medios que podrían ser apropiados en las aplicaciones de multimedia.

Ensamblar una aplicación en multimedia, requiere de una evaluación tanto de hardware como de software enfocada a las necesidades del sistema. Dicha evaluación involucra criterios de selección del equipo a utilizar dependiendo de la aplicación, por ejemplo si se trata de un kiosco de información.

Hardware de entrada

Este tipo de hardware incluye dispositivos físicos tales como: teclado, ***ratón*** (mouse), pantallas sensibles al tacto (touchscreen), lápiz óptico, por mencionar algunos. Los scanners gráficos son utilizados para capturar y digitalizar imágenes convertidas en archivos aplicables a multimedia. Las cámaras digitales son útiles en la creación de imágenes personalizadas de documentos, equipos, personas, etc., que pueden también estar almacenadas en archivos para su uso posterior.

El micrófono, audio cassettes o teclados acústicos son usados en la creación de música y sonidos que serán utilizados directa o indirectamente en grabaciones de audio para su integración en las aplicaciones.

El video en movimiento con o sin sonido asociado es capturado vía videocintas (videotape) por ejemplo. Sin embargo, tales videos podrían ya existir en

CD-ROMs, videodiscos, o posiblemente estar capturados en tiempo real vía receptores de televisión o emisión satélite.

No debemos olvidar que la calidad de la salida final generalmente depende de la calidad de los dispositivos de entrada.

Hardware de salida

El más popular dispositivo de salida en el desarrollo de aplicaciones multimedia es un monitor VGA. Existen varias alternativas de resolución para este tipo de monitor, este dispositivo es considerado como un elemento muy importante cuando se trabaja con imágenes, gráficas y video.

Dependiendo de la naturaleza de la aplicación, los desarrolladores de multimedia deben estar enfocados hacia el tipo de usuario final, tales como monousuarios, grupos, simposiums o conferencias. Los dispositivos de entrada y salida más comunes se encuentran en la tabla 1.

Hipertexto

Cuarenta y tres años después de que el concepto de hipertexto fuera concebido por Vannevar Bush, las aplicaciones de este concepto cada vez han pasado más de los laboratorios de investigación al mercado. El hipertexto semeja a la habilidad del cerebro humano para acceder a información rápidamente y de manera intuitiva a través de una referencia. El hipertexto es un método para presentar información donde las palabras seleccionadas en el texto pueden ser "expandidas" en cualquier momento para proporcionar otra información sobre la palabra.

Dispositivos de Entrada	Dispositivos de salida
Teclado	Monitores
Ratón	
Touchscreen	Pantallas Gigantes
Lápiz Óptico	
Scanner	
Cámaras Digitales	Proyectores
Videodiscos	
CD-ROM	
Micrófono	Bocinas
Audiocassettes	Sistemas Stereo

Tabla 1

Esto significa que las palabras forman "vínculos" con otros documentos, que pueden ser texto, imágenes o cualquier otra cosa. Para un sistema de información electrónico en hipertexto, uno es capaz de acceder a información de una manera no secuencial en los archivos fácilmente. En un buen sistema de hipertexto, se tendrá la capacidad de hojear rápidamente una serie de documentos, así como la facilidad de poder moverse de la misma forma entre ellos. Aunque muchos sistemas de hipermedia son muy similares en cuanto a su arquitectura básica, estos varían mucho en cuanto a la manera que adoptan para realizar búsquedas de información, en cuanto a la cantidad de usuarios que lo van a estar utilizando y a los tipos de tareas a las cuales están dirigidas. Por ejemplo, cuando son clasificados por la cantidad de usuarios existen por lo menos las siguientes cuatro clasificaciones:

- Sistema monousuario
- Sistema multiusuario
- Sistema corporativo
- Sistema mundial

A partir del crecimiento de los sistemas de hipertexto, las limitaciones de compatibilidad se han venido incrementando actualmente, pero con la debida globalización de criterios de desarrollo de hipertexto esas limitaciones en el futuro podrán ir llegando a menos e integrar un sistema de hipertexto completamente compatible y amigable.

Hipermedia

Las aplicaciones de hipertexto han sido limitadas únicamente a los enlaces entre texto e información numérica, ocasionalmente un pequeño número de imágenes digitales. Sin embargo, dada la tecnología de cómputo que hoy en día existe, se ha llegado a la necesidad de contemplar enlaces hacia todo tipo de información. En lugar de limitarse al simple uso de texto, los usuarios tienen la facultad de enlazar datos, gráficas, videos, animación y voz. Es por eso que hipermedia extiende el concepto de hipertexto al poder ligar todo tipo de material que pueda ser digitalizado para el almacenamiento del mismo en cualquier sistema de computo, y a su vez ser recuperados mediante el uso de sistemas de manejo de imágenes, sonido, gráficas y animación.

En la Figura 2.3 se muestra una configuración típica de hipermedia en términos de hardware y software, sistemas de comunicaciones y sistemas de información. Los componentes de éste último, también muestran cómo varios tipos y

formatos para almacenar imágenes, gráficas y audio para aplicaciones en multimedia de información están almacenados así como los medios ópticos que son usados.

Definiremos a un sistema de hipermedia como aquel que utiliza tecnología disponible para facilitar aquellas actividades significativas encaminadas al incremento en la eficiencia de la interacción entre gente y material relacionado al conocimiento.

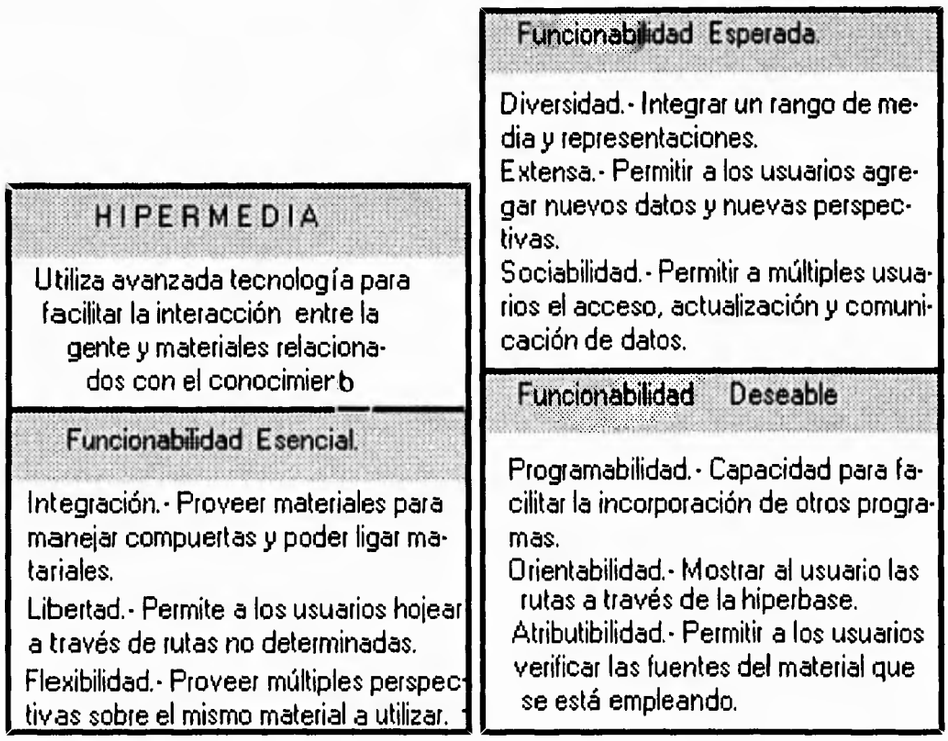


Figura 2.3 Configuración típica de Hipermedia

2.3.3 Aplicaciones en Kioscos de Información

Factores poderosos están impulsando sistemas de multimedia interactivos al mercado. Cada factor representa el interés de grupos de industrias específicos que perciben a la multimedia como un medio para promover y desarrollar nuevos

productos y capturar nuevos mercados. Los factores que está manejando la multimedia interactiva incluyen:

- Avance rápido en tecnologías de hardware
- Desarrollo de nuevo software

Un camino más preciso para examinar los mercados de multimedia interactivos es categorizándolos por plataformas de computadoras usadas para desarrollar tales aplicaciones. Esos rangos abarcan hasta las más poderosas estaciones de trabajo operadas por desarrolladores de multimedia y kioscos en mercadotecnia.

Los kioscos son mencionados para referir a información en casillas. Muchos de ellos están desarrollándose en aplicaciones utilizando *touchscreen*[3] . Los kioscos más viejos casi siempre destacaban información en cintas de video o audiocassettes. Un buen ejemplo de un kiosco en mercadotecnia, es una pantalla de información interactiva la cual provee a los viajeros información de actividades regionales del lugar a visitar, así como la comida típica del lugar e información de restaurantes y hoteles. Estas unidades integran sonido y *touchscreen* para manejar el sistema.

Algunos kioscos manejan la opción de llamadas telefónicas para tener servicio de cuarto tales como comidas, así como realizar reservaciones de cuartos de hotel conectándose directamente a una operadora.

A continuación presentamos algunos ejemplos de multimedia aplicados a kioscos de información.

[3] Pantalla del monitor que responde al ser tocado directamente con el tacto sin necesidad de teclado u otro instrumento.

Mercadeo interactivo

También conocido como promoción electrónica de ventas al por menor, esta aplicación básicamente consiste en un kiosco que incluye una pantalla *touchscreen*, una computadora, un reproductor de video disco y una impresora para impresión de instrucciones, confirmaciones de compra y consejos del producto. Estos kioscos son presentados en atractivas cajas y en lugares comerciales de alta concurrencia de gente para llamar la atención del cliente, tales como farmacias, supermercados, centros de información al cliente, simposios y conferencias.

Los kioscos son utilizados para demostrar productos, proveer información, reproducir situaciones relacionadas con el producto o inclusive interactuar con clientes para el diseño de productos y su posible comercialización. Otros kioscos también tienen la facultad de ofrecer vales de descuento, recibir información de nombres de posibles clientes por correspondencia, aceptar órdenes de compra y procesar transacciones de tarjetas de crédito.

La principal justificación para el empleo de esta tecnología consiste en la gran reducción de costos en cuanto a la inversión para poder detectar posibles prospectos de clientes potenciales sin ser necesariamente compradores inmediatos, es decir, que se puede inferir en quién podría llegar a comprar y quién no.

En muchos de los casos los kioscos ofrecen mejor y más consistente información que la que pudiera dar el personal de ventas; de hecho los kioscos pueden mejorar las comunicaciones. Ellos cuentan con la ventaja de poder trabajar las 24 horas del día sin descanso y sin costo adicional, pueden ser fácilmente modificados y reprogramables, y pueden ser diseñados para ser manejados inclusive por gente no muy calificada, y por lo tanto ahorrar valioso tiempo para el personal de la compañía.

Kioscos en tiendas departamentales

Excelentes ejemplos de kioscos de mercadeo se pueden observar en las 500 unidades distribuidas en las tiendas de zapatos de Florsheim, 95 de las cuales están localizadas en los concesionarios de Florsheim en las tiendas Sears. Los clientes pueden utilizar los kioscos para realizar sus compras de zapatos escogiendo la talla, el modelo y estilo mediante el simple contacto de la pantalla del kiosco y siguiendo las instrucciones de la voz que está previamente grabada en el mismo. El sistema incluye un teclado que permite que los clientes tecleen su nombre, dirección y datos de su tarjeta de crédito, para que de esta manera se puedan entregar en su mismo domicilio. Las tiendas de Florsheim con kioscos han reportado un aumento en las ventas de zapatos en hasta un 20%, no sólo a través del sistema automatizado, sino también han permitido que los vendedores atiendan a los clientes más rápido y versátilmente.

Otro kiosco fue desarrollado por Weyerhouser para brindar asesoría a las personas que requerían un mejoramiento de sus hogares, estos kioscos fueron instalados en alrededor de 100 tiendas de artículos para el hogar. En ellos, los clientes pueden interactuar con el sistema para poder diseñar ellos mismos lo que deseen construir y poderlo modelar previamente en el sistema, el cual también proporciona el presupuesto del proyecto diseñado por el cliente e imprime una lista de los materiales que se necesitarían para el proyecto.

Durante el año que llevan de operación los kioscos, los centros comerciales de este tipo han aumentado en forma considerable sus ventas. Algo similar ocurre en los kioscos desarrollados por Servistar, los cuales aconsejan a los clientes en cuanto a las alternativas y cantidades recomendadas para pintar algo, así como técnicas de pintura.

La máquina automática de fotografiar es otro ejemplo de un kiosco interactivo. En él, los clientes son filmados de tal manera que el cliente escoge el momento preciso para ser fotografiado. Puede pagar en este sistema en efectivo o con tarjeta de crédito. Los clientes también reciben cupones de descuento mientras se realiza el proceso. El servicio de fotografías mediante filme está disponible las 24 horas del día, y los precios son más bajos que en cualquier centro fotográfico.

Las cadenas deportivas de Herman, utilizan kioscos de multimedia para dar información a sus clientes de lugares recomendables para vacacionar, asociando los artículos deportivos más recomendados para poder practicar algún deporte en ese lugar.

Kioscos en el medio corporativo

En este medio los kioscos son vistos como elementos que tienen el principal objetivo de ahorrar costos administrativos, tal es el caso del departamento de Recursos Humanos, el cual tiene la tarea de proporcionar al personal la información de cómo satisfacer necesidades tales como: seguro médico, oportunidades de capacitación ofrecidas por la compañía, jubilación, etc. Una clase separada de kioscos de multimedia existen en las áreas en donde la seguridad es de mayor prioridad, pero muchos de estos sistemas son manejados mejor por sistemas de seguridad más elaborados.

La compañía de seguros de Nueva York, desarrolló un sistema interactivo de multimedia, para informar a los empleados prospecto acerca de la compañía, su historia, productos y oportunidades de desarrollo. El sistema utiliza tecnología Intel, y existen planes para poder expandir su uso hacia otras oficinas en Boston y Massachusetts. Bajo situaciones similares, la cadena de hoteles Marriot ha instalado una serie de kioscos en diversas universidades con el fin de conseguir personal para

la compañía. Estos sistemas incluyen texto, animación, gráficas, audio y video, y ofrecen a los estudiantes el conocimiento de la historia de la cadena de hoteles Marriot, oportunidades de empleo e información relacionada con el mismo fin. Los kioscos incluyen una computadora Macintosh, pantallas de video, discos láser Sony y las aplicaciones interactivas de multimedia que fueron desarrolladas utilizando el mismo sistema.

Multimedia en servicios del gobierno

En cuanto a servicios para el gobierno los kioscos se han ido desarrollando rápidamente. El departamento de vehículos del estado de California, desarrolló un sistema interactivo para conductores que quisieran renovar su licencia. Este sistema está actualmente siendo probado en algunas ciudades de California como son: San Mateo, Glendale, y Folsom.

Además en California se está desarrollando un convenio entre el gobierno de ese estado e IBM para poder distribuir una serie de kioscos por toda la ciudad de *Los Angeles* para brindar información acerca de los servicios de asistencia social y de salud entre otros. La revista *Information Week* reportó que el estado de California quiere que los kioscos de información sean tan comunes como el teléfono, para permitir a todos los ciudadanos el acceso de datos de cualquier tipo relacionado con el gobierno y los servicios del estado.

2.4 Sistemas distribuidos

En los años 70's, las redes de computadoras comenzaron a ser un aspecto importante en los sistemas de cómputo. Manejadas inicialmente en aerolíneas e

industria militar, los sistemas de cómputo fueron interconectados comenzando así, con las primeras operaciones remotas. Durante la década de los años 80's, los sistemas distribuidos comenzaron a ser un aspecto vital en muchos sistemas de cómputo. La gente comenzaba a depender de los sistemas conectados en red para sus transacciones financieras, registros y correo electrónico entre oficinas. Esta dependencia fue incrementándose rápidamente y con ella, las limitantes del trabajo en red. La década de los años 90's brincará completamente a un esquema de red donde los sistemas distribuidos serán muy importantes; la gente podrá acceder a recursos de cómputo independientemente de su localización física.

En el proceso electrónico de información, el concepto de red implica la interconexión de computadoras. El tipo de conexión puede variar desde línea telefónica hasta enlace satelital. Su tamaño también puede ser variable, el cual puede ir desde un pequeño departamento hasta un edificio, e incluso llegar a una compleja red internacional con cientos de miles de usuarios usando sistemas diferentes. Usaremos la frase *cómputo distribuido* para enfatizar el concepto general de sistemas de cómputo interconectados, esta definición intenta incluir nociones del procesamiento distribuido en un modelo cliente/servidor, llamadas a procedimientos remotos, etc.

El concepto de red por lo tanto, es más que la interconexión de sistemas de cómputo. El valor de un sistema en red, depende de la capacidad de transferencia de información entre computadoras y la habilidad de producirla combinando los sistemas conectados. Referimos esta noción como parte en la definición de un *sistema distribuido*, aunque este último significa muchas cosas para la gente, nosotros usaremos este concepto para referirnos a la noción general del acceso de una computadora a los recursos de otra, por ejemplo; discos, impresoras, etc., el usuario no distingue entre operaciones locales y remotas, los programas no necesariamente se ejecutan en la computadora donde el comando fue especificado.

Los sistemas distribuidos y redes de computadoras tienen sus orígenes en la comunicación de datos. La tarea de transferencia de información hacia un equipo de cómputo central demandaba una solución electrónica en la década de los años 50's, varios mecanismos fueron desarrollados, entre ellos las primeras terminales. Posteriormente esos desarrollos permitieron a los sistemas, comunicarse con otros a grandes velocidades de transmisión de datos, dando origen a las primeras redes de computadoras con las topologías y protocolos de comunicación que actualmente conocemos. Los sistemas distribuidos, sin embargo, no aparecieron sino hasta la década de los 80's, donde emergieron nuevas tecnologías y el precio de las computadoras fue más económico, aparecieron las primeras computadoras personales y estaciones de trabajo, cada una de ellas contaba con su propio microprocesador de tal forma que se podían conectar con casi cualquier sistema de cómputo en la red, con la ventaja adicional que este tipo de sistemas no dependen de una computadora maestra para que los controle, sino que pueden trabajar independientemente, por lo tanto, fueron reemplazando a las terminales simples. Por otra parte, la introducción de las primeras GUI's y el *ratón*, dieron origen a un nuevo conjunto de usuarios, quienes comenzaron a disfrutar de un tiempo de respuesta consistente e independencia asociada en sus propias computadoras personales, silenciosamente demandaban los recursos de los grandes sistemas. La compatibilidad de recursos llegó a ser un objetivo de los diseñadores de sistemas, y los avances ocurrieron en muchas áreas; mecanismos de transferencia de archivos, ejecución de comandos remotos, entre otros.

La demanda de acceso remoto a recursos no es tarea fácil, las ventajas económicas asociadas con las estaciones de trabajo y computadoras personales no han extendido por completo este ambiente, el software requerido para administrar la distribución y compatibilidad de recursos es complejo. La cantidad de administradores de red, administradores de sistemas y desarrollo de software para

aplicaciones distribuidas determinan el paso de las innovaciones en los sistemas distribuidos.

Por otra parte, el desarrollo paralelo en comunicaciones y tecnología de redes han acelerado la conexión de estaciones de trabajo y equipos de cómputo en general, estos desarrollos están cambiando la vida de las personas. Por ejemplo, actualmente una persona puede obtener dinero desde un cajero automático usando alguna de las varias redes, independientemente de la localización física de la institución bancaria. Otro ejemplo, es cuando un agente de viajes tiene acceso a las principales bases de datos de aerolíneas, hoteles, etc., independientemente de su localización geográfica.

2.4.1 Sistemas Abiertos

Uno de los conceptos que actualmente tiene mucho apogeo es sin duda alguna el de *Sistemas Abiertos*. Este concepto es difícil de definir, pero de acuerdo a la IEEE :

> *" Un Sistema Abierto es aquel que soporta un conjunto completo de estándares tecnológicos internacionales de información, acoplando interoperabilidad y portabilidad de aplicaciones, datos y gente ".*

En este contexto, el término estándar se refiere a una especificación, un conjunto de funciones, o una implantación física que ha sido aceptada ampliamente formal o informalmente como un método preferido o interfaz. No necesariamente significa *el mejor* sino *el más deseable*.

El concepto de Sistema Abierto por otro lado, es una filosofía, una actitud, un punto de diseño y una característica de los sistemas. El objetivo es compartir información y tecnología fomentando su intercambio, eliminando redundancia y

minimizar la necesidad de que la gente aprenda diferentes caminos para hacer la misma cosa.

Muchas organizaciones han instalado una variedad de plataformas, y en cada una de ellas manejan información necesaria para la vida de la organización. El reto de los sistemas Abiertos, y con ellos los estándares, está en hacer que esas diferentes plataformas trabajen de maneja conjunta. Los estándares permiten mejorar la compatibilidad, dejan mayor libertad al usuario para elegir lo que más convenga a sus necesidades (tanto en hardware como en software), permiten explotar nuevas tecnologías y generan un ambiente de competencia tanto en precio como en desempeño dentro del mercado.

A pesar de que los estándares de programación y de comunicaciones han existido y evolucionado por décadas, UNIX de AT&T fue el primer sistema operativo que funcionó en diversas plataformas con éxito, permitiendo escalabilidad, portabilidad e interoperabilidad, como principales ventajas.

Sin embargo, existen varias versiones de UNIX, que no son totalmente compatibles entre sí. Es por ello que ahora, Sistemas Abiertos no es precisamente sinónimo de UNIX, sino de escalabilidad, portabilidad e interoperabilidad basada en estándares. Refiriéndonos específicamente a la familia de UNIX, este sistema operativo tiene como atributos significativos, los siguientes:

- *Una arquitectura abierta.* UNIX fue diseñado para ser expansible tanto en hardware como software.

- *Disponibilidad de código fuente.* Este aspecto es especialmente interesante en universidades y pequeñas compañías para poderlo modificar de acuerdo a sus intereses.

- *Portabilidad del Sistema Operativo.* El diseño e implantación del sistema es tal que el trabajo de adaptarlo a otra máquina es relativamente mínimo.

- *Portabilidad de aplicaciones.* El sistema y las herramientas asociadas con el ambiente de trabajo están escritas en lenguaje C. Este lenguaje está diseñado también para correr en múltiples arquitecturas. Por lo tanto, el software correspondiente a un sistema UNIX, puede ser recompilado en otro sistema. Esta portabilidad probablemente ha sido el factor más importante en el crecimiento y popularidad del sistema operativo UNIX.

2.4.2 Arquitectura de los Sistemas Distribuidos

Hay algunos conceptos básicos que forman parte del diseño de software en los sistemas distribuidos, los cuales deben tomarse en cuenta como aspectos principales de cualquier sistema de cómputo. En este contexto discutiremos los más importantes:

- *Transparencia.* La red debería ser *invisible* para los usuarios, programas de aplicación y administradores. ¿Cuáles son la ventajas y limitantes?

- *Integración.* ¿Cómo pueden ser relacionadas las diferentes máquinas actualmente en un ambiente distribuido?

- *Funcionamiento.* Últimamente, cualquier sistema de cómputo debe "ganarse" su lugar dentro de la red, con un funcionamiento que justifique su permanencia.

- *Balanceando carga.* Los sistemas distribuidos ofrecen el poder de esparcir la demanda de servicios entre la colección de máquinas disponibles en el ambiente de red corporativo.

Transparencia

La transparencia es un aspecto fundamental que forma parte del diseño, configuración, software y consideraciones operativas en un sistema distribuido. Por un lado, las computadoras en una red operan como sistemas individuales, con interacciones limitadas a hacer una conexión remota y transferencia de archivos por ejemplo. En esos casos, el usuario y administrador están explícitamente enterados de la existencia de la red y deben manejar sus errores directamente. Por otro lado, se contempla un sistema operativo de red, con el objetivo de permitir a usuarios, programadores y administradores tratar a una colección grande de computadoras, como si este conjunto fuera uno solo. Obviamente, este sistema está compuesto de múltiples componentes capaces de operar individualmente en forma eficiente.

Así, por ejemplo, en el caso de UNIX, un sistema operativo corriendo en algunas computadoras podría tener, jerárquicamente un sólo sistema de archivos identificados con una sola "raíz".

Integración

La forma, naturaleza de servicios disponibles y funcionamiento son profundamente afectados al tratar de asociarlos con otras máquinas que componen al sistema distribuido. El grado de integración es afectado por un número de consideraciones: madurez en la tecnología de software, el grado de similitud entre

los sistemas involucrados y el nivel de administración presente, así como aspectos tecnológicos tales como el ancho de banda de la red.

Hay numerosas formas mediante las cuales los servicios remotos se hacen disponibles a una computadora determinada. La más simple y comúnmente usada es manejar el concepto de conectar una terminal de una computadora a otra utilizando un software especializado. Este método, también conocido como emulación de terminal ha sido usado por casi una tercera parte de este siglo y se aplica principalmente a la transferencia de archivos. En este caso, el usuario y programas asociados actúan directamente con las máquinas involucradas.

Después, fue posible el acceso remoto a archivos manejándolos con las mismas operaciones que uno usaría para un archivo local por ejemplo: abrir, cerrar, leer y escribir. Así, usuarios y programas no tienen que emplear diferentes medios para acceder a datos remotos.

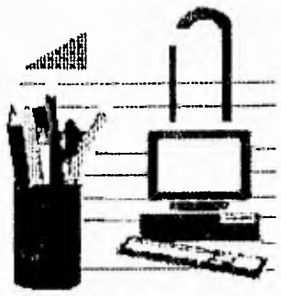
Posteriormente, continúa una etapa en la que se pueden ejecutar programas remotamente entre máquinas compartiendo así los primeros recursos entre las mismas.

Funcionamiento

Cualquier sistema de cómputo debe proveer un nivel de funcionamiento lo suficientemente alto para satisfacer las necesidades del usuario.

Balanceando la carga

Es importante aprovechar los recursos conjuntamente de las estaciones de trabajo que forman parte de nuestra red. Tal vez algunas puedan realizar una tarea específica y otras puedan desplegar los resultados, etc.



Capítulo 3

Recursos Disponibles

Es importante considerar en la realización de todo proyecto, los recursos con los que se cuenta para determinar los alcances y limitantes del mismo. En este capítulo, describiremos algunos de los recursos con los que cuenta la UNAM en materia de cómputo los cuales se utilizan para ofrecer el servicio que proponemos. Es importante señalar que podemos hacer uso de la gran infraestructura de RedUnam, compartiendo información entre las diversas bases de datos que se encuentran en las diferentes dependencias y no solo eso, sino que podemos manipular esa información y realizar aplicaciones combinando los variados servicios que ofrece la Red Internet, tal y como lo muestra el resultado del presente proyecto.

En este capítulo también describiremos los servicios más importantes con los que cuenta la Red Internet, ya que gracias a ellos, nos enteramos de lo que está sucediendo al otro lado del mundo, es impresionante saber que desde una simple PC uno puede ver la información más sobresaliente que acontece en diversas partes del planeta accediendo a bases de datos que pudieran encontrarse en algún otro lugar de la red, ya no es solo el correo electrónico el servicio que mantiene comunicado a las personas, sino que ahora contamos con otros con capacidades más poderosas y a su vez, día con día se difunden en la red convirtiéndose en estándares para el manejo y recuperación de la información.

3.1 Dirección de Cómputo para la Administración Académica

La Dirección de Cómputo para la Administración Académica (DCAA) proporciona servicios de cómputo y desarrollo de sistemas de información para la administración académica de las dependencias de la Universidad Nacional Autónoma de México, y de las instituciones externas con las que se establecen convenios; a la vez que se supervisa el cumplimiento de las normas y procedimientos relativos a los servicios que se prestan, a fin de lograr un uso racional de los recursos con que se cuenta.

La Dirección de Cómputo para la Administración Académica, siendo una dependencia fundamentalmente de servicio, la cual proporciona a través de tiempo de máquina, aplicaciones en equipos especiales, desarrollo de sistemas de información y asesorías, cuenta con un componente académico muy importante, ya que gran parte de sus actividades están dedicadas a la formación de recursos humanos y a la difusión de la cultura informática.

La DCAA realiza entre otras, las siguientes actividades:

- Operación y administración de Sistemas de Cómputo y periféricos.
- Operación y Administración de equipos especiales y auxiliares para la captura óptica y digital, microfilmación, corte, separación y firma de formas continuas.
- Elaboración de nuevas aplicaciones para lectura óptica y microfilmación.
- Transferencia y recepción de archivos para micros PC y equipos grandes (mainframes).

- Coordinación de los servicios de mantenimiento preventivo y correctivo a los equipos.
- Elaboración de sistemas de información para la administración académica.
- Coordinación de cursos para usuarios, becarios y prebecarios.
- Concertación de contratos de compraventa y mantenimiento de los diferentes equipos.
- Reuniones con los miembros del Comité de Usuarios de la Dirección.
- Reuniones con la Dirección General.
- Reuniones con proveedores de bienes y servicios informáticos.
- Atención a usuarios en los diferentes servicios.
- Control de partidas presupuestarias de mantenimiento y compras de equipo de Cómputo de las diferentes dependencias de la UNAM.

Actualmente, la DCAA cuenta con algunas coordinaciones y subdirecciones, entre ellas, destaca la Coordinación Técnica de Redes e Interoperabilidad, donde se desarrollan proyectos de innovación además de dar servicio de soporte técnico a los equipos con los que cuenta la DCAA, uno de los primeros proyectos fue precisamente el que estamos presentando como tema de tesis: "Kioscos de Información".

3.1.1 Proyectos desarrollados

Dentro de la Dirección de Cómputo para la Administración Académica (DCAA) se han desarrollado proyectos muy importantes, entre los cuales destacan los sistemas de inscripciones para diversas dependencias de la UNAM así como sistemas de información, utilizando una plataforma de hardware y software acorde a las necesidades del usuario para manejar las cantidades de información que se

requiere. Específicamente hablando, en el área técnica no sólo tenemos el soporte a equipos y periféricos de esta dependencia, sino que el desarrollo de proyectos en los últimos meses ha alcanzado una etapa en la que la innovación, mantenimiento y difusión de los mismos es muy importante.

Tomando en cuenta los recursos en materia de cómputo con los que contamos no sólo dentro de la Universidad sino de todo el mundo (gracias a la Red Internet de la cual hemos venido hablando), tenemos en nuestras manos herramientas con las cuales podemos comenzar a diseñar sistemas para solucionar diversos problemas; tal es el caso del Sistema de Información Académica (SINAC) el cual propone el almacenamiento masivo de la información académica de los alumnos de la UNAM en CD-ROM's y su consulta vía RedUnam. Este proyecto es el antecedente para la realización del que presentamos en esta ocasión, SINAC incorpora variadas herramientas del dominio público disponibles para el manejo de la información tales como: Mosaic, gopher y Wais.

A partir de entonces, las perspectivas de desarrollo del proyecto descrito en el párrafo anterior se incrementaron, ahora con los *kioscos de Información* y otros sistemas involucrando los mismos conceptos pero aplicados a situaciones y problemas diferentes. Además, cada vez se encuentran nuevas herramientas dentro de la red y nuestro trabajo es tratar de incluirlas con las ya existentes en la solución de problemas reales.

3.1.2 Equipo disponible

La DCAA se caracteriza por contar con el equipo y software adecuados para ofrecer los servicios de cómputo a diversas dependencias de la UNAM, los cuales incluyen: mainframes, estaciones de trabajo, computadoras personales, manejadores de bases de datos y otros.

Los siguientes, son los principales equipos de cómputo con los que cuenta la Dirección de Cómputo para la Administración Académica.

Equipo A-12 de Unisys

Este sistema pertenece a la serie A de Unisys (que reemplazó a la serie B5000, B6000 y B7000 de Burroughs antes de que esta compañía se fusionara con Univac) y cuenta con una amplia gama de terminales de propósito general, impresoras, procesadores de comunicaciones y dispositivos de almacenamiento de datos. Entre las características más sobresalientes de la serie A cabe mencionar:

- Tienen una alta eficiencia en la ejecución de programas escritos en lenguajes de alto nivel, comparable a la de ensambladores y lenguaje máquina de otros equipos.
- Permiten un fácil incremento de productividad mediante el uso de herramientas de tercera y cuarta generación sin afectar el rendimiento.
- Requieren de un mínimo esfuerzo para la implantación de sofisticadas redes de comunicación.
- Su hardware es totalmente modular, lo que proporciona una gran facilidad de expansión.
- El hardware y el software están acoplados para ajustar la operación del sistema en la forma más eficiente.

El principal uso de este equipo es el almacenar y procesar la mayor parte de la información académica dentro de la UNAM e instituciones incorporadas; se

generan historias académicas, comprobantes de inscripción, actas de examen ordinario y extraordinario, etc.

Lectura Óptica

La introducción de las computadoras electrónicas en las organizaciones, ha significado la posibilidad de desarrollar sistemas de información automatizados, que apoyan las actividades de planeación, administración y control de una institución en forma más ágil.

La UNAM a través de la DCAA, tiene en sus instalaciones dos equipos de lectura óptica de marcas. El sistema captura la información por medio de marcas plasmadas en el documento fuente (hoja óptica), su velocidad de lectura es de 85 hojas por minuto y tiene una capacidad máxima de 7,076 caracteres por hoja.

Unidad de Microfilmación

Otro de los recursos con los que cuenta la DCAA es la *microfilmación* como un medio de almacenamiento y consulta de grandes volúmenes de información. Es muy útil en el caso de las historias académicas, la capacidad de almacenamiento de esta unidad es 208 hojas tamaño carta de información en una microficha a una velocidad de 70 microfichas por hora.

Equipo SparcStation SUN

Este tipo de equipos han adquirido una enorme importancia en los últimos 2 años dentro de la DCAA, debido a la necesidad de tener un *sistema abierto* que

permita en un momento dado la descentralización de ciertos procesos administrativos.

El equipo SUN está basado en la arquitectura RISC (Reduced Instruction Set Computer). La filosofía de esta arquitectura es buscar el incremento de la velocidad de procesamiento a partir de una simplificación en el diseño del CPU (Unidad Central de Proceso).

En 1987, SUN Microsystems lanzó su primera computadora basada en una nueva arquitectura de CPU denominada SPARC (Scalable Processor Arquitecture). A diferencia de los microprocesadores existentes, este nuevo CPU tan solo constaba de 20,000 compuertas. Además, se presentó con el esquema de arquitectura abierta.

Desde entonces, SPARC llegó a ser una familia de estándares, un conjunto de circuitos integrados de al menos una docena de vendedores, un vehículo para la estandarización y una base en la cual docenas de fabricantes están implantando sus nuevas estaciones de trabajo.

La arquitectura SUN SPARC no está relacionada específicamente a alguna de hardware en especial, se han concedido licencias a diferentes fabricantes, por lo que podemos encontrar en el mercado diferentes microprocesadores, todos ellos bajo el esquema SPARC. El concepto de escalamiento puede ser interpretado como el número de CPU's que pueden ser utilizados en varios productos, todos ellos con la arquitectura SPARC[4] .

El sistema Operativo UNIX

En el punto anterior mencionamos algunas características del hardware a utilizar en el proyecto. Sin embargo, no podemos dejar de mencionar otra característica indispensable: el sistema operativo UNIX.

[4] Las características a detalle de este tipo de equipo son discutidas en el capítulo 5.

El origen de UNIX se remota a mediados de 1960 cuando las compañías American Telephone & Telegraph (AT&T), Honeywell, General Electric y el Instituto Tecnológico de Massachusetts (MIT) trabajaron en un sistema de información denominado MULTICS (Multiplexed Information and Computing Service). El objetivo era tener un servicio de cómputo las 24 horas del día y los 365 días del año, diseñado de manera modular para que fallas en alguna parte no afectaran al resto del sistema.

Sin embargo, en 1969 el proyecto fue abandonado debido a que no se terminó en el tiempo establecido dada su complejidad. A pesar de ello, en el mismo año uno de los desarrolladores de Multics decidió rescatar parte del proyecto y realizar un diseño más sencillo en una PDP-7. Se le unieron en el desarrollo Ken Thompson y Brian Kernighan, este último fue quien sugirió el nombre de UNIX, un juego de palabras entre Multics y el objetivo del nuevo proyecto; mientras Multics trató de hacer muchas cosas, Unix sólo iba a correr programas.

La primera versión en la PDP-7 fue hecha en lenguaje ensamblador en 1969. Para 1971, Ken Thompson reescribió el código fuente en lenguaje C. Esta nueva característica hizo al sistema *portable*, es decir, un sistema operativo capaz de funcionar casi en cualquier plataforma con sólo realizar modificaciones en funciones primitivas que están íntimamente ligadas con el hardware de la máquina.

Muy pronto se hizo popular en varias compañías y universidades siendo una de ellas, la Universidad de Berkeley, que hizo una serie de aportaciones importantes al sistema.

Hoy en día, una enorme cantidad de computadoras en todo el mundo utilizan a UNIX como sistema operativo. Las diferentes versiones corren en casi todo tipo de computadoras existentes, desde una PC hasta en una supercomputadora.

Entre las características del sistema operativo UNIX, están las siguientes:

- Es un sistema operativo multiusuario, con capacidad de simular multiprocesamiento.
- Está escrito en lenguaje C.
- Dispone de un lenguaje de control programable llamado Shell.
- Ofrece facilidades para la creación de programas y sistemas, y un ambiente muy propio para las tareas de diseño de software.
- Emplea manejo dinámico de memoria.
- Tiene capacidad para interconectar o comunicar procesos.
- Emplea un sistema jerárquico de archivos, con facilidades de protección de los mismos.
- Emplea un manejo consistente de archivos de diversos tipos.
- Tiene facilidad para redireccionamiento de entradas/salidas.
- Garantiza un alto grado de portabilidad.

El sistema se basa en un núcleo conocido como *kernel* el cual reside permanentemente en memoria y atiende todas las llamadas del sistema, administra el acceso a los archivos y el inicio de tareas de los usuarios, tiene las siguientes funciones:

- Creación de procesos, asignación de tiempos de atención y sincronización.
- Asignación de la atención del procesador a los procesos que lo requieran.
- Administración de espacio en el sistema de archivos, que incluye: acceso, protección y administración de usuarios.
- Comunicación entre procesos y entre usuarios.

- Manipulación de Entrada/Salida y administración de periféricos.
- Supervisión de la transmisión de datos entre la memoria principal y los periféricos.

Después del kernel, se encuentra el *shell* que sirve como interfaz entre el usuario y el *kernel*, entre sus funciones se encuentran:

- Presenta el prompt del sistema operativo al usuario.
- Interpreta comandos.
- Envía al *kernel* los comandos que el usuario especifica.
- Devuelve al usuario los resultados de los comandos ejecutados.
- Sirve como lenguaje de programación.

Por último, tenemos las aplicaciones que se componen básicamente de: procesadores de texto, comandos, programas, manejadores de bases de datos, etc.

Sistema de impresión Xerox Print Manager (XPM)

La DCAA cuenta también, con un sistema de impresión electrónica de alta velocidad. Una impresora láser marca Xerox, modelo 4850 conectada a la RedUnam, permite que estaciones de trabajo puedan imprimir vía red. En este caso, tenemos a los equipos SparcStation que actúan como clientes, una impresora láser con una velocidad de impresión de 50 páginas por minuto a 2 colores (negro y azul) por ambos lados, interpretando diferentes formatos de archivos: PostScript, PCL, Interpress, texto, y otros.

Y por último, un servidor de impresión (recibe los trabajos y se encarga de enviarlos a la impresora) con un software llamado *Xerox Print Manager* instalado en

una estación de trabajo SparcStation 5 y sistema operativo Solaris 1.1.1 Revisión B, este sistema proporciona la habilidad de recibir archivos desde los clientes vía TCP/IP (o comandos de impresión de UNIX: lpr, lp) y la salida a una impresora Xerox 4850.

Entre los beneficios que se tienen al tener este sistema, se encuentran:

- Permitir la impresión de alta velocidad en ambiente de red.
- Restringe la impresión y otras actividades a usuarios autorizados.
- Permite definir grupos de usuarios y privilegios asociados.
- Requiere mínima intervención del operador.
- Sostenta una robusta manipulación de colas de impresión.
- Proporciona soluciones de conectividad entre LAN y una amplia variedad de impresoras láser Xerox dúplex con capacidad desde 35 hasta 135 páginas por minuto.
- Ofrece una interfaz de usuario gráfica muy amigable a través de ventanas.

XPM ofrece además, una capacidad de formateo de reportes. Al tiempo de envío del trabajo, el usuario puede llamar un archivo de formato residente en XPM que se encargará de asociarlos para que el listado tenga el formato deseado. Esto es muy útil por ejemplo, en la generación de historias académicas o constancias, donde se tiene un mismo formato para la hoja y lo único que resta es llenarla con la información que proviene de una aplicación.

3.2 Tecnologías de Redes Locales

En menos de 20 años las redes locales dejaron de ser experimentos de laboratorio para convertirse en productos comerciales de enorme utilización en las empresas.

La topología de una red local se refiere a la forma en la que se conectan físicamente las computadoras (u otros dispositivos) a la red. En la Figura 3.1 se muestran las tres topologías básicas de las redes locales: estrella, anillo y bus. Los medios de transmisión empleados con mayor frecuencia son el par trenzado, el cable coaxial y la fibra óptica.

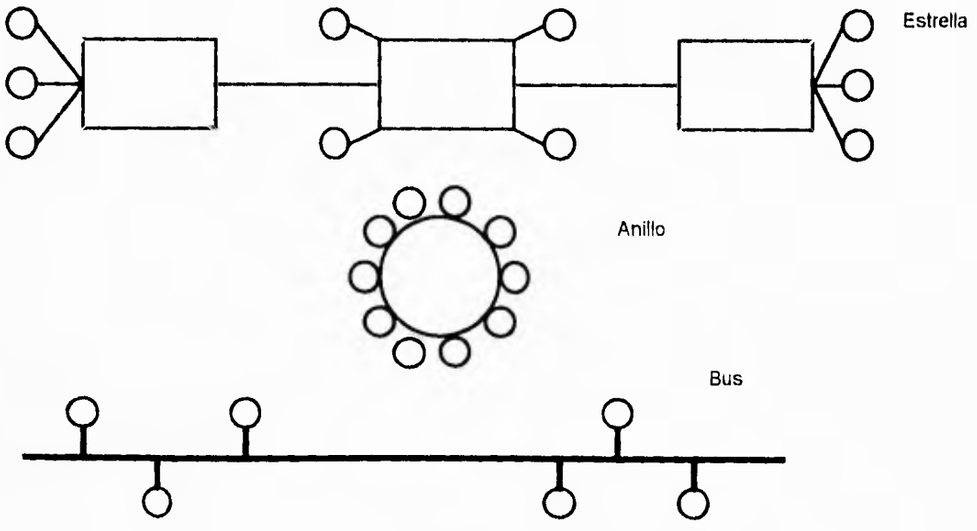


Figura 3.1 Topologías básicas de las redes locales

En una topología en estrella cada computadora se encuentra conectada por una línea punto a punto a un nodo central, esta topología presenta la desventaja de depender completamente de un nodo en particular.

Las topologías en anillo y bus no tienen este inconveniente. En la topología en anillo las computadoras se conectan entre sí por líneas punto a punto formando un circuito, mientras que en una topología en bus todas las computadoras se conectan a una línea multipunto. Las topologías en anillo y bus son las más empleadas en la actualidad.

3.2.1 Redes tipo Ethernet

Una red local está formada por un conjunto de computadoras y otros dispositivos interconectados dentro de un área geográfica limitada, con el fin de intercambiar recursos e información. La mayoría de las redes locales se caracteriza por:

- Radio de acción pequeño, hasta de unos cinco kilómetros.
- Velocidad de transmisión en el orden de millones de bits por segundo.
- Ambiente relativamente libre de errores de transmisión.
- Medio de comunicación compartido por todos los dispositivos conectados a la red.
- Flexibilidad en la topología, es decir, facilidad en la modificación y reconfiguración de la distribución física de los dispositivos conectados a la red.

Ethernet, el tipo de red local más difundido en la actualidad, cumple con todas las características anteriores. Su primera implantación fue desarrollada en Xerox por Robert M. Metcalfe a principios de los años 70's, para conectar hasta

100 estaciones de trabajo en un área de 1 km. transfiriendo información a 2.94 Mbps. Es concebida y recomendada en ambiente de oficina y para áreas de industria ligera donde no se requieran tiempos de respuesta determinísticos. Ethernet toma su nombre como recuerdo de aquella teoría del siglo XIX según la cual el universo estaba suspendido en una especie de éter por el que las ondas electromagnéticas podían propagarse.

En 1978 se publica la primera norma como un trabajo conjunto de las empresas Xerox, Intel y DEC. Esta es la base del estándar ANSI/IEEE 802.3 publicado en 1983 por el IEEE. La norma internacional ISO 8802/3 de la Organización Internacional de Estándares (ISO) también está basada en la especificación de 1978. Nos referiremos más adelante al protocolo de acceso CSMA/CD, implantada en un bus coaxial con transmisión serie en banda base a 10 Mbps utilizando codificación Manchester.

En Ethernet, el canal de comunicación común es un cable coaxial y un bus con impedancias de terminación en los extremos, al que se conectan todos los dispositivos que forman la red como se muestra a continuación en la figura:

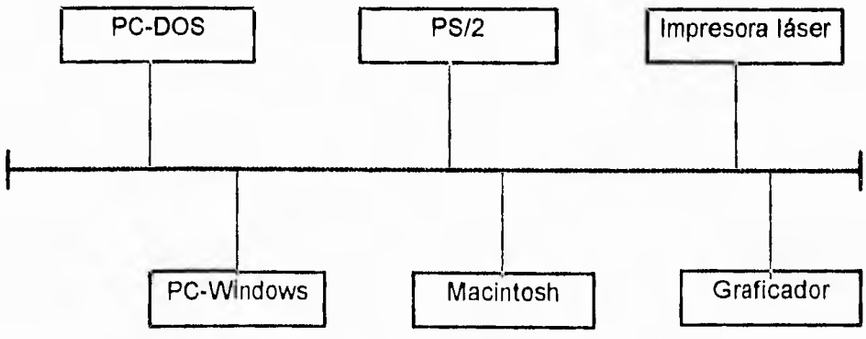


Figura 3.2 Red local con topología de bus

Cada sitio en la red local tiene un identificador único: su dirección. Cuando una computadora desea mandar información a otro dispositivo, simplemente forma un paquete con el mensaje, la dirección del destinatario, su propia dirección y otra información. En la nomenclatura de redes locales, estos paquetes se llaman tramas. Una vez formada una trama, ésta se envía en serie, es decir, bit por bit a través del cable coaxial. Las señales en el bus son omnidireccionales, esto es, se difunden en los dos sentidos del cable, de tal manera que todos los sitios conectados a la red detectan la información. Aquél dispositivo que reconozca en la dirección destino su propia dirección, sabe que la trama contiene información dirigida hacia él y por lo tanto la leerá del bus. Los demás sitios ignoran esta trama.

El protocolo de acceso CSMA/CD

Como en casi todas las redes locales, se tiene un canal común a todas las estaciones a través del cual se envía información. Todos los nodos que se conectan a la red deben de obedecer una serie de reglas y convenios para establecer cuándo y cómo se puede acceder el canal común. A estas reglas se les conoce como protocolos de acceso al medio, y para Ethernet el protocolo utilizado es *Acceso Múltiple con Sensado de Portadora y detección de Colisiones*, o *CSMA/CD* (Carrier Sense Multiple Access/Collision Detection).

Una estación que desea transmitir una trama escucha primero si alguien está trasmitiendo (está sensando la presencia de portadora). Si encuentra libre el medio, envía la trama. En caso de encontrarla ocupada, espera a que esta se libere y transmite su trama inmediatamente.

Con el comportamiento anterior, conocido como 1-persistente, es posible imaginar el siguiente escenario:

- En el tiempo tl la computadora A inicia una transmisión.
- Poco tiempo después, en t2, la computadora B desea enviar una trama pero encuentra el bus ocupado y espera.
- Más adelante, en t3, otra computadora C también desea enviar una trama y encuentra el bus ocupado por A, por lo que también espera.
- En t4, A termina de transmitir y el cable coaxial queda libre. Esto lo notan las computadoras B y C que inician la transmisión de sus respectivas tramas casi simultáneamente.

Como las señales se difunden a lo largo de todo el bus, en el escenario antes descrito, las tramas de las computadoras B y C van a interferir entre sí. Se dice entonces que ha ocurrido una colisión. En CSMA/CD, la estación que transmite también "escucha" la señal en el cable. Si lo que recibe es lo que está enviando todo va bien. En caso contrario, supone que ha ocurrido una colisión.

Al detectar la colisión, la estación transmisora aborta su trama y en su lugar envía una pequeña señal de cuatro a seis bytes reforzando la colisión (jamming signal) para garantizar que las demás estaciones involucradas en la interferencia también puedan detectar la colisión. Después de enviar esta señal, la estación espera durante un intervalo de tiempo "aleatorio" y vuelve a intentar la transmisión de la trama escuchando en el bus para ver si éste se encuentra libre.

El tiempo de espera que debe guardar una estación, antes de intentar retransmitir su trama, se calcula mediante un algoritmo llamado: "*espera exponencial binaria truncada*".

k=mínimo(Número de intentos,10)
r=Valor aleatorio entre 0 y 2^k-1
retraso=r*ranura de tiempo de colisión.

Con este protocolo de acceso, las redes Ethernet ofrecen un tiempo de respuesta inmediato cuando el tránsito en el bus es muy poco. Conforme se va incrementando el tránsito, las probabilidades de colisión aumentan así como los tiempos de espera, precisamente para tratar de disminuir la posibilidad de colisiones múltiples. En un ambiente de mucho flujo de información, los tiempos de respuesta serán considerables y sobre todo, no predecibles.

3.2.2 Token Ring

Una red local con topología de anillo está integrada por un conjunto de estaciones conectadas en serie, por medio de enlaces (unidireccionales) punto a punto (de par trenzado, cable coaxial o fibra óptica), de manera que se forma una trayectoria cerrada o anillo que permite la comunicación entre éstas.

Cuando una estación recibe datos por un enlace, los retransmite bit a bit por el otro enlace a la misma velocidad que los recibió. Debe observarse que cada estación introducirá un retardo durante el proceso de regeneración y repetición de los bits.

Las estaciones que integran el anillo envían los datos en forma de tramas que también contienen información de control y las direcciones de las estaciones de origen y destino.

De la misma forma que en otros tipos de redes locales (donde el medio físico se comparte entre otras estaciones), una estación que esté preparada para transmitir una o varias tramas deberá esperar su turno de acuerdo al protocolo de control de acceso al medio. Cuando una estación tenga el permiso para transmitir, insertará las tramas al anillo (una a la vez). Cada trama viajará por el anillo (en una sola dirección), pasando por todas las estaciones.

Aunque en la actualidad se conocen diversas técnicas para controlar el acceso al medio en una red con topología de anillo, la técnica más común se conoce como "Token Ring". Esta técnica basada en la circulación de un patrón de bits único (*token*) que otorga el permiso de transmisión, fue adoptada por el comité IEEE 802 encargado de los estándares para redes de área local y metropolitana. En 1989, el IEEE emitió el estándar IEEE 802.5 que describe el método de acceso y las especificaciones de la capa física para una red del tipo Token Ring. Este estándar es compatible con las redes Token Ring de IBM.

El estándar IEEE 802.5 describe el funcionamiento de una red Token Ring con las siguientes características:

- Topología: anillo
- Medio: Par trenzado. El uso de otros medios está sujeto a futuras consideraciones.
- Señalización: Manchester diferencial.
- Velocidad: 4 y 16 Mbps.
- Protocolo de control de acceso al medio: paso de *token.*

Protocolo de control de acceso al medio

En el caso de las redes Token Ring, el control de acceso al medio está distribuido entre todas las estaciones del anillo. Es decir, es un esquema que no depende de ninguna estación y por lo tanto es bastante robusto.

Mientras ninguna estación esté transmitiendo, estará circulando por el anillo un *token* o canastilla y todas las estaciones estarán en estado de repetición (retransmiten por un enlace todos los bits recibidos por el otro). La recepción del *token* ofrece la oportunidad de ofrecer la oportunidad de transmitir una trama o una secuencia de tramas.

Cuando una estación tenga uno o varios paquetes listos para transmitir, les deberá agregar información de control y las direcciones de origen y destino y deberá esperar la llegada de un *token*. En el instante en que la estación detecte que se están repitiendo los bits de un *token*, se cambiará el valor de uno de uno de ellos (bit T) convirtiéndolo en el campo AC de una trama de información. La estación pasará del estado de repetición al estado de transmisión. Inmediatamente después se insertarán al anillo los demás campos de la trama. Durante la transmisión, se determina una secuencia de verificación de trama que se agrega después del campo de información.

En este momento, las otras estaciones continúan en estado de repetición y ninguna de ellas podrá transmitir datos por el anillo dado que no hay un *token* circulando. Para evitar que una estación transmita indefinidamente, se define un tiempo máximo de posesión del *token*, después del cual deberá emitirse un nuevo *token* que proporcione la oportunidad de transmitir tramas de las otras estaciones.

Cada trama transmitida se repetirá de una estación a otra y durante la repetición se analizará su campo de control (FC). En caso de tratarse de una trama de tipo MAC, perteneciente al protocolo de acceso, la información será procesada por todas las estaciones del anillo. Si la trama es de tipo LLC, lo cual indica que transporta información entre usuarios, la estación que reconozca su dirección dentro del campo de dirección destino, hará una copia de la trama. En cualquiera de los casos se continúa en el estado de repetición.

Dado que las estaciones forman una trayectoria cerrada, se observa que debe contarse con un mecanismo para evitar que las tramas circulen por el anillo indefinidamente y garantizar que las otras estaciones puedan transmitir. Para ello, la estación que transmitió la trama debe realizar dos funciones: retirar la trama del anillo y emitir un nuevo *token* después de terminar la transmisión.

Cuando una estación termina la transmisión de una o varias tramas, verifica si su dirección ha aparecido en el campo de dirección origen de los bits que está

recibiendo (se verifica si la trama transmitida ha dado una vuelta al anillo). Mientras no aparezca esta dirección, la estación continuará en estado de transmisión enviando al anillo bits de relleno. Cuando se cumple la condición, la estación transmite un nuevo *token*.

Después de esto, la estación continúa en estado de transmisión (enviando bits de relleno) hasta que termine de recibir la trama que transmitió. Lo anterior se hace con el fin de repetir los bits de la trama o tramas enviadas y de esta forma retirarlas del anillo. Una vez retiradas las tramas transmitidas, la estación pasa al estado de repetición. Debe observarse que una estación en el estado de transmisión no repite los bits recibidos.

En el estándar IEEE 802.5 se define un mecanismo de prioridades con 8 niveles, donde los niveles se asignan dinámicamente dependiendo de la clase de servicio requerido por el usuario. El esquema funciona de tal forma que las estaciones que tengan tramas con la misma prioridad tengan la misma oportunidad de acceso al anillo.

El esquema de control de acceso al medio con prioridades es una variante del esquema explicado anteriormente. Para poder manejarlo, el *token* cuenta con un campo que indica la prioridad del *token* y otro campo que permite reservar el siguiente *token*.

Ahora debe considerarse que para poder transmitir una trama, no basta con recibir un *token*. También debe cumplirse la condición de que el *token* tenga una prioridad menor o igual a la trama que quiera transmitirse.

Cuando una estación esté repitiendo el campo AC de una trama, podrá solicitar que el siguiente *token* sea enviado a una determinada prioridad cambiando el valor de los bits de reservación (RRR), siempre y cuando la prioridad de la trama que desea transmitir sea mayor que la prioridad que aparece en esos bits. Conforme el campo AC de una trama va pasando por todas las estaciones, aquellas que tengan

una trama lista para transmitir intentarán hacer una reservación en este campo. Cuando regrese la trama a la estación transmisora, se tendrá en el campo RRR la máxima prioridad reservada. El nuevo *token* se transmitirá con una prioridad igual al máximo valor entre:

- La prioridad actual del *token*.
- La máxima prioridad reservada.
- La prioridad de las tramas listas para transmitir por esta estación (si es que las hay).

Con esto se garantiza que el próximo *token* será usado por la estación que tenga las tramas de mayor prioridad.

Si el nivel de prioridad asignado al nuevo *token* es mayor que el nivel que tenía anteriormente, la estación que emitió el *token* almacenará en una pila los valores de las prioridades anterior y actual. De esta forma, cuando ya no se tengan más tramas para transmitir con la prioridad actual o mayor, la estación que haya incrementado la prioridad del *token* a este nivel será la encargada de regresar la prioridad del *token* al nivel que tenía anteriormente. Se puede observar que todas las estaciones del anillo son responsables del esquema de prioridades.

3.2.3 FDDI

FDDI es una interfaz de datos distribuida sobre fibra óptica (por sus siglas en inglés Fiber Distributed Data Interface). Es una fibra óptica de alto rendimiento Token Ring LAN corriendo a 100 Mbps cubriendo distancias mayores a los 100 kilómetros de radio con hasta 1000 estaciones conectadas. Puede ser utilizada en la misma forma que las LAN's del tipo 802, pero con un ancho de banda más alto,

generalmente se utiliza como red troncal para la interconexión de LAN del tipo 802.3 y 802.5 .

FDDI utiliza un anillo doble -primario y secundario- de fibra óptica, este último como respaldo en caso de fallo del anillo primario. Los datos circulan en sentidos opuestos en cada anillo para facilitar la reconfiguración en caso de rotura de cualquiera de ellos, siendo posible construir redes FDDI con un único anillo.

FDDI utiliza fibras multimodo porque el costo adicional de las fibras de modo sencillo no se necesitan para redes corriendo a solo 100 Mbps. También utiliza LED's en lugar de lásers, no sólo debido a su bajo costo sino también porque FDDI puede algunas veces ser utilizado para conectarse directamente a las estaciones de trabajo de los usuarios. Existe el peligro de que por curiosidad algún usuario pudiera ocasionalmente desconectar el conector de la fibra y mirar directamente al flujo de bits viajando a 100 Mbps. Con el rayo láser activo el usuario podría terminar con su retina, en cambio los LED's son muy inofensivos para causar daño a la vista y capaces de soportar la transferencia de datos a esa velocidad.

La especificación del diseño de la FDDI no exige más de un error en 2.5 x 10^{10} bits. Muchos desarrollos diferentes logran un funcionamiento mejor.

El cableado de FDDI está constituido por dos anillos de fibras, uno transmitiendo en el sentido de las manecillas del reloj, y el otro en sentido contrario, como se muestra en la figura 3.4a. Si alguno de los dos se llega a desactivar, el otro puede emplearse como respaldo; si los dos se desactivaran en el mismo punto, por ejemplo, como consecuencia de un incendio o alguna otra causa en el conducto del cable, los dos anillos podrán unirse para formar un solo anillo que tendrá una longitud de casi el doble, como se muestra en la figura 3.4b. En cada estación hay redes que pueden emplearse para unir a los dos anillos o *puentear* una estación, en el momento que ocurre un problema en cualquiera de ellos. También pueden utilizarse centrales de cables, como se vio en el protocolo 802.5.

Mediante la FDDI se definen dos clases de estaciones, la A y B. Las estaciones tipo A se conectan a los dos anillos; en tanto que las estaciones clase B, que son más económicas, sólo se conectan a uno de los anillos. Dependiendo de la importancia que pueda tener la tolerancia a fallos, una instalación puede optar por seleccionar la estación de clase A o B, o una mezcla de ellas.

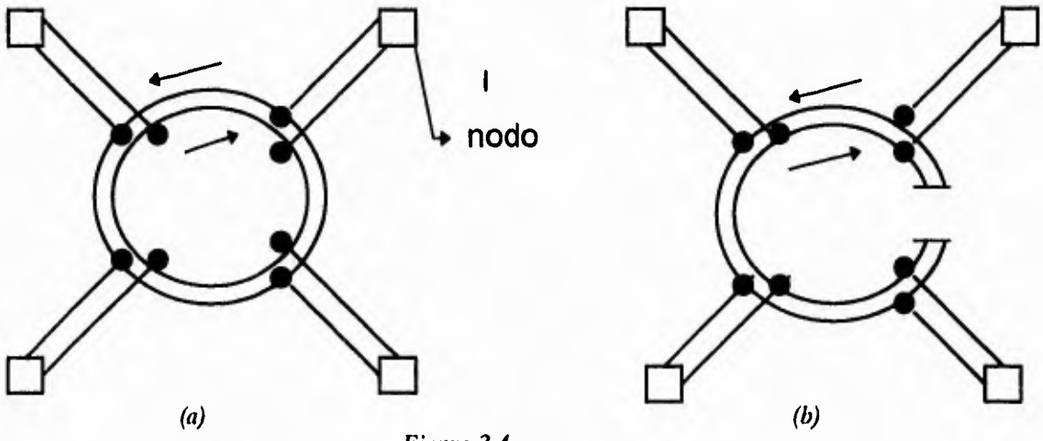


Figura 3.4

3.3 RedUnam

Tal vez uno de los ejemplos más complejos de la forma en que se puede llegar a enlazar los equipos de cómputo utilizados en las universidades mexicanas lo brinda REDUNAM. Red de la Universidad Nacional Autónoma de México.

En octubre de 1985, la UNAM e IBM de México suscribieron un convenio con el cual se puso en marcha un proyecto conjunto de investigación y desarrollo en el que se contemplaba: La instalación de una red universitaria de cómputo de apoyo a la docencia que permitiera el acceso remoto a los sistemas de procesamiento de

datos actuales y futuros en las dependencias de la UNAM y la creación de un laboratorio para el diseño y la manufactura apoyado por computadoras. Dicho convenio fue firmado por el rector de la UNAM, el presidente y director general de IBM y por el secretario de Comercio y Fomento Industrial. A raíz del convenio se integraron dos grupos de especialistas de la UNAM e IBM con objeto de definir detalladamente el plan de trabajo y así cumplir con los objetivos propuestos.

En el caso específico de la red, el planteamiento tecnológico original sufrió modificaciones a medida que se analizaron problemas similares en otras universidades y al conocimiento que se fue adquiriendo sobre comunicaciones, así como el avance tecnológico en el campo de las redes de computadoras.

El esfuerzo de cuatro años concluyó finalmente en septiembre de 1989 con la inauguración oficial de REDUNAM por parte del rector José Sarukhán Kermez.

Una red homogénea de computadoras es aquella compuesta por máquinas del mismo tipo y con características similares. REDUNAM, por el contrario es una red heterogénea de cómputo integrada por equipos de distintos proveedores y con características variadas. La diversidad de la red dificulta la labor de enlazar los sistemas, pero permite utilizar de manera transparente todos los recursos de cómputo ya existentes. Para lograr la comunicación del equipo se utilizan varios métodos de transmisión como Token Ring, Ethernet y x.25.

El corazón de la red es un sistema medular (backbone) de fibra óptica que transmite la información a una velocidad de 16 Mbs en un anillo (token-ring) que enlaza cuatro subanillos. El backbone tiene una longitud de 10 kms y se comunica con los subanillos mediante puentes (bridges) que sirven de enlace entre el anillo principal y los subanillos satélite. La información fluye a una velocidad de 4 Mbs, combinando señales ópticas de edificio y señales eléctricas dentro de ellos.

Como se mencionó anteriormente, la comunicación con los anillos satélite se realiza mediante puentes lógicos, los cuales son físicamente microcomputadoras con

dos tarjetas de expansión para token-ring y en donde se ejecuta el programa IBM Token-ring Network Bridge.

La red token-ring cubre la zona de centros e institutos de investigación de la comunidad científica y la zona de las ingenierías. Dentro de los subanillos, los diferentes edificios están conectados por fibra óptica a una velocidad de 4 Mbs. Se utilizó una fibra óptica para exteriores, para poder cablear conjuntamente la token-ring y la Ethernet.

Parecería que para el volumen de datos de una Universidad, el utilizar fibra óptica es un dispendio innecesario, mas no es así debido primordialmente a dos factores:

- La propiedad que posee la fibra óptica de no verse afectada por descargas eléctricas, lo que constituye en caso de la Ciudad Universitaria, un problema crítico, dadas las particularidades físicas del terreno. Este se encuentra en una zona volcánica que en algunos lugares supera los 20 metros de profundidad, lo que dificulta el logro de una conexión eléctrica adecuada, y

- La velocidad de transmisión que se requiere es alta, ya que se estimaba que para 1990 habría 1500 estaciones de trabajo conectadas a los diferentes subanillos.

En Figura 3.5 se aprecia el primer anillo satélite de la red está formado por la Dirección de Cómputo Académico, la Facultad de Ciencias, el Instituto de Investigaciones Nucleares y la División de Estudios de Posgrado de la Facultad de Contaduría y Administración. El puente de este anillo hacia el backbone se encuentra en la Dirección General de Servicios de Cómputo Académico.

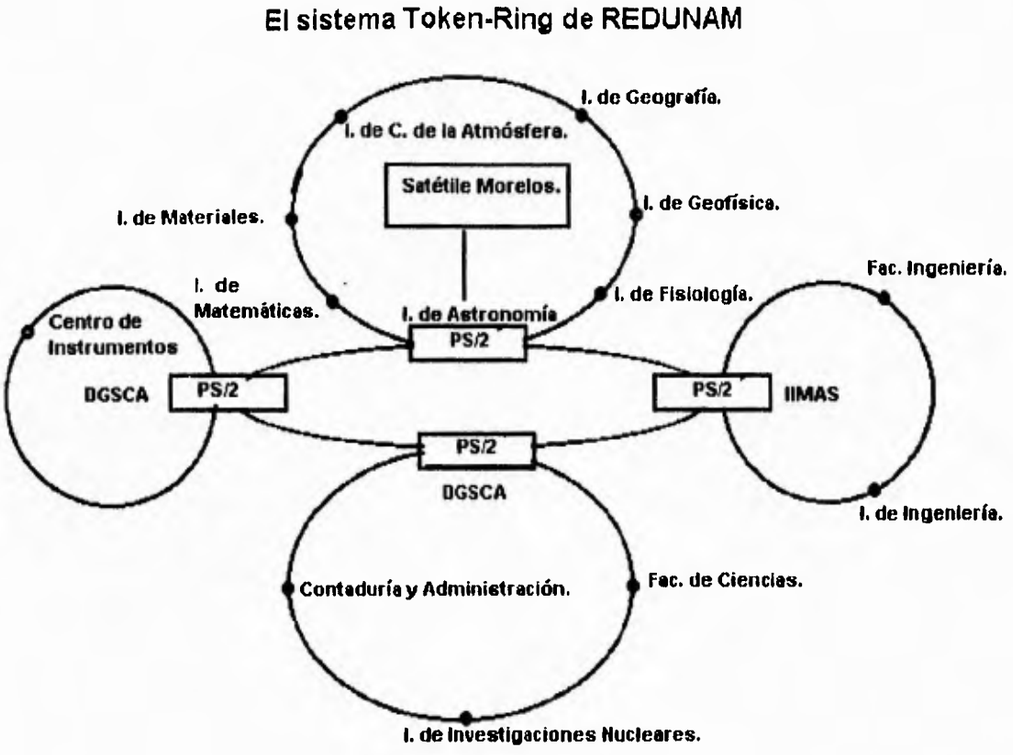


Figura 3.5 Arquitectura de RedUnam

El segundo anillo satélite está integrado por los institutos de Astronomía, de Matemáticas, de Materiales, de Ciencias de la Atmósfera, de Geografía, de Geofísica y de Fisiología Celular. La compuerta de este anillo se encuentra en el primero de estos.

El tercer anillo satélite está integrado por el Instituto de Investigaciones en Matemáticas Aplicadas y Sistemas (IIMAS); la División de Estudios de Posgrado de la Facultad de Ingeniería, el Centro de Cálculo de la Facultad de Ingeniería (CECAFI) y el Instituto de Ingeniería. El puente de este anillo se encuentra en el IIMAS. Además del anillo y subanillos con base en el protocolo token-ring, se decidió instalar en la UNAM una red que utiliza el protocolo conocido como

Ethernet y cuya principal función es enlazar los sistemas existentes del tipo digital, así como otras computadoras.

En RedUnam también se utiliza el protocolo x.25, el cual permite conectar a la red a los equipos Hewlett Packard. Usuarios externos al campus de Ciudad Universitaria tienen acceso a la red a través de Telepac (servicios de transmisión de datos públicos de la Secretaría de Comunicaciones y Transportes).

La complejidad de RedUnam es bastante grande. El número de usuarios aumenta día con día. En el caso de los usuarios de BITNET en la UNAM, éstos suman más de 700 personas.

3.4 La Red Internet

Esta es una red de área amplia (mundial), de conmutación de paquetes, que emplea el protocolo de comunicaciones TCP/IP (Transmision Control Protocol/Internet Protocol).

A principios de los 70's, el Departamento de Defensa de los Estados Unidos (DoD) encargó a la Agencia de Proyectos e Investigación Avanzada (ARPA) el desarrollo de un conjunto de protocolos de comunicación, denominado TCP/IP. Inicialmente se conectaron 3 universidades y un centro de investigaciones (Universidad de California en Los Angeles y Santa Barbara, la Universidad de Utah y el Instituto de Investigaciones de Stanford) empleando dicho protocolo para conformar la Red ARPAnet.

ARPAnet era una red experimental que apoyaba a la investigación militar, en particular la investigación sobre cómo construir redes que pudieran soportar fallas parciales (como las producidas por bombardeos) y aún así funcionar (piense en esto cuando se describa como trabaja la red; puede dar una buena idea del porqué del

diseño de Internet). En el modelo ARPAnet, la comunicación ocurre entre una computadora fuente y una destino. La red asume por sí misma que es falible; cualquier parte de la red puede desaparecer en cualquier momento (por ejemplo los cortes de cables). La red fue diseñada para requerir un mínimo de información de las computadoras que forman parte de ellas. Para enviar un mensaje en la red, una computadora sólo tiene que poner la información en un *sobre*, llamado paquete de protocolo Internet (IP:Internet Protocol) y le asigna el domicilio destino en forma correcta. Las computadoras que se comunican -no la red- tienen la responsabilidad de asegurar que la comunicación se lleve a cabo. La filosofía era que cada computadora en la red se pudiese comunicar, como un elemento individual, con cualquier otra computadora.

Esta red (la antecesora de la actual Internet) creció en forma acelerada debido a que los programas fuentes del protocolo de comunicaciones fueron de dominio público, por lo que los diferentes fabricantes incorporaron TCP/IP a sus equipos. Uno de los principales objetivos de TCP/IP fue el poder interconectar equipo heterogéneo eliminando de esta manera la barrera impuesta por los protocolos propietarios tales como el de IBM (SNA), Digital (DECnet), Xerox (XNS), etc.

Los desarrolladores de Internet en Estados Unidos, el Reino Unido y Escandinavia, en respuesta a las presiones del mercado, empezaron a poner el Software de IP en todo tipo de computadoras: se llegó a convertir en el único método práctico para comunicar computadoras de diferentes fabricantes. Esto resultó muy práctico para el gobierno y las universidades, quienes no tenían políticas que especificaban la compra de una determinada marca de computadoras. Todos compraron la computadora que mejor les pareció y esperaban poder comunicarse con otras en la red.

Al mismo tiempo que Internet se consolidaba, las redes locales Ethernet eran desarrolladas. La tecnología de redes locales maduró hasta 1983, cuando aparecieron

las primeras estaciones de trabajo para escritorio y las redes locales se multiplicaron. La mayor parte de las estaciones de trabajo tenían el sistema UNIX de Berkeley instalado, que incluía el software de red IP. Esto creó una nueva demanda: en lugar de conectar una computadora de tiempo compartido en un centro de cómputo, las organizaciones requerían conectar toda su red local a ARPAnet, lo cual permitiría que todas las computadoras que estuviesen en la red usaran los servicios de ARPAnet. Al mismo tiempo, muchas compañías y otras organizaciones empezaron a construir redes privadas usando los mismos protocolos de comunicación de ARPAnet, es decir, IP y sus protocolos asociados. Parecía obvio que si estas redes podían comunicarse entre sí, los usuarios de una red podrían comunicarse con usuarios de otra y todo el mundo sería beneficiado.

La Red Internet está compuesta actualmente por entidades autónomas que van desde universidades, centros de investigación, oficinas gubernamentales, grandes corporativos, etc., por lo tanto, la administración y las políticas de tráfico difieren grandemente entre dichas redes; algunas permiten el libre paso del tráfico mientras que otras lo prohiben proporcionando servicios de red restringidos a sus usuarios.

La demanda sigue creciendo ahora que la mayoría de las universidades están conectadas a la Internet, se está tratando de incluir también a primarias y secundarias, al igual que a las bibliotecas locales. Las personas que recientemente se graduaron en una Universidad saben para qué sirve Internet y hablan con sus compañeros de trabajo acerca de conectar la empresa en que laboran a dicha red.

Toda esta actividad apunta a un crecimiento continuo, a la solución de los problemas de conectividad, a la evolución de las tecnologías y a la seguridad en el empleo de los expertos en comunicaciones. Mucha gente va más allá; pues una vez que se cuenta con una conexión de red en el trabajo, el siguiente paso lógico es querer conectarse directamente desde la casa.

3.4.1 ¿Cómo trabaja Internet?

El sistema de redes moderno está constituido sobre el concepto de "niveles o capas de servicio". Usted comienza tratando de mover bits de un lugar a otro, perdiendo algunos en el trayecto. Este nivel se compone de cables y hardware, y no necesariamente de cables muy confiables. Después, agrega una capa de software básico que permite aislar los problemas de hardware. Incorpora otra capa de software para darle al básico algunas características deseadas. Continúa agregando funcionalidad e inteligencia a la red, capa por capa, hasta que se obtiene algo amigable y útil.

El Modelo TCP/IP

Una red de cómputo está formada por un conjunto de computadoras interconectadas para compartir recursos e intercambiar información. Con el fin de permitir que distintas arquitecturas diseñadas por diferentes fabricantes puedan interactuar entre sí, la Organización de Estándares Internacionales (ISO, por sus siglas en inglés) inició en 1977 la definición de un modelo de referencia llamado el *Modelo de Interconexión de Sistemas Abiertos* (o modelo OSI), que divide la arquitectura de red en 7 capas, cada una con funciones específicas independientes de las demás.

En este modelo, la capa N en una computadora realiza sus funciones comunicándose con la capa del mismo nivel en la otra computadora. Esta comunicación se lleva a cabo a través de reglas bien definidas -*protocolos*- para una arquitectura de red en particular.

Las funciones de la capa N sirven para que ésta pueda ofrecer un *servicio* a la capa inmediatamente superior en la misma computadora. De esta manera se van

agregando servicios conforme se asciende por las capas de la arquitectura hasta llegar a la capa de aplicación. El modelo OSI no define protocolos ni servicios, éstos dependen de la implantación específica a cada arquitectura de red. Lo que el modelo define es la función que debe realizar cada capa.

El modelo de red definido por el conjunto de protocolos TCP/IP es mucho más simple que el definido por otras arquitecturas de red.

En particular TCP/IP sólo define 3 capas del modelo OSI:

Aplicación
Transporte
Red

Llama la atención la ausencia de protocolos en las capas de sesión y presentación. Lo que ocurre es que TCP/IP se definió antes que el modelo OSI, y en un ambiente donde la necesidad de los servicios ofrecidos por estas dos capas no era evidente.

Redes de conmutación de paquetes

Cuando uno trata de imaginar qué es Internet y cómo opera, es normal pensar en un sistema telefónico. Después de todo, ambos son electrónicos y permiten abrir una conexión y transferir información, e Internet está compuesta principalmente por líneas telefónicas permanentemente dedicadas a este uso. Desafortunadamente, esto crea una idea errónea y provoca mucha confusión sobre la forma en que opera Internet. La red telefónica es una red de conmutación de circuitos. Cuando usted habla por teléfono, se separa una parte de la red para dedicarla a atender su llamada. Aún cuando no esté utilizando su parte de la red (por ejemplo cuando la línea está en

espera), ésta es inaccesible para otras personas; lo que provoca una subutilización de un recurso muy costoso: la red.

Un mejor modelo para comparar Internet es el Servicio Postal. El Servicio Postal es una red de conmutación de paquetes. Usted no cuenta con una parte de la red dedicada a sus actividades. Lo que se quiere enviar se mezcla con los mensajes de otras personas, se pone en un conducto, se transfiere a otra oficina postal y se clasifica todo nuevamente. Aunque las tecnologías son completamente diferentes, el Servicio Postal muestra una analogía sorprendentemente similar.

El Protocolo Internet, IP

Un cable puede llevar información de un lugar a otro. Sin embargo, ya se sabe que Internet puede hacer que la información llegue a distintos lugares distribuidos en todo el mundo. ¿Cómo es esto?

Las diferentes partes de Internet están conectadas por un conjunto de computadoras llamadas *enrutadores*, que interconectan las redes. Estas redes pueden ser Ethernet, token ring o en ocasiones líneas telefónicas.

Las líneas telefónicas y las redes Ethernet son equivalentes a los camiones y aviones del Servicio Postal. Son el medio a través del cual el correo va de un lugar a otro. Los enrutadores son sucursales postales; estos equipos deciden cómo dirigir la información ("paquetes"), de la misma forma que una oficina postal decide cómo distribuir los sobres por correo. No toda subestación o todo enrutador cuentan con una conexión a cada uno de los otros enrutadores de la red. Si se envía un sobre de correo desde Dixville Notch, New Hampshire, con destino a Boonville, California, la Oficina Postal no reserva un avión de New Hampshire a California para llevarlo, sino que envía el sobre a la sucursal de correo y ésta a su vez lo envía a otra, y así sucesivamente hasta alcanzar su destino final. Esto significa que cada subestación

sólo necesita conocer las conexiones con las que cuenta y cuál es el mejor "siguiente salto" para acercar el paquete a su destino. Internet trabaja de manera similar: un ruteador se fija en el destino de la información y decide a dónde enviarla. El enrutador elige cuál es el enlace más apropiado para enviar la información.

¿Cómo sabe la red a dónde se dirige la información? Si se quiere enviar una carta, no basta con poner el papel escrito en el buzón y esperar a que sea entregado. Es necesario poner el papel con la información en un sobre, escribir el domicilio del destinatario y pegar los timbres postales. De la misma manera que la Oficina Postal tiene reglas que definen la operación de su red, también existen reglas que definen la operación de Internet. Las reglas son llamadas protocolos. El Protocolo Internet (IP) se hace cargo de establecer domicilios o se asegura de que los enrutadores sepan qué hacer con la información que les llega. Continuando con la analogía de la Oficina Postal, el Protocolo Internet trabaja justo como un sobre, ver figura 3.6.

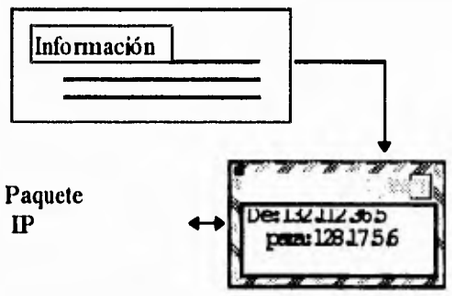


Figura 3.6 Sobres IP

Una parte de la información del domicilio va al principio del mensaje; estos datos dan a la red información suficiente para hacer la entrega del paquete.

Los domicilios de Internet constan de cuatro números, cada uno menor de 256. Cuando dichos números se escriben se separan por puntos, como se muestra a continuación:

192.118.39.8
128.172.6.9

No debemos preocuparnos por memorizar estos números para poder utilizar la red, en efecto, el domicilio está compuesto por varias partes. Como Internet es una red de redes, los primeros números del domicilio indican a los enrutadores cuál es la red a la que usted pertenece. Los últimos números indican qué computadora personal o equipo anfitrión de la red debe recibir el paquete, bajo este esquema cada computadora de Internet tiene un domicilio único. De nuevo el Servicio Postal es una buena analogía. Considere el domicilio "Río Pánuco 170, Cuauthémoc, D.F.". La parte "Cuauthémoc, D.F." es como la parte del domicilio correspondiente a la red, el cual permite llegar al sobre llegar a la Oficina correcta, que es la que tiene la información acerca de las calles en un área determinada. La parte "Río Pánuco 170" es como el domicilio del equipo anfitrión; éste identifica a un buzón particular en el área de servicio de la Oficina Postal. La Oficina Postal concluye su trabajo cuando entrega el correo a la oficina local correcta y ésta lo pone en el buzón correcto. De la misma forma, Internet concluye su trabajo cuando los enrutadores llevan la información a la red local correcta y ésta entrega dicha información a la computadora personal o equipo anfitrión correctos, localizados en dicha red.

Por muchas razones prácticas (sobre todo por limitaciones de hardware), la información enviada a través de las redes IP se divide en pedazos de tamaño distinto, llamados paquetes. La cantidad de información en un paquete normalmente se encuentra entre 1 y aproximadamente 1500 caracteres de largo. Esto previene que cualquier usuario monopolice la red, permitiendo que todos tengan un acceso equitativo. También significa que cuando la red se sobrecarga, su comportamiento sólo desmerece un poco para todos los usuarios: la red no se inutiliza cuando algunos usuarios la monopolizan.

Una de las propiedades más impresionantes de Internet es que, en un nivel básico, el protocolo IP es todo lo que se necesita para participar en la red.

No obstante, habrá que resolver varios problemas:

- La mayoría de las transferencias de información es mayor que 1500 caracteres.
- En ocasiones se presentan errores: ocasionalmente los paquetes pueden ser extraviados, pero esto se resuelve exitosamente.
- Los paquetes pueden llegar en desorden.

Para evitar esto, la siguiente capa de la red nos permitirá enviar grandes cantidades de información y corregirá las alteraciones que puedan ser causadas por la red.

El protocolo de control de transmisión (TCP)

TCP es el protocolo que se menciona frecuentemente junto con el IP y que se utiliza para resolver los problemas mencionados. El protocolo TCP toma la información que se desea enviar y la divide en segmentos. Además, enumera cada segmento para que el receptor pueda verificar la información y ponerla en el orden adecuado. Para que el protocolo TCP pueda enviar esta secuencia de números a través de la red, cuenta con su propio sobre que le permite "escribir" en él la información requerida para su reordenamiento. Un segmento de la información a transmitir se coloca en el sobre del protocolo TCP. Este sobre es puesto, a su vez, dentro del sobre del protocolo IP y posteriormente es transmitido a la red. Una vez que se pone algo en un sobre IP, la red lo puede transmitir.

Del lado del destinatario, una parte del software de TCP reúne los sobres, extrae la información de ellos y la pone en el orden adecuado. Si algún sobre se pierde en la transmisión, el receptor solicita su retransmisión al emisor. Una vez que el protocolo TCP tiene toda la información en el orden adecuado, la pasa a la aplicación del programa que esté utilizando sus servicios.

La descripción anterior del funcionamiento del protocolo TCP es ligeramente utópica. En la realidad, los paquetes no sólo se pierden, además de esto pueden ser modificados por el mal funcionamiento durante la transmisión a través del medio. TCP también resuelve este tipo de problemas. Así como coloca la información en un sobre, el protocolo calcula algo llamado número de verificación (checksum). Ese número permite que el receptor TCP detecte errores en el paquete transmitido.

Cuando un paquete llega a su destino, el receptor calcula el número de verificación y lo compara con el enviado por el transmisor. Si no coinciden, significa que ocurrió un error en la transmisión. El receptor deshecha el paquete y solicita la retransmisión.

3.4.2 Servicios disponibles

En Internet existen muchos recursos disponibles, pero no hay una lista oficial. Cualquiera que tenga una conexión a Internet puede poner en línea un recurso nuevo en cualquier momento sin avisar a nadie, y de hecho así ocurre. De manera que el truco es encontrar qué está disponible. En esta sección se tratarán algunos recursos conocidos que son considerados muy especiales. Las listas de los mismos son muy útiles, pero no dicen la última palabra. Nunca están al día, siempre les faltan los avances más recientes y novedosos de la red.

Otro problema es que la calidad de recursos de la red varía enormemente. Mientras que las listas tratan de enfocarse sobre todo a los "mejores" recursos que se

pueden encontrar, también es cierto que lo que es bueno para unos no lo es para otros. Para poder llegar a emplear correctamente la Red, el usuario debe aprender a escoger lo que realmente es útil entre todo lo que está disponible. Entre las herramientas más importantes con las que cuenta Internet se encuentran:

telnet: Se utiliza para establecer sesiones de trabajo en las computadoras de Internet y para tener acceso a muchos servicios públicos, que incluyen catálogos de bibliotecas y otros tipos de bases de datos.

ftp: Sirve para transferir archivos de un lugar a otro. Su mayor uso está en la recuperación de archivos de depósitos públicos localizados en toda Internet. A estos depósitos se les llama FTP anónimos debido a que no es necesario contar con una clave de usuario para poder tener acceso a ellos.

correo electrónico. Permite enviar mensajes desde y a cualquier parte de la Red.

USENET News: Le permite leer y colocar mensajes que han sido enviados a los "grupos de interés" públicos. Esto puede sonar no muy claro, pero realmente se refiere a los tableros de foros o grupos de discusión. USENET es el servicio de tableros de discusión más grande del mundo.

TELNET

Telnet es el protocolo de sesión de trabajo remota de Internet. Le permite estar frente al teclado de una computadora y establecer una sesión en una computadora remota en la red. *Telnet* ofrece sus servicios sobre el protocolo de

transporte TCP. La sesión puede ser en una máquina en la misma oficina, en la misma universidad, o al otro lado del mundo. Cuando usted se conecta es como si su teclado estuviese conectado a esa computadora remota. Podrá tener acceso a todos los servicios que esa máquina provee a sus terminales locales. Puede realizar una sesión interactiva normal (conectándose y tecleando comandos) o tal vez tener acceso a muchos servicios especiales, como buscar en algún catálogo de biblioteca, saber que está pasando en teoría, tener acceso al texto del periódico *USA Today* y aprovechar muchos de los servicios disponibles en los diferentes equipos de la red. La forma más sencilla de usar telnet es teclear:

telnet domicilio_internet_de_computadora_remota

en el nivel de comandos. Cuando termina la sesión en el sistema remoto, telnet termina su ejecución. Cualquier comando que se dé será ejecutado por el sistema local.

¿Que ocurre realmente?

Se hará una revisión más profunda para ver que es lo que pasa cuando se inicia una sesión de TELNET. Una aplicación consta de dos partes de software que colaboran entre sí: el *cliente*, que corre en la computadora que solicita el servicio, y el *servidor* que corre en la computadora que provee el servicio. La red es el medio que permite la comunicación entre ambos.

El cliente, que es el programa que corre en su computadora cuando se teclea el comando telnet, deberá hacer lo siguiente:

- Crear una conexión de red TCP con un servidor.

- Aceptar lo que se escriba en el teclado proveniente del usuario, de manera conveniente.
- Reformatear la entrada de datos a un formato estándar y enviarla al servidor.
- Aceptar la respuesta del servidor en un formato estándar.
- Reformatear esa salida para desplegarla en la pantalla del usuario.

El software del servidor corre en la máquina que provee el servicio; si el servidor no está funcionando, el servicio no está disponible. En los sistemas UNIX, los servidores son conocidos como *daemons,* procesos que corren residentes en memoria todo el tiempo. Estos *auxiliares silenciosos* esperan a que sus servicios sean requeridos y cuando esto sucede comienzan a trabajar. Cuando un servidor típico está listo para aceptar solicitudes de servicio, realiza lo siguiente:

- Informa al software de red que está listo para aceptar conexiones.
- Espera una solicitud en un formato estándar.
- Atiende la solicitud.
- Envía los resultados de regreso al cliente en un formato estándar.
- Vuelve a entrar en el proceso de espera.

Transferencia de archivos: FTP

Muy comúnmente será posible encontrar información en Internet que no querrá examinar mientras se encuentra conectado a algún sistema remoto: sería muy bueno contar con una copia para usted. En muchos de los casos, es necesario trasladar una copia del archivo a su computadora para poder trabajar con ella ahí. La herramienta para lograr esto es *ftp.*

Este comando recibe el nombre de *ftp* debido al protocolo de aplicación que usa: el *Protocolo de Transferencia de Archivos* (FTP: File transfer Protocol). Como su nombre lo indica, la función del protocolo es transferir archivos de una computadora a otra, no importando dónde se localicen, cómo estén conectadas o si tienen o no el mismo sistema operativo. Dado que ambas computadoras "hablan" el protocolo FTP y tienen acceso a Internet, es posible utilizar el comando *ftp* para transferir archivos. Algunas de las características de su uso cambian con cada sistema operativo, pero la estructura básica de comandos es la misma en cualquier máquina.

Al igual que *telnet*, *ftp* ha provocado la proliferación de una vasta gama de bases de datos y servicios. De hecho, usted puede además encontrar cualquier cosa, desde opiniones legales y recetas, hasta software *gratuito* en una gran cantidad de bases de datos en líneas disponibles, o depósitos, a los que usted puede tener acceso mediante *ftp*.

Ftp es un programa complejo porque tiene muchas formas diferentes de manejar archivos a estructuras de archivos. Las diversas formas de guardar archivos (en binario o ASCII, comprimidos o sin comprimir, etc.) introducen algunas complicaciones y se requiere de conocimientos adicionales para hacer las cosas correctamente. En seguida se discutirán algunos aspectos referentes al tipo de transferencia y el FTP anónimo, el cual es un servicio especial que permite el acceso a bases de datos públicas sin necesidad de obtener una cuenta.

Transferencias ASCII y Binarias

El comando FTP tiene dos formas comunes de transferir datos, llamadas *binaria* y *ASCII*. En una transferencia binaria se preserva la secuencia de bits del archivo, de tal forma que el original y la copia del archivo sean idénticas bit por bit,

aunque el archivo contenga una secuencia de bits que no tenga ningún significado en la máquina receptora.

El concepto del modo ASCII realmente no es apropiado: debería llamarse modo *texto*. En el modo ASCII las transferencias son tratadas como un conjunto de caracteres, el cliente y el servidor tratan de asegurar que los caracteres que se transfieren tengan el mismo significado tanto en la máquina receptora como en la emisora.

La gran ayuda de transferir en ASCII es porque se desconoce de que tipo de máquina se están tomando y realmente no importa, de lo único que hay que asegurarse es de poder leer los archivos en la máquina local. Para hacer que FTP trabaje en modo ASCII, se debe teclear el comando *ascii*. Para cambiar a modo binario, se debe teclear el comando *binary*.

Es importante saber el tipo de información que se desea transferir. La tabla 2 proporciona algunos tipos comunes de archivos. Los archivos ejecutables generalmente son binarios, sin embargo, existen algunas excepciones. Los programas que son compilados y ejecutados por el mismo procesador siempre son binarios.

Unix provee varios lenguajes (incluyendo al Shell) con los cuales es muy común escribir programas con comandos básicos del sistema operativo, regularmente estos programas se almacenan en un archivo siendo de tipo texto.

En UNIX, es posible usar el comando **file** para saber el tipo de la mayoría de los archivos. Esta utilería no fue escrita tomando en cuenta a FTP, por lo que proporciona más información de la que realmente necesita.

Archivo	Modo
Archivo de texto	ASCII por definición
Hoja de Cálculo	Probablemente binario
Archivo de base de datos	Probablemente binario, posiblemente ASCII
Archivo de procesador de palabras	Probablemente binario, posiblemente ASCII
Programa en código fuente	ASCII
Mensajes de correo electrónico.	ASCII
Archivos encapsulados UNIX	ASCII
Archivos tar de UNIX	Binario
Archivo de respaldo	Binario
Archivo comprimido	Binario
Archivo en formato uencode o binhexed	ASCII
Archivo ejecutable	Binario
Archivo postScript (impresora láser)	ASCII
Documento en Hipertexto (HTML)	ASCII
Archivos de imagen (GIF,JPEG,MPEG)	Binario

Tabla 2

FTP Anónimo

Con FTP anónimo no se tiene la necesidad de tener una clave de usuario o una contraseña para poder tener acceso a archivos en una máquina. Obviamente existen algunas restricciones: normalmente los usuarios anónimos sólo pueden ejecutar el comando get de los archivos (esto es, copiarlos), no podrán instalar

archivos nuevos o modificar archivos que ya existen, lo que sería equivalente a ejecutar el comando put. Y existen límites estrictos sobre que archivos se pueden copiar.

Cuando se habilita un servidor FTP anónimo, se crea una clave de usuario especial llamada *anonymous*. Si usted inicia FTP, establece una sesión en una computadora remota y da la clave de usuario *anonymous*, FTP aceptará cualquier conjunto de caracteres como contraseña. Se considera de buen gusto usar como contraseña su domicilio de correo electrónico, para que los administradores del servidor puedan comunicarse con usted en caso necesario (de hecho algunos sistemas empiezan a demandar que se proporcione un domicilio de correo electrónico válido para permitir el acceso). Después de haber establecido la conexión mediante la clave de usuario *anonymous*, podrá obtener con el comando *get* aquellos archivos que estén permitidos expresamente a los usuarios de los servidores FTP anónimo. Una vez que se ha inicializado la sesión de FTP es posible cambiarse a otros subdirectorios dando el nombre o moverse al directorio "padre" usando el argumento (..), y el uso de varios comandos más.

Correo Electrónico

El correo electrónico difiere de las otras aplicaciones porque no es un servicio de usuario a usuario: no es necesario que las máquinas emisora y receptora del correo electrónico se comuniquen directamente entre sí. Al correo electrónico se le conoce como un servicio de *almacenaje y reenvío*. El correo pasa de una máquina a otra hasta que llega a su destino final.

El Servicio Postal opera como una red de *almacenaje y reenvío*. Se escribe el domicilio en el sobre y se deposita en el buzón. La carta es recogida por un camión para llevarla a otro lugar y almacenarla. Ahí se clasifica y se reenvía a otro lugar.

Este paso se repite hasta que llega a su buzón destino. Si el buzón destino no está en el área de cobertura del Servicio Postal, el mensaje se envía al servicio postal del país destino.

A partir de esta analogía es posible inferir un par de cosas sobre Internet. Primero, si pone el domicilio correctamente a un mensaje, la red se hará cargo de entregarlo. No se necesita estar muy al tanto de lo que pasa. Es posible inferir también que el correo puede viajar entre Internet y otras redes de correo. Esto es cierto, pero el domicilio se hace más complejo si se requiere que un mensaje vaya de alguna red externa a Internet y viceversa.

Justo como en el Servicio Postal, si el emisor y el receptor no están conectados en la misma red, es necesario colocar el mensaje en algún lugar en donde se concentre todo el correo que vaya a una determinada red. Los puntos de conexión entre redes de correo electrónico se denominan *puertas de aplicación*. Se les llama *puertas* porque pueden ser vistos como puertas mágicas entre dos mundos; son *puertas de aplicación* porque conocen las aplicaciones de correo electrónico de ambos lados, para que los mensajes sean reformateados a una forma congruente al pasar de una red a otra. Para enviar correo a través de una puerta, casi siempre es necesario proporcionar en el domicilio del mensaje información sobre cómo llegar a la puerta e información y sobre como entregar el correo en la otra red.

Listas de distribución de correo

Con el correo electrónico, existe la misma facilidad de enviar un mensaje tanto a un grupo como una persona. La herramienta para poder hacer eso se llama *lista de distribución de correo.* Permite que un alias o un sobrenombre represente a un grupo de destinatarios; por ejemplo, el alias **staff** puede definir a un grupo al que

pertenecen *todos los empleados* de un negocio. Cuando se envía un mensaje a **staff**, el correo se distribuye a todos los que pertenecen al grupo.

Una vez que el alias es creado, es posible enviar un mensaje al grupo que se será entregado a sus miembros. Este es un medio para implantar grupos de discusión a través del correo electrónico. Funciona bien para grupos pequeños o para grupos personales que solo son usados por un usuario en particular. Al crecer el grupo y cuando existen otras personas que quieren usar la misma definición de grupo, la administración del grupo se convierte en algo muy complicado. Cada vez que se incorpora o se elimina a alguien del grupo, todos los miembros de éste tendrán que modificar su alias personal. Esto no se lleva a cabo por todos, así que casi siempre alguien queda fuera del grupo y no recibe el mensaje y aquí es donde empiezan los problemas.

Lo que realmente se necesita es mantener una lista de correo en forma centralizada, para que cuando se realicen cambios surtan efecto en todos los miembros. Es común implantar esto con un *reflector de correo*. Un reflector de correo es un domicilio especial de correo electrónico configurado para que, cualquier mensaje enviado a él, automáticamente sea reenviado a todos los participantes de la lista. El reflector de correo funciona bien para un grupo privado, que puede ser grande. Pero ¿Qué pasa si en lugar de ser una lista privada, existiera la posibilidad de que cualquier persona pudiera ingresar a la lista de discusión?. Si éste es el caso; el usuario que desea participar en la lista de discusión debe de hacer una petición, donde él indica lo siguiente: *Por favor, inclúyame en la lista*. Enviar un mensaje como éste a la lista no es una solución razonable, porque la lista enviará el mensaje a todos los integrantes. Hacer esto puede funcionar, pero se considera de mal gusto, porque hará que el usuario sea considerado impertinente entre los integrantes de la lista, y peor aún, el administrador de la lista podría no recibir su mensaje. La persona que mantiene la lista puede ser al administrador de algún

sistema de correo al que no le importa el tema de discusión. Por desgracia, la forma correcta de suscribirse a una lista depende de cómo sea administrada. Por tradición, Internet utiliza comúnmente domicilios especiales para solicitudes de este tipo.

Siempre que se crea un reflector de correo público, también se crea un segundo buzón de correo en la misma máquina. Este buzón de correo especial es *cerrado*, nada de lo que se recibe ahí es distribuido sino que se envía a la persona que mantiene la lista.

Network News

Network News brinda servicio a través de servidores de noticias, es el equivalente en Internet a los grupos de discusión. Network News representa un medio para tomar parte en muchas pláticas más, pero manteniéndolas organizadas y separadas de su correo electrónico. Los artículos de News tienen otra ventaja: son ideales para visualizar y navegar entre ellos y no es necesario que se comprometa profundamente. Para el usuario, dentro de Network News se organizan pláticas bajo un conjunto de amplios apartados que son conocidos como grupos de interés (newsgroups). Un programa despachador de artículos se encarga de presentar tales pláticas de manera ordenada. Mantiene el rastreo sobre los temas que ya vio y solo muestra los temas nuevos que llegaron desde la última sesión. Una vez que el despachador de artículos ha mostrado que artículos están disponibles sobre determinado tema, puede seleccionar y leer los temas que son de interés.

Archie

Históricamente, uno de los grandes problemas de Internet ha sido encontrar información en la red. Los servidores FTP anónimo surgieron rápidamente para

brindar la oportunidad de buscar archivos en la Red. Dentro de ella podemos encontrar cualquier cantidad de información e inclusive software o programas de dominio público (recuerde que se denomina "de dominio público" al software que es ofrecido a la comunidad en general para su uso libre no comercial, respetando la autoría del mismo), pero es un poco difícil. Suena como un trabajo solo para computadoras, pero se puede resolver con ***Archie***, un sistema que permite explorar índices disponibles en los servidores públicos especiales. Archie es un servicio que surgió en la Universidad McGill, primeramente de la manera más sencilla. Se fue buscando por toda la red y preguntando a quienes estuvieran ejecutando programas servidores FTP anónimo para catalogarlos. Los perpetradores ejecutan su programa una vez al mes, el cual se conecta con los servidores mencionados a través de FTP. Cuando se enlaza con los servidores, construye un directorio para listar todos los archivos que se encuentran en cada uno de ellos, usando comandos de FTP estándar. Al realizar una búsqueda, Archie explora los directorios catalogados y devuelve los nombres de los archivos que concuerdan con la cadena proporcionada, junto con los nombres de los servidores que contienen disponibles tales archivos. Así fue creado el servicio básico de Archie. Un problema al que se enfrentaba este servicio es que había ocasiones en que se realizaban búsquedas de cadenas muy largas o palabras muy extrañas, es por eso que los creadores de Archie pidieron a las personas responsables de cada servidor que les enviaran la información sobre los paquetes más importantes que manejaran, y utilizaron esta información para crear un servicio llamado ***whatis***. Se trata de un conjunto de palabras clave alternativas y catalogadas para archivos de la red, que pueden ser utilizadas para localizar software o archivos de datos, incluso si el nombre del archivo no guarda ninguna semejanza con su contenido. Cuando aumentó el uso de Archie, el servicio se transformó para satisfacer la demanda que iba en aumento. En la actualidad, existen muchos servidores Archie repartidos por toda la Internet. Cada servidor construye un índice

de archivos FTP cerca de él y después los diversos servidores comparten información. Esto permite que las actualizaciones sean más oportunas, sin sobrecargar la red. Con esto se debe empezar si se está buscando programas, datos o archivos de texto. Actualmente, Archie cataloga cerca de 1200 servidores y 2.5 millones de archivos. Como usuario, se pueden solicitar que se encuentren nombres de archivo que correspondan a ciertos criterios de búsqueda o que se muestren archivos que contengan ciertas palabras. Archie devuelve los nombres de archivo que concuerdan con el criterio de búsqueda y el nombre de los servidores que contienen esos archivos. Una vez que se decida cuál de esos archivos satisface mejor las necesidades, puede ser trasladado a la computadora local con FTP anónimo.

Gopher

Gopher es un servicio que permite buscar recursos utilizando menús. El nombre "Gopher" (tuza) comenzó como un servicio de distribución interna en el campus de la Universidad de Minnesota, hogar de los "Golden Gophers". El nombre **gopher** fue acuñado debido a que, al igual de las tuzas, su principal función consiste en "ir por" ("go per") cosas. El servicio fue diseñado para que cada parte de la burocracia pudiera tener control sobre su propio servidor y sus propios datos. Es decir, que la administración de la escuela pudiera contar con una computadora en su edificio administrativo para distribuir información del personal administrativo; que el departamento deportivo pudiera tener un servidor para la agenda deportiva en sus oficinas; que cada departamento académico contara con un servidor para proporcionar los horarios de clases, etc. Podrían existir tantos servidores como grupos que quisieran proporcionarlos. Los inventores de Gopher crearon una aplicación especial que pudiera guiar a los estudiantes a la información que requerían, sin necesidad de una capacitación. Para lograr esto, organizaron el

sistema por temas de manera que es posible verlo como una enorme base de datos, en lugar de cientos de bases de datos pequeñas interconectadas. Se puede tener acceso a archivos FTP, catálogos de biblioteca y otros servicios de bases de datos con propósitos especiales (basados en Telnet). Pero sólo Gopher sabe dónde se localizan realmente los datos, la forma de tener acceso a ellos y que existen muchos servidores que los proporcionan.

No tomó mucho tiempo darse cuenta de que si el sistema funcionaba bien para muchos servidores colocados en diferentes departamentos, también podía funcionar para servidores colocados en diferentes partes del mundo. Todo lo que se quería era que Internet enlazara a todos. En cuatro años, más o menos, la utilización del sistema Gopher creció de un servidor a más de 1300 servidores.

Al utilizar Gopher y cuando se ha encontrado algo de interés, esta información puede ser accedida, y no es necesario conocer la dirección IP ni cambiar de programas que permitan realizar la transferencia o la conexión a otros servidores. Todo es transparente para el usuario. La gran ventaja que ofrece Gopher no es tanto que ahorre la búsqueda de domicilios o nombres de recursos, o que no se tenga la necesidad de utilizar varios comandos para obtener lo que se desea. La ventaja real consiste en que permite navegar a través de los recursos de Internet, sin importar su tipo (texto, sonido, imagen, etc.). Gopher también cuenta con un servicio de consulta en línea a bibliotecas que ayudan a buscar títulos, temas o autores.

Gopher no permite tener acceso a cualquier cosa que no esté disponible por otros medios. No existe un formato especial de "recursos Gopher" para que se pueda tener acceso, por lo menos no lo hay en el mismo sentido que en los servidores FTP o los directorios donde algunos archivos puedan estar disponibles sólo a través de Gopher, pero es parte de la política de seguridad. Si se tiene acceso a esos archivos a través de Gopher, llegarán al sistema local a través de FTP. Una vez que se ha encontrado la información que se quiera revisar, Gopher sabe que aplicación debe

utilizar para obtener un elemento en particular (telnet, ftp, etc). Cada tipo de recurso se maneja de manera diferente. Sin embargo, todos son manejados de manera intuitiva, dependiendo del tipo de cliente Gopher que se esté utilizando. Gopher es lo suficientemente inteligente para imponer restricciones de licencias. Algún software o algunos recursos (por ejemplo periódicos en línea) pueden tener licencia para ser utilizados únicamente dentro de una cuidad o campus en particular.

Para tener acceso al sistema Gopher, se necesita un programa cliente ***gopher***. El programa cliente especial debe estar instalado en una computadora que se encuentre en Internet. Cada cliente tiene la "forma y las maneras" del sistema en que se está ejecutando. Casi todas las tareas que puede realizar con un cliente gopher dado, se pueden realizar con otras versiones del mismo. Cualquier cliente que se encuentre instalado, estará configurado con el domicilio de Internet del servidor del cual se extrajo.

WAIS

WAIS (Wide Area Information Service) es otro de los servicios de Internet. Sirve para realizar búsquedas a través de material indizado y localizar artículos basándose en su contenido. WAIS permite realizar exploraciones a través de archivos de Internet en busca de documentos que contengan ciertos grupos de palabras.

WAIS es una herramienta para trabajar con conjuntos de datos o con bases de datos y no examina los datos en el proceso de la búsqueda, examina el índice, puede seleccionar la información y presentarla sin importar el formato que tenga. Como Gopher, WAIS permite encontrar y tener acceso a recursos de información de la red sin importar el lugar exacto de residencia. En Gopher se encuentran los recursos informativos buscando a través de una secuencia de menús hasta hallar algo

apropiado. WAIS lleva a cabo la misma tarea, sólo que realiza la búsqueda completa. Un comando WAIS puede ser descrito de la siguiente manera: el usuario ordena "encontrar documentos acerca de algo en alguna biblioteca". Entonces WAIS revisa los documentos que existen en esa biblioteca (o bibliotecas) que le han sido indicados e informa cuáles son los documentos son los que probablemente contengan la información requerida. Después si así lo desea el usuario, WAIS despliega los documentos en pantalla.

En la red actualmente existen más de 500 bibliotecas WAIS gratuitas, y en este servicio se puede preguntar "¿Existe una biblioteca para este tema?".

Así se puede averiguar con facilidad si WAIS cuenta con los recursos informativos de utilidad para cualquier usuario.

WWW (World Wibe Web)

La World Wibe Web, también conocida como WWW, es el servicio de información de Internet más reciente. Web está basada en una tecnología llamada hipertexto. La mayor parte del desarrollo, ha sido realizada en el CERN, el Laboratorio Europeo de Física de Partículas. Aunque fue realizado en este laboratorio, no es una herramienta diseñada por y para físicos, a pesar de que estos científicos fueron quienes lo desarrollaron inicialmente, Web es una de las herramientas más flexibles para navegar por Internet, aún continua desarrollándose por lo que no debe sorprenderse si no trabaja como se espera.

Para hacer uso de Web el primer paso es conectarse vía telnet con info.cern.ch. Se trata de un examinador orientado a texto que funciona en cualquier terminal. Existen muchos examinadores disponibles, si se decide instalar uno en otro sistema (lo cual es altamente recomendable si se planea utilizar Web con frecuencia) se puede escoger el que más convenga entre examinadores orientados a texto para el

apropiado. WAIS lleva a cabo la misma tarea, sólo que realiza la búsqueda completa. Un comando WAIS puede ser descrito de la siguiente manera: el usuario ordena "encontrar documentos acerca de algo en alguna biblioteca". Entonces WAIS revisa los documentos que existen en esa biblioteca (o bibliotecas) que le han sido indicados e informa cuáles son los documentos son los que probablemente contengan la información requerida. Después si así lo desea el usuario, WAIS despliega los documentos en pantalla.

En la red actualmente existen más de 500 bibliotecas WAIS gratuitas, y en este servicio se puede preguntar "¿Existe una biblioteca para este tema?".

Así se puede averiguar con facilidad si WAIS cuenta con los recursos informativos de utilidad para cualquier usuario.

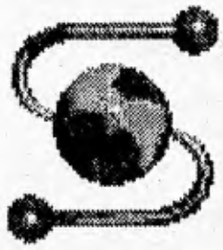
WWW (World Wibe Web)

La World Wibe Web, también conocida como WWW, es el servicio de información de Internet más reciente. Web está basada en una tecnología llamada hipertexto. La mayor parte del desarrollo, ha sido realizada en el CERN, el Laboratorio Europeo de Física de Partículas. Aunque fue realizado en este laboratorio, no es una herramienta diseñada por y para físicos, a pesar de que estos científicos fueron quienes lo desarrollaron inicialmente, Web es una de las herramientas más flexibles para navegar por Internet, aún continua desarrollándose por lo que no debe sorprenderse si no trabaja como se espera.

Para hacer uso de Web el primer paso es conectarse vía telnet con info.cern.ch. Se trata de un examinador orientado a texto que funciona en cualquier terminal. Existen muchos examinadores disponibles, si se decide instalar uno en otro sistema (lo cual es altamente recomendable si se planea utilizar Web con frecuencia) se puede escoger el que más convenga entre examinadores orientados a texto para el

sistema X Window y estaciones de trabajo, así como computadoras Macintosh y PC. WWW es la base para Mosaic, examinador que funciona en Unix bajo el sistema X Windows , lo mismo que en Macintosh y Microsoft Windows.

WWW es un intento de organizar toda la información en Internet, además de cualquier otra aplicación local que se necesite, a manera de un conjunto de documentos de Hipertexto. La cantidad de hipertexto que se encuentra en la Red, se ha incrementado considerablemente en los últimos años. Muchas exhibiciones de museo, revistas y otras presentaciones en hipertexto están disponibles, incluyendo a Global Network Navigator (GNN), de O'Reilly & Associates (ORA). El problema radica en la escasez de herramientas para construir la estructura de vínculos. La mayoría de los documentos de hipertexto actualmente disponibles se construyeron manualmente. Los editores de hipertexto son muy recientes y tienen mejores herramientas para crearlos. Todos los elementos que se encuentran en hipertexto pueden ser modificados en cualquier momento.



Capítulo 4

MOSAIC para sistema X-Window

Es indiscutible que día a día, Mosaic como ambiente gráfico es más popular. Actualmente con sus versiones para X-Window, Windows y Macintosh hacen posible la estandarización del manejo de información que se encuentra en la Internet. Los comandos básicos para telnet, ftp, Archie, Wais y algunos para el correo electrónico, son poderosos pero no muy intuitivos y el rápido crecimiento de usuarios en la Internet quienes no tienen el deseo ni paciencia de aprender el uso de esas interfaces, ha obligado al desarrollo de nuevos ambientes más amigables que permitan manipular y explotar los recursos de la red de la forma más eficiente. En el capítulo anterior discutimos algunos de ellos, y World Wide Web es la base para el desarrollo de Mosaic como visualizador gráfico de documentos hipertexto.

Con un visualizador gráfico para la WWW, el usuario puede ver documentos formateados conteniendo gráficas y ligas a otros que se encuentren en alguna otra parte de la Internet. No debemos olvidar que la WWW no es un programa, sistema o protocolo específico; se trata de un concepto. Técnicamente, la WWW no es más que un sistema distribuido de hipermedia[5] .

[5] conceptos discutidos en el capítulo 2.

Ahora, describiremos algunos conceptos importantes sobre el software en el cual desarrollamos nuestra aplicación presentando las características principales del mismo.

Como habíamos mencionado en la introducción, Mosaic forma parte del software del dominio público, por lo tanto, no tiene costo alguno para la UNAM, así que esa es la primera razón por la cual elegimos a este visualizador para la realización de nuestro proyecto, por otra parte, es el único que tiene capacidades multimedia para manejar hipertexto e hipermedia. Algo muy importante que debemos mencionar es el hecho de que también podemos realizar interfaces con otros programas de aplicación, esto es, no sólo nos limitamos a manejar hipertexto, sino que aprovechamos una de las ventajas que hacen a Mosaic poderoso, el poder comunicarse con bases de datos externas (discutiremos más detalles de esto en el capítulo 6 utilizando a SYBASE como ejemplo) a través de CGI's (ver sección 4.6.2), o bien, generar documentos en hipertexto desde un programa externo.

4.1 Descripción

NCSA Mosaic (NCSA: National Center for Supercomputing Aplications de la Universidad de Illinois en Urbana-Champaign) es un sistema de hipermedia distribuido, diseñado para compartir y recuperar información en Internet. Mosaic provee una interfaz unificada para varios protocolos, formatos de datos y archivos usados en Internet, habilita el poder de nuevos métodos para compartir, usar y recuperar información. Mosaic se encuentra disponible para tres tipos de plataformas: Macintosh, Microsoft Windows y el Sistema X Window con sistema operativo Unix.

Mosaic fue diseñado como un examinador de la World Wide Web, probablemente es el más agradable que existe. Presenta una interfaz multimedia para Internet. Realiza más tareas que la simple presentación de hipertexto con vínculos hacia otros documentos; se trata de una herramienta de hipermedia, lo cual significa que puede manejar audio, imágenes e incluso video (imágenes en movimiento) y es la interfaz con diferentes servicios. En algunas ocasiones, las herramientas especializadas realizan mejor una determinada tarea, pero si se está en condiciones de instalar sólo un software de navegación en el sistema, Mosaic es lo que se necesita.

Mosaic se basa en un modelo cliente/servidor para distribuir información. Un *servidor* instalado en una máquina en algún lugar de Internet puede responder a consultas enviadas por los *clientes* que también pueden estar localizados en cualquier parte dentro de la red. Las unidades de información enviadas por los servidores hacia los clientes son documentos, los cuales pueden contener texto con o sin formato, gráficas, sonido y otros datos para hacer multimedia, ligas (o también llamadas hiperligas) a otros documentos que pueden estar localizados en algún otro lugar dentro de Internet.

Mosaic está basado en la tecnología World Wide Web del CERN en Suiza y usa las bibliotecas de los clientes comunes de WWW. La primera versión oficial apareció en abril de 1993 para el sistema X-Window diseñada por Marc Andreessen y Eric Bina en NCSA, a partir de entonces tuvo un crecimiento explosivo dentro de la WWW, para el otoño del mismo año, la versión 2.0 llegó y con ella, las primeras versiones beta para Macintosh y Windows de Microsoft.

Dentro de un documento en Mosaic es fácil distinguir de entre todos los elementos que lo integran, las *ligas*. Estas son palabras o frases que se encuentran subrayadas y resaltadas (en ocasiones en color azul) que un con solo un *click* del *ratón* sobre ellas, realiza la conexión a un servidor apropiado en cualquier parte de

Internet y despliega el documento referenciado. Cabe señalar la importancia de las ligas, ya que es posible navegar en un número ilimitado de documentos y se puede mantener la liga anterior por si se desea regresar.

Un término muy utilizado en Mosaic es el de "home page". Cuando se ejecuta el programa, la primer pantalla que el usuario observa es el "home page" (ver Figura 4.1), que es un documento de presentación, por default lo toma del servidor de NCSA (donde Mosaic fue desarrollado), en él se puede ver la pantalla de presentación del sistema, y cuando nos estamos conectando a Mosaic por primera vez, también lo hacemos al servidor que se encuentra en la Universidad de Illinois, y dando un *click* con el *ratón* en cualquiera de las frases en color azul subrayadas, podemos automáticamente trasladarnos hacia otra pantalla o documento HTML. Si lo deseamos, podemos cambiar ese "home page" por uno diseñado por nosotros y que lo tome de nuestra máquina.

Para la completa interacción con Mosaic y para manejar una gran cantidad de datos con formato de tipo imagen, audio y video, es necesario contar con elementos externos a este programa ya que Mosaic no provee estas herramientas por sí solo. Estas herramientas que nos ayudaran a tener una mejor manipulación del documento son llamadas ***visualizadores externos*** los cuales son programas separados de Mosaic que se invocan cuando es necesario desplegar cierto tipo de información. Esto no quiere decir que si no los tenemos instalados Mosaic no va a funcionar, solo que la forma de apreciar el documento completo se encuentra restringida, ya que no es lo mismo ver una imagen que ver una en movimiento.

Una pieza de información de algún documento o varias de ellas pueden ser referenciadas, anotadas y almacenadas por Mosaic. El esquema que se usa para nombrar la fuente de la información sobre la gran red global es el mecanismo URL (Uniform Resource Locator) que puede hacer referencia a la misma residente en servidores FTP, HTTP, servidores de noticias NNTP, consultas a bases de datos

WAIS u otros documentos, búsquedas sobre servidores Gopher, archies, etc. El resultado es un completa transparencia de localización de datos y de elaboración de procesos.

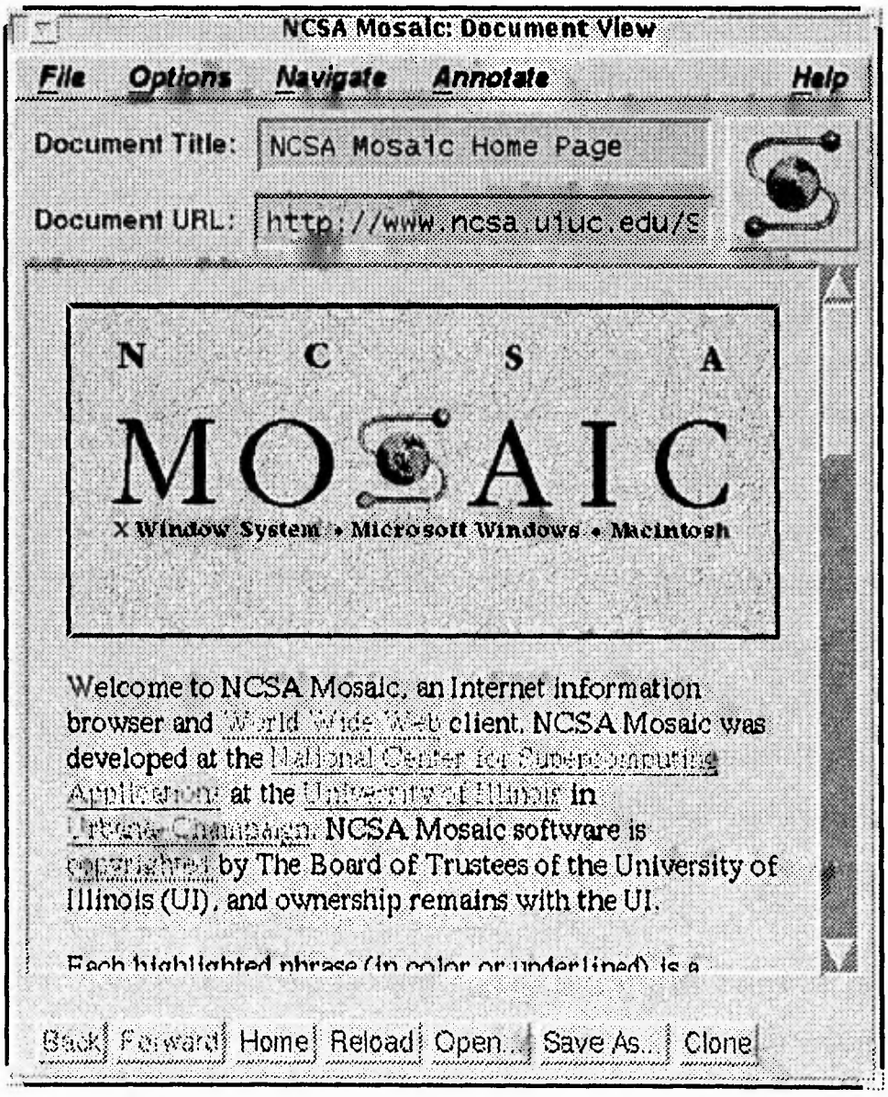


Figura 4.1 home page de Mosaic

4.2 Requerimientos

Antes de instalar NCSA Mosaic para un sistema X-Window, se necesita una estación de trabajo que maneje sistema operativo UNIX por ejemplo: Sun Sparc, IBM RS/6000, SGI, DEC, etc., así como el sistema de ventanas antes mencionado.

Mosaic utiliza ***visualizadores externos*** (mencionados la sección anterior) para poder desplegar la gran variedad de formatos de archivos (imágenes, video, audio, PostScript y otros) en la red Internet.

Para que Mosaic pueda desplegar este tipo de archivos, se recomienda tener los siguientes visualizadores considerados los más importantes.

ghostview

ghostview es una herramienta que nos permite visualizar documentos tipo PostScript, ghostview puede obtenerse del servidor FTP anónimo en la siguiente dirección: *uxc.cso.uiuc.edu* bajando los siguientes archivos:

/gnu/ghostscript-2.5.2.tar.Z
/gnu/ghostscript-fonts-2.5.2.tar.Z
/gnu/ghostview-1.4.1.tar

No debemos olvidar que para el caso de los archivos con terminación .Z, habrá que descomprimirlos mediante el comando *uncompress* de UNIX.

mpeg-play

mpeg-play se encarga de desplegar secuencias de animación y video en formato MPEG, se obtiene de la Universidad de California en Berkeley (y muchos otros lugares) del servidor FTP anónimo *toe.cs.berkeley.edu* recuperando el siguiente archivo:

/pub/multimedia/mpeg/mpeg_play-2.0.tar.Z

showaudio

Es un *script* en shell que puede reproducir varios formatos de audio en varias plataformas y es parte de las herramientas de correo de multimedia en la red. Podemos obtener *showaudio* del servidor FTP anónimo *thumper.bellcore.com* bajando el siguiente archivo:

/pub/nsb/mm2.x.tar.Z

(donde *x* se refiere a la versión que actualmente se encuentra disponible)

xv

Es un visualizador de imágenes tipo GIF, JPEG, TIFF y algunos otros formatos. Conectándose al servidor FTP anónimo *ftp.x.org* se puede obtener mayor información del mismo dependiendo de la máquina con la que se desee trabajar.

/contrib/xv-version.readme

4.3 Cliente

La interfaz de usuario dentro de Mosaic está dada por los clientes que han sido desarrollados por NCSA. Estos clientes fueron diseñados inicialmente para estaciones de trabajo corriendo sistema operativo UNIX y sistema X-Window, posteriormente se desarrollaron versiones para Macintosh y Windows de Microsoft.

Las primeras versiones de Mosaic tienen las siguientes funcionalidades:

- Despliegue gráfico de texto e hipertexto así como acceso a documentos en línea tales como: gráficas, imágenes, audio, secuencias de video y datos científicos desarrollados en hipermedia.
- Una interfaz de usuario gráfica para poder seleccionar el tipo de letra, opciones para pegar y cortar texto.
- Información de cómo navegar dentro del ambiente de Mosaic.
- *listas de interés* (hostlists) de documentos interesantes que puedan ser monitoreados durante la navegación con Mosaic.
- Capacidad para acceder bases de datos WAIS y realizar búsquedas dentro de la red.
- Acceso a documentos localizados en "cualquier lugar de la red".
- Un completo soporte de comunicaciones basado en TCP/IP incluyendo HTTP, Gopher, FTP, NNTP, WAIS, Archie y otras fuentes de datos.
- Información sobre la construcción de ambientes gráficos configurables.
- Manejo de correo y noticias editadas en hipermedia.
- Soporte para creación de filtros de información a través de grandes cantidades de información.

Para instalar Mosaic en un sistema X-Window, se debe hacer una conexión al servidor FTP anónimo *ftp.ncsa.uiuc.edu*, los archivos binarios ejecutables se encuentran en el directorio */Mosaic/Unix/binaries*.

La documentación técnica y planes futuros para Mosaic, se encuentra en el directorio */Mosaic/mosaic-papers*, el código fuente está disponible en el directorio */Mosaic/Unix/source*.

4.4 Servidor

Mosaic puede recuperar documentos desde cualquier parte de la red Internet haciendo una conexión con servidores FTP anónimos. Sin embargo, por razones de funcionamiento, se recomienda instalar un servidor conocido como *servidor HTTP* (HyperText Transfer Protocol).

NCSA ha desarrollado el código para servidores HTTP el cual se encuentra disponible en el servidor FTP anónimo *ftp.ncsa.uiuc.edu* bajo el directorio */Mosaic/ncsa_httpd*. Las instrucciones de instalación y configuración se encuentran disponibles para varias plataformas en ese mismo directorio.

En ocasiones el lector puede encontrar que comúnmente se habla de servidores HTTP. Sin embargo, pudiera encontrarse también al *httpd*, en estos casos cabe mencionar, que HTTP se refiere exclusivamente al protocolo de transferencia de hipertexto mientras que httpd se refiere al *deamon* o programa residente en la memoria de la estación de trabajo (HyperText Transfer Protocol deamon).

El código fuente y versiones precompiladas para instalar un servidor HTTP se encuentran disponibles en el servidor ftp anónimo *ftp.ncsa.uiuc.edu* bajo el directorio */Web/httpd/Unix/ncsa_httpd/current*.

4.4.1 El protocolo HTTP (HyperText Transfer Protocol)

Cuando muchas fuentes de información en la red están disponibles al lector, y cuando una referencia entre las mismas existe, es posible acceder a cada una de ellas como unidades de información independientes en forma rápida y eficiente, sobre todo cuando requerimos accederlas en diferentes máquinas. Como el tiempo de respuesta es un factor importante para transferir hipertexto, se requiere de un protocolo para la recuperación de información que realice esa operación en forma rápida. Este protocolo permite a los servidores manejar una variedad de métodos en la transferencia de información dentro de la red y nos referimos al HTTP (HyperText Transfer Protocol) o al protocolo de transferencia de hipertexto. Este protocolo especifica que una transacción entre un cliente y el servidor HTTP consta de las siguientes partes:

- **Conexión:**

  - El cliente hace una conexión TCP/IP a un host utilizando el *nombre* o la dirección IP del mismo, y el número de puerto dado en la dirección.
  - Si el número de puerto no se especifica, el número 80 será el default para HTTP.
  - El servidor acepta la conexión.

  Nota: HTTP corre sobre TCP.

- **Petición:**

  - El cliente envía una petición que consiste de una línea de caracteres ASCII terminada por un par de códigos: CR LF (Carriage Return, Line Feed) que son el retorno de carro y línea de alimentación.

- Esta petición se forma de la palabra "GET", un espacio, el nombre del documento omitiendo las partes "http:, host y número de puerto", debido a que son únicamente la referencia para hacer la conexión (ver sección 4.5 sobre URL's) y en este caso nos interesa el nombre del documento.

- **Respuesta:**

  - La respuesta a una petición es un mensaje en lenguaje HTML (ver sección 4.6).

  - Este formato permite que el mensaje sea regresado al cliente como hipertexto. El mensaje es terminado por el cierre de la conexión por parte del servidor.

- **Cierre de la conexión:**

  - Esta operación la realiza el servidor cuando el documento ha sido totalmente transferido. El cliente puede abortar la transferencia cerrando así la conexión antes de que esto ocurra, en caso que el servidor no devuelva alguna condición de error.

4.5 Los URL's (Uniform Resource Locator)

Muchos protocolos y sistemas para búsqueda y recuperación de documentos están actualmente en uso, y otros protocolos o refinamientos de los ya existentes están en un campo cuya expansión avanza rápidamente.

Esos sistemas intentan lograr una búsqueda global de documentos a través de diferentes plataformas de cómputo. A medida que los protocolos se desarrollan, los gateways permiten un acceso global cuando es posible. Sin embargo, hay un área en que es impráctico hacer conversiones, y ésta es en los nombres y direcciones usados

para identificar objetos. Esto es porque los nombres y direcciones se han pasado en diferentes maneras, desde sus primeros desarrollos hasta documentos hipertexto.

Una característica común de casi todos los modelos de datos es algo que puede ser concebido dentro del concepto de *objeto* y alguna clase de nombre, dirección o identificador asociado a él. Uno puede por lo tanto, definir un conjunto de nombres en el cual esos objetos existen.

Los sistemas prácticos necesitan acceder a objetos que son parte de diferentes sistemas existentes. Por lo tanto, el concepto de un conjunto universal de esos objetos formado por nombres y direcciones, llega a ser importante. Esto permite tratar a los objetos en un camino común, aunque tengan diferentes características.

En esta sección definimos un camino para encapsular un nombre y etiquetarlo produciendo un miembro de un conjunto universal. Tal miembro de este conjunto es conocido como un Identificador de Recursos Uniforme o URI (Uniform Resource Identifier). La sintaxis de un URI no involucra propiedades de nombres y direcciones en ese conjunto universal, éstas dependen de las especificaciones de los protocolos y convenciones usadas para cada uno.

Para protocolos de acceso existentes en Internet, es necesario en muchos casos definir una forma concisa y suficiente los términos de direcciones. Los URI's asociados a objetos accedidos con estos protocolos son conocidos como *Uniform Resource Locator* (URL) cuya sintaxis ha sido usada por el software de World-Wide-Web desde 1990.

El *objeto* es la unidad de referencia y no necesita corresponder a alguna unidad de almacenamiento. Nos referimos a objetos que pueden buscarse como "índices". En este capítulo, usamos el término *dirección* generalmente reservado para una cadena que especifique la localización física del objeto. El término *locator* se refiere a un URL bien definido.

Sintaxis

Esta sección describe la sintaxis para un *Uniform Resource Locator* (URL's). Básicamente direcciones de objetos que son recuperables usando protocolos ya existentes en Internet.

Describiremos sólo la sintaxis utilizada comúnmente por Mosaic, asociada a los protocolos más usados dentro de la red Internet. Para mayor información acerca de URL's, puede referenciar a la especificación que existe en el servidor FTP anónimo *ftp.w3.org* bajo el directorio */pub/www/doc/url7a-spec.txt*.

Un URL completo consiste de un nombre de esquema específico seguido de una cadena de caracteres cuyo formato está en función de ese esquema. Los componentes se describen a continuación.

El primer elemento es el nombre del esquema, separado del resto del objeto por 2 puntos. Aquellos que se refieran a protocolos de Internet tienen una sintaxis común para el resto del nombre del objeto. Esta sintaxis comienza con una doble diagonal "//" seguida por el nombre del host donde se localiza. A partir de aquí, la sintaxis depende del esquema usado, estos esquemas se refieren a protocolos de Internet.

En la siguiente sección describiremos los URL's comúnmente usados por Mosaic.

4.5.1 Tipos de URL's

Piense en el concepto de un URL como el nombre de un archivo extendido en red; no sólo puede "apuntar" a un archivo en algún directorio, sino que el mismo pueden existir en otra máquina dentro de la red. Los URL's pueden "apuntar" a consultas, documentos ordinarios, el resultado de un archie, etc.

File URL's

Imagine que un documento llamado *foobar.txt* se encuentra en un servidor FTP anónimo llamado *ftp.yoyodyne.com* bajo el directorio */pub/files*. El URL para este archivo es entonces:

file://ftp.yoyodyne.com/pub/files/foobar.txt

El directorio "pub" de este servidor es entonces:

file://ftp.yoyodyne.com/pub

Gopher URL's

Usted puede visitar a un servidor gopher en el servidor *gopher.yoyodyne.com*, usando el siguiente URL:

gopher://gopher.yoyodyne.com/

Algunos servidores Gopher pueden aceptar peticiones en un puerto de red poco usual, es por eso que si se conoce el número de puerto de la máquina, es posible invocarlo con su correspondiente número por ejemplo:

gopher://gopher.banzai.edu:1234/

Donde 1234 es el número de puerto.

News URL's

Estos URL's apuntan a servidores de noticias, el URL tiene que ser de la forma:

news:rec.gardeling

Actualmente, los clientes de la red como Mosaic no permiten especificar a un servidor de noticias como normalmente se esperaría, por ejemplo: *(news://newsyoyodyne.com./rec.gardeling)*; este estilo ya podría venir en camino pero mientras tanto se tendrá que especificar el servidor de noticias vía algún otro método. El método más común consiste en especificar la variable de ambiente NNTPSERVER a el nombre de su servidor de noticias antes de comenzar a utilizar Mosaic.

HTTP URL's

Los servidores HTTP son comúnmente utilizados para transferir documentos hipertexto, HTTP es un protocolo que se basa en el hecho de que la información que navega por la red puede ser comprimida en tales documentos directamente y por lo tanto el protocolo por sí mismo no tiene que soportar todas las características de navegación a través de la red tal como FTP y Gopher lo hacen.

Un archivo llamado "foobar.html" en el servidor HTTP llamado "www.yoyodyne.com" en el directorio "pub/files" corresponde al siguiente URL:

http://www.yoyodyne.com/pub/files/foobar.html

El puerto HTTP por default de red es el 80; si un servidor HTTP reside en otro puerto de red (digamos, puerto 1234 en *www.yoyodyne.com*), entonces el URL se convierte en:

http://www.yoyodyne.com:1234/pub/files/foobar.html

URL's Parciales

Una vez que se está visualizando algún documento localizado en algún lugar de la red (digamos, el documento *http://www.yoyodyne.com/pub/afile.html*), se puede entonces utilizar un URL *parcial* o *relativo* para apuntar a otro archivo dentro del mismo directorio en la misma máquina, siendo atendido por el mismo servidor. Por ejemplo, si otro archivo llamado *"otro.html"* existe en el mismo directorio, entonces ese archivo es un URL parcial válido.

Esta es una forma de construir conjuntos de documentos hipertexto. Si alguno de ellos está situado en un directorio común, se puede hacer referencia entre ellos mismos (por ejemplo, estar ligados) mediante sus nombres de archivos, un salto de un documento a otro puede realizarse a cualquier otro documento en el mismo directorio con tan solo utilizar el nombre del archivo del otro documento como un URL parcial. La información adicional (método de acceso, nombre del host, número de puerto, nombre de directorio etc.) se asume que es de la misma forma del URL utilizado inicialmente, por ejemplo HTTP.

Otros URL's

Muchos otros tipos de URL's son posibles, pero hemos cubierto los más comunes que se pueden construir manualmente. En la parte superior de cada

documento de Mosaic se encuentra un texto llamado: *Document URL*; si se observan los contenidos mientras se va navegando en Mosaic, entonces se podrá notar como los URL's son puestos juntos para varios y diferentes tipos de información.

4.6 El Lenguaje HTML (HyperText Markup Language)

El lenguaje HTML es utilizado para crear documentos hipertexto los cuales son portables de una plataforma a otra. Este lenguaje puede representar hipermedia, menús, resultado de consulta a bases de datos, y algunos otros formatos de información.

El lenguaje HTLM fue por primera vez especificado en 1991 por Tim Berners Lee, como una parte de la World Wide Web, inicialmente para facilitar la comunicación entre físicos investigadores de energía. Las especificaciones del lenguaje se fueron desarrollando para aplicarlas a la comunidad de la Web. En 1993 se escribió el primer documento SGML (Lenguaje Estándar de señalización generalizado por sus siglas en inglés: Standard Generalized Markup Languaje) en donde se describía a HTLM y se comenzó a ensamblar una serie de documentos para probar los *paginadores* de Web.

Para principios de 1994, el HTLM estaba obsoleto y no se reflejaba una práctica del mismo dentro de los mismos usuarios. Entonces se recopiló una serie de documentos que realizaban ligas de documentos hechos bajo NSCA Mosaic, y de esa manera se dieron una serie de especificaciones para poder desarrollar lo que sería la especificación 2.0 de HTLM. Ésta especificación se divide en cuatro categorías:

- El núcleo, el cual consiste en todas las características que toda aplicación de WWW debe soportar.
- Características opcionales, incluyendo imágenes y cambio de los tipos de letra.
- *Formas* para diseñar aplicaciones interactivas, y para la creación de aplicaciones sofisticadas para la Web.
- Elementos obsoletos que pudieran estar presentes como herramientas, pero que no deberían ser utilizados en los nuevos documentos, porque podrían tener un mal funcionamiento en el futuro.

Las *formas* en HTLM son la parte más innovadora de la especificación 2.0. Estas *formas* permiten al autor de un documento Hipertexto presentar al lector una forma interactiva que incluye: campos para poder introducir datos, presentaciones en pantalla para que el usuario pueda aceptar o no una opción mediante un click del *ratón* (botones) y menús desplegables, así como aplicaciones donde la entrada de la *forma* determina la naturaleza y el contenido del documento desplegando.

La especificación del HTML obedece a un gran modelo de desarrollo. Si una nueva característica es propuesta, entonces se implementa en algunos clientes y se prueba en algunas aplicaciones.

Si la demanda para la nueva característica es suficiente, entonces todos los paginadores se encargarán de implantarla, es cuando ésta llega a ser ampliamente utilizada por toda la comunidad. Durante este proceso, el diseño es revisado y posiblemente modificado.

Eventualmente, cuando ya existe suficiente experiencia y confiablidad con la nueva característica, llega a formar parte del conjunto estándar de propiedades de HTLM.

Escribiendo HTLM's

Este lenguaje de hipertexto es utilizado para definir un documento y como guía para su desplegado. El principio básico se basa en el uso de etiquetas para poder identificar el inicio y el final de algo que se quiera utilizar en HTLM.

Esta sección describe en forma breve cómo escribir documentos en HTML (por sus siglas en inglés HyperText Markup Language) que se utiliza en la World Wide Web y en Mosaic. Hay que aclarar que ésta no es una completa descripción del lenguaje HTML, pero esta guía cubre los requerimientos mínimos suficientes para poder realizar un documento completo en HTLM en el transcurso de una hora o dos.

La guía contiene las siguientes secciones:

- Principios básicos de HTML.
- Ejemplos para principiantes.
- Títulos y encabezados.
- Párrafos y formateo de los mismos.
- Efectos especiales básicos.
- Imágenes en línea.
- Ligas en Hipertexto.
- Listas numeradas y con viñetas.

Principios básicos de HTML

HTLM es un lenguaje basado en SGML diseñado para soportar formatos y la presentación de documentos de hipermedia "básicos".

Los documentos en HTML utilizan *etiquetas* para indicar un formato o información estructurada. Una *etiqueta* es simplemente un signo *menor que* (<) seguido por una directiva y cero o más parámetros seguidos por un símbolo *mayor que* (>). No debemos olvidar que este documento explica las directivas más usadas de HTML.

Un ejemplo de inicio

Para aquellas personas que prefieren aprender haciendo las cosas, aquí se muestra un ejemplo de un documento realizado en HTML:

```
<title> Ejemplo sencillo de un documento HTML.</title>
<h1> Un ejemplo simple </h1>
Este es un ejemplo simple de un documento en HTML. Este es el primer parrafo <p>
Este es el segundo parrafo. Esta es una palabra en <i> italica </i>. Esta es una palabra en
<b> bold </b>
Aqui tenemos una imagen en GIF: <img src="mi _imagen.gif">
<p>
Este es el tercer parrafo. Aqui tenemos una liga en hipertexto de la palabra
<a href="subdir/mylife.html"> foo</a>
a un documento llamado "subdir/myfile.html". <p>
<h2> Un encabezado de segundo nivel. </h2>

Aqui aparecera una seccion de texto que debera aparecer en una fuente mas
reducida y estrecha (como si estuviera una computadora listando o como si fuera un
verso de poesía): <p>
Esta es una lista con dos líneas separadas: <p>
<ul>
<li> primer tema va aqui
```

```
<li> segundo tema va aqui
</ul>
Este es el final de mi documento de ejemplo. <p>
<address>CARLOS SANCHEZ LARIOS</address>
```

Visualizado desde Mosaic (ver Figura 4.2), se puede observar que los tamaños de letra determinados son respetados, así como la imagen que se predeterminó, también se pueden ver los cambios en los tipos de letra, así como el título en la parte superior de la pantalla de Mosaic. La característica más importante es la facilidad que se puede tener para poner una liga de referencia hacia otro documento, ya sea mediante un texto o una imagen, de esta manera se podrán tener un cierto número de pantallas ligadas desde y hacia cualquier y por lo tanto, desde cualquier parte del mundo. Observando la Figura 4.2, note que cualquier documento en HTML de cualquier lugar de la red que puede accederse con Mosaic, puede fácilmente utilizarse; simplemente utilice la opción *Document Source* (fuente del documento) en el menú de Mosaic para llamar a una ventana que desplegará el código fuente del actual documento que se está visualizando.

Títulos y Encabezados

Todo documento en HTML deberá de tener una leyenda llamada *title* (título) que describa en algunas palabras el propósito del documento. Los títulos no son desplegados como parte del texto del documento, pero sí aparecen en la parte superior de la ventana de Mosaic y son utilizados como identificación de ese documento en ciertos contextos.

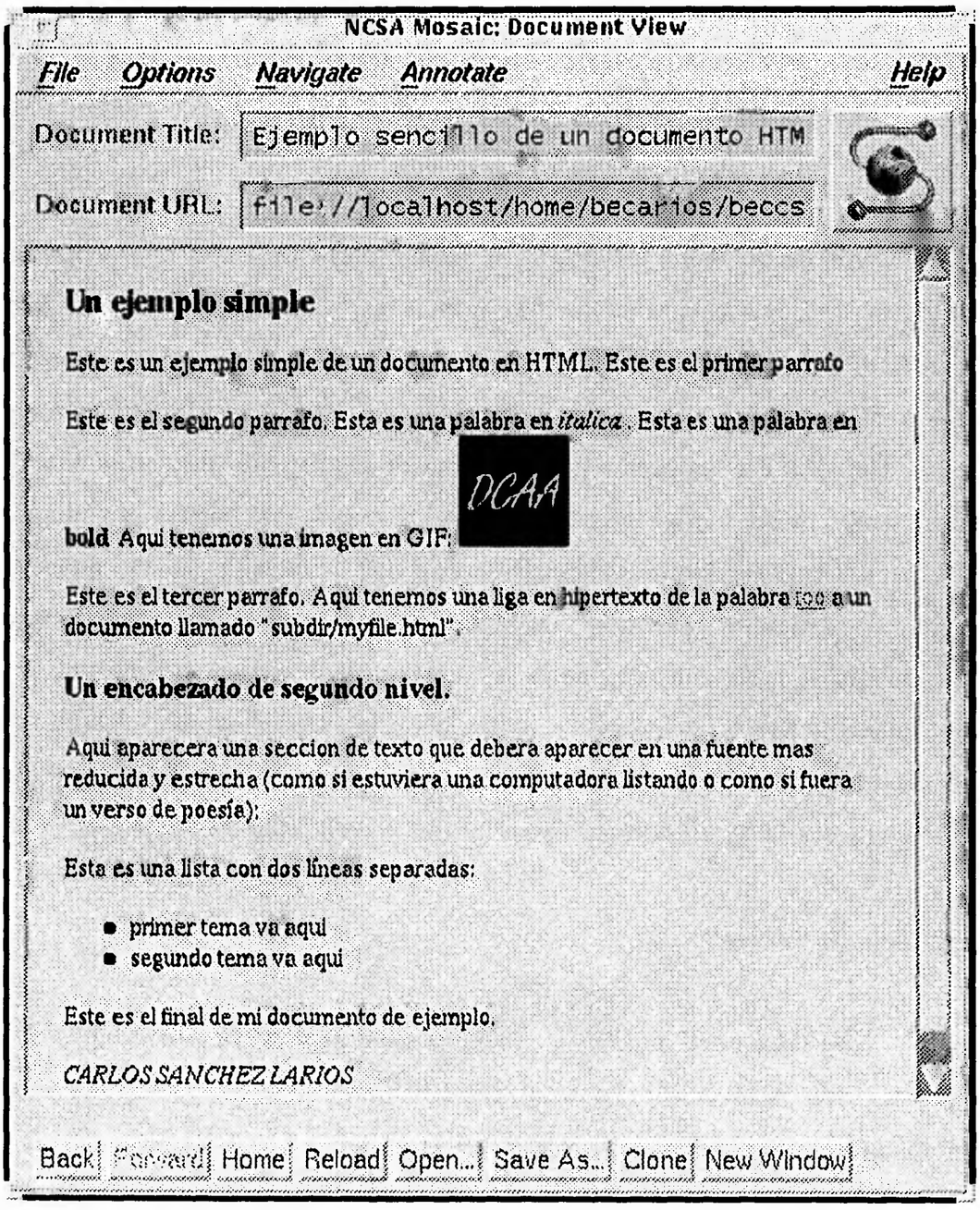


Figura 4.2 Ejemplo de un documento en HTML

El título generalmente va en la primera línea del documento. Aquí presentamos un ejemplo de un título:

<title> Este es el titulo de mi documento. </title>

Note que la directiva para la etiqueta de título se reconoce fácilmente: *title*. Notará también que existen dos etiquetas iguales, una que indica el principio de esa etiqueta, y otra final que difiere por la existencia de una diagonal (/).

Los encabezados son desplegados dentro del documento, generalmente utilizando fuentes más grandes y/o "negritas" en lugar del texto normal. Existen seis niveles de encabezado (numerados del 1 al 6) siendo el encabezado tipo 1 el más grande (generalmente sólo los niveles del 1 al 3 son utilizados con más frecuencia).

Aquí tenemos un ejemplo con encabezado nivel 1:

<h1>Este es un encabezado con nivel 1 </h1>

Aquí tenemos un ejemplo con encabezado nivel 2:

<h2>Este es un encabezado con nivel 2 </h2>

La mayoría de los documentos utilizan las mismas cinco o seis palabras tanto para el encabezado 1 como para el título del documento, por ejemplo:

<title> Una guia de principiantes para HTML </title>
<h1> Una guia de principiantes para HTML </h1>

Párrafos y formatos

Algo muy importante a considerar es que las líneas y espacios en blanco no tienen significado en el texto. Por lo tanto, si se termina un párrafo con una línea en

blanco sencilla por ejemplo, no es suficiente: **cada párrafo deberá terminar con una etiqueta**, que en HTML es <p>.

El siguiente ejemplo muestra el uso de la etiqueta de fin de párrafo:

```
Esta es mi primer sentencia. Esta es mi segunda sentencia. Esta es mi tercer sentencia.
Este es el final del parrafo. <p>
```

Caracteres especiales

Existen tres caracteres especiales dentro del código ASCII que no pueden ser utilizados como tales dentro de un documento de HTML. Estos caracteres son los símbolos "mayor que" (>), "menor que" (<), y el de ampersand (&).

¿Porqué es esto? Porque los dos primeros son utilizados para representar etiquetas, mientras que el ampersand es utilizado para realizar *mecanismos de escape* para éstos y otros caracteres, es decir, el ampersand va a permitir que los símbolos tomen su valor tal cual, por ejemplo:

- < es la secuencia de escape para <
- > es la secuencia de escape para >
- & es la secuencia de escape para &

Note que "secuencia de escape" sólo significa que dada una secuencia de caracteres (que representa el caracter sencillo en un documento en HTML), la conversión a su correspondiente caracter toma lugar por sí misma cuando el documento es desplegado al lector. Note también que existen secuencias de escape adicionales, por ejemplo:

- ñ es la secuencia de escape para una n con una tilde
- È es la secuencia de escape para una E mayúscula con un acento

Efectos especiales básicos

Palabras individuales o sentencias en párrafos pueden desplegarse en estilos de letra: **bold**, *itálica*, o estilos con ancho de línea. Correspondientemente deberá saber lo siguiente acerca de los tipos de letra:

- texto pone el texto en itálica (el resultado de este ejemplo será *texto*).
- texto pone el texto en bold (el resultado de este ejemplo será **texto**).
- <code>texto</code> pone el texto más ancho (el resultado de este ejemplo será texto).

Imágenes en línea

Un atributo más de Mosaic es que las imágenes (en formato X bitmap o GIF) pueden ser desplegadas dentro de un documento.

Por ejemplo, aquí se muestra como se introduciría una imagen llamada "imagen.gif" dentro de un documento Mosaic: <img src="imagen.gif">.

El uso de múltiples imágenes dentro de un documento hace que para el desplegado de cada una, se utilice tiempo de proceso y por lo tanto el despliege sea más lento. Utilizando una imagen múltiples veces causa que no haya rendimiento en comparación de sólo utilizar la misma una vez dentro del documento (note que la etiqueta img es una extensión que es solamente interpretada por Mosaic y no por otros paginadores de World Wide Web).

Ligas en Hipertexto

A partir de este momento podemos hablar de la existencia de Hipertexto en red. Existe una directiva principal para elaborar ligas y es conocido como *anchor* (el cual es un término común para un final de liga de hipertexto).

Un *anchor* es comúnmente usado para apuntar a algún lugar de un documento actual e incluso dentro de la red. Explicaremos su sintaxis:

- Se comienza abriendo una etiqueta con el siguiente símbolo: <a.
- Nombre del documento al que se hace referencia de la siguiente forma: *href="document.html"*, y por último el símbolo ">".
- Ponga el texto que deberá aparecer en el documento actual, el cual va a indicar una liga hipertexto (el texto aparecerá en otro color y subrayado para indicar que al hacer un click en él, seguiremos a la liga o documento referenciado).
- Terminar con la etiqueta:

Un ejemplo de referencia a hipertexto sería:

```
<a href="subdir/document.html">algun texto</a>
```

Lo cual causa que "algún texto" sea una liga al documento llamado "subdir/document.html".

También las imágenes pueden servir como elementos para formar ligas. Por ejemplo, en las siguientes sentencias, "imagen.gif" es una liga hacia "file.html".

```
<a href="http://machine.name/subdir/file.html"><img src="imagen.gif"></a>
```

Listas numeradas y con viñetas

Una *lista con viñetas* básicamente puede ser producida como sigue:

- Comenzar con una etiqueta de apertura *<ul>*
- Dar los temas uno a la vez, cada uno precedido por una etiqueta *
* (no hay etiquetas para cerrar del tipo para lista de temas)
- Terminar la etiqueta de apertura *
*

Para una lista *numerada,* haga lo mismo excepto que ahora utilice la directiva *ol* en lugar de la *ul*. Por ejemplo:

```
<ol>
<li> El primer tema va aquí
<li> El segunda tema va aquí
</ol>
```

Las listas pueden ser arbitrariamente anidadas, cualquier tema de la lista puede por si misma contener listas. También note que no son necesarios los separadores de párrafos (o cualquier otra cosa) al final de la lista; la etiqueta <li> nos sirve para ese fin. (Uno también puede tener un número de párrafos, cada uno entre ellos mismos conteniendo listas anidadas, en un solo tema de la lista, y así sucesivamente).

Un ejemplo de lista anidada se muestra a continuación:

```
<ul>
<li> Este tema incluye una lista anidada.
```

```
<ul>
    <li> Primer tema de una lista anidada

<li> Segundo tema de una lista anidada
</ul>
<li> El segundo tema va aqui
<ul>
<li>Unico tema de la segunda lista anidada.
</ul>
</ul>
```

Esto se despliega de la siguiente manera:

- Este tema incluye una lista ligada.
- Primer tema de una lista ligada
- Segundo tema de una lista ligada
- El segundo tema va aquí
- Unico tema de la segunda lista aninada.

El hipertexto tiene la habilidad de ligar tanto gráficas como texto (aunque otros se utilizan para realizar ligas a gráficas y formas), además existen opciones para poder desplegarlo en la parte inferior o en la parte superior de Mosaic utilizando sentencias como ALIGN=BOTTOM o ALIGN=TOP.

Es importante recordar que cuando se están utilizando imágenes, se debe tratar de reducir su tamaño si éstas son muy grandes u ocupan mucha memoria, ya que agilizan el proceso de transferencia del documento. En el caso de que se tenga una imagen muy grande a desplegar, se recomienda crear una versión más pequeña de la misma que a su vez sea una liga a la imagen original. Notará sin embargo, que la imagen reducida, la cual ha sido definida como una liga, será desplegada dentro

del documento, mientras que la imagen original podrá ser desplegada por un visualizador externo o en una ventana separada.

4.6.1 Formas

Antes del uso de las *formas* en HTLM, el flujo de información por la WWW era unidireccional. Las *formas* integran todos los medios para poder traer y manejar los datos que necesite el usuario. También abren una gran variedad de posibilidades para realizar transacciones de información tales como: solicitudes para obtener nuevos artículos de noticias o hasta poder ordenar una pizza.

Un gran número de opciones están disponibles para la construcción de *formas*, incluyendo cajas de texto, botones de verificación, y menús. Un programa en el servidor después interpreta los datos y los maneja de la forma más eficiente, ya sea regresando el resultado en hipertexto, transfiriéndola a un archivo, o notificando electrónicamente al restaurante de pizzas (en el caso que se haya ordenado una mediante este medio).

En esta sección describiremos los elementos más importantes que están involucrados para la realización de *formas* en HTML. Los detalles aquí descritos forman parte de la especificación 2.0 de HTML para la creación de *formas* en hipertexto.

Etiqueta FORM

La etiqueta FORM se utiliza para poder representar una *forma* dentro de un documento en HTML. Más de una pueden incluirse en un solo documento, pero no pueden ser anidadas entre sí. Su sintaxis es la siguiente:

<FORM ACTION="url"> ... </FORM>

Los atributos son los siguientes:

- ACTION. Es el URL del servidor el cual procesará el contenido de la *forma*.
- METHOD. Es el método en el servidor HTTP utilizado para procesar la *forma* de manera completa. Dependiendo de que método se utilice será como el servidor seleccionado trabaje; es altamente recomendable el uso de la opción POST. Las opciones válidas son:

  GET. Este es el método por default y causa que la forma completa sea agregada al URL como si fuera una consulta (query) normal.

  POST. Este método ocasiona que los contenidos de la forma sean enviados al servidor con todos los atributos de las etiquetas en lugar de ser consideradas como parte del URL.

Dentro de una *forma* se puede tener cualquier etiqueta de hipertexto excepto otro FORM. Específicamente, las etiquetas INPUT, SELECT y TEXTAREA son utilizados para especificar los elementos de interfaz dentro de la *forma*.

Etiqueta INPUT

Se utiliza para especificar un simple elemento de entrada dentro de una *forma*, se utiliza sin etiqueta de terminación. En Mosaic para X window existen varios tipos de etiquetas de entrada de datos por ejemplo, para texto, opciones de selección múltiples o booleanas. Los atributos de la etiqueta INPUT son las siguientes:

- TYPE:

  text. texto de entrada; este es el tipo por default.

  password. texto de entrada; los caracteres introducidos son representados por asteriscos.

  checkbox. Es la representación de un simple botón encendido o apagado

  radio. Es semejante al checkbox, solo con la variante de que puede haber sólo una selección.

  submit. Un botón de activación para enviar los datos de la forma actual a un servidor.

  reset. Un botón que causa que los múltiples elementos que pudiera llegar a contener una *forma* sean borrados o vuelvan a adoptar sus valores originales.

- NAME:

  Es el nombre simbólico (no el nombre desplegado) para el campo de entrada. Este deberá estar presente para todos los tipos *submit* y *reset*, y es utilizado cuando se envían en forma conjunta la cadena de consulta y los datos al servidor remoto cuando la forma ha sido aceptada.

- VALUE:

  Para un campo de entrada tipo texto o password, puede especificarse el contenido por default para el mismo. En el caso de los tipos *checkbox* y *radio* se le podrán asignar valores por default en los botones, en forma de encendido o apagado. Para las entradas de tipo *submit* y *reset* se les puede dar un valor que aparecerá desplegado en los cajones de texto de esas *formas*.

- CHECKED:

  Especifica que el botón de entrada se active por default (no se necesita valor).

- SIZE:

  Es el tamaño físico del campo de entrada en caracteres; éste es sólo apropiado para tipo texto y password. Si este elemento no está presente, el valor por default será de 20. La entrada de texto en una ventana puede especificarse con SIZE=width,height (ancho,alto); por ejemplo SIZE=60,12. Nota: El atributo SIZE no se deberá utilizar para especificar texto en forma de ventana una vez que haya decidido utilizar la etiqueta TEXTAREA.

- MAXLENGTH:

  Es el máximo valor de caracteres que pueden ser aceptados como entrada; sólo se utiliza para los campos de tipo password y texto (y solamente para campos de línea sencilla y no para TEXTAREA), si este atributo no se encuentra presente, el valor por default será ilimitado.

Etiqueta SELECT

Dentro de la *forma* <FORM> .... </FORM>, cualquier número de etiquetas SELECT son permitidas (incluyendo INPUT y TEXTAREA pero no más *formas* adicionales). INPUT y SELECT tienen tanto etiquetas de abrir y cerrar, dentro de SELECT sólo una secuencia de etiquetas de tipo OPTION (opción) cada uno seguido por una cantidad arbitraria de texto es permitida. Por ejemplo:

```
<SELECT NAME="menu">
<OPTION> Primera opción.
<OPTION> Segunda opción.
</SELECT>
```

Los atributos de SELECT son los siguientes:

- NAME:

  Es el nombre simbólico para este elemento. Siempre deberá estar presente, y es utilizado cuando se coloca junto con la cadena de petición para la *forma*.

- SIZE:

  Si SIZE es 1 o si no se coloca, por default SELECT estará representado como un menú de opciones Motif. Si SIZE es de tamaño de 2 o más, SELECT estará representado como una barra de corrimiento.

- MULTIPLE:

  Especifica que la opción SELECT permitirá múltiples selecciones. La presencia de esta etiqueta forza al menú a ser representado como una barra de corrimiento ignorando el valor de SIZE.

  El atributo opcional de OPTION puede ser el siguiente:

  SELECTED.

  Especifica que esta opción es seleccionada por default. Si SELECT permite múltiples selecciones, entonces las opciones múltiples pueden ser especificadas como SELECTED.

Etiqueta TEXTAREA

La etiqueta TEXTAREA se utiliza para colocar texto en una forma de ventana. Los atributos de esta etiqueta son los siguientes:

- NAME. Es el nombre simbólico para el campo de entrada.
- ROWS. Es el número de filas que contendrá la ventana.
- COLS. Es el número de columnas que contendrá la ventana.

Las etiquetas de tipo TEXTAREA tienen barras de corrimiento; cualquier cantidad de texto puede ser capturada en ellas.

TEXTAREA requiere etiquetas tanto de abrir como de cerrar, una etiqueta sin contenidos por default se declararía así:

<TEXTAREA NAME="foo" ROWS=4 COLS=40></TEXTAREA>

Una etiqueta TEXTAREA con contenidos por default se declararía así:

<TEXTAREA NAME="foo" ROWS=4 COLS=40>
Los contenidos por default van aquí
</TEXTAREA>

Dichos contenidos por default deberán ser caracteres ASCII.

4.6.2 Los CGI's (Common Gateway Interface)

Generar *formas* en HTML es sólo la mitad del camino. El reto ahora es procesar los datos capturados desde las mismas y enviarlos a un programa ejecutable. Todos los programas que utilizan datos provenientes de un forma en HTML se les conoce como CGI (Common Gateway Interface). Cualquier lenguaje capaz de generar archivos ejecutables es apropiado para crear alguno. El lenguaje más popular es C.

En un servidor UNIX, el programa CGI recibirá los datos de la *forma* vía entrada estándar (stdin: por sus siglas en inglés "standard input"), si es que se utiliza el atributo METHOD=POST. Por la misma vía, la salida producida por el programa CGI es enviada a una salida estándar (stdout: por sus siglas en inglés "standard output"), la cual tomará el examinador.

¿ Para que se utiliza un CGI ?

Los CGI's son programas que manejan solicitudes de información y devuelven resultados. Con un CGI, el servidor puede desplegar información que no puede leer el cliente directamente (como una base de datos de SQL), y actuar como un gateway entre los dos para producir algo que los clientes puedan utilizar.

Los gateways se utilizan para una variedad de propósitos, el más común está en el manejo de *formas* para servidores HTTP. Los siguientes son algunos ejemplos de usos de CGI's:

- Convertir páginas de algún manual en HTML y enviar el resultado al cliente.
- Generar una interfaz con WAIS y bases de datos archie, convirtiendo los resultados en HTML y mandándolos al cliente.
- Permite al usuario la retroalimentación con el servidor a través de formas HTML acompañadas con un CGI.

¿Que son los CGI's exactamente?

Son programas ejecutables o scripts que pueden correr por sí mismos. Hacen posible que programas externos puedan comunicarse con diferentes servidores y puedan intercambiar información entre ellos.

Los CGI's, pueden ser escritos en cualquier lenguaje que genere un archivo ejecutable. Entre los lenguajes más populares se incluyen:

- C/C++
- PERL
- Bourne Shell
- C Shell

4.7 Especificación de Recursos

Mosaic para X Window es como cualquier otra aplicación X. Los valores asociados con los diferentes recursos controlan diferentes aspectos y comportamiento de cualquier aplicación. El usuario puede alterar los valores por default de una aplicación X especificando directamente otro valor en un *archivo de recursos*. Todos los programas X utilizan el mismo mecanismo para determinar los valores y la sintaxis que van a tener esos recursos.

Archivos de Recursos

Cuando Mosaic comienza a correr, el programa busca en 2 archivos para sus recursos especificados. El primero lo toma por default. Todas las aplicaciones X almacenan sus archivos de recursos por default en el mismo directorio, en el caso del Sistema Operativo Unix Solaris 2.3 se encuentran bajo el directorio */usr/openwin/app-defaults*. El nombre del archivo de recursos es el *class name* o nombre de la aplicación; Mosaic tiene un *class name* llamado *Mosaic*, en ese archivo se pueden ver los atributos por default de la aplicación. Si uno es el

administrador del sistema, debe verificar que este archivo esté instalado apropiadamente. Pero en el caso de ser un usuario, lo que necesita saber es donde está localizado este archivo y examinar los recursos por default que contiene.

Si ya se ha instalado Mosaic, también se necesitará transferir un archivo para recursos por default. Este archivo se encuentra en el siguiente URL del servidor FTP anónimo del NCSA :

ftp://ftp.ncsa.uiuc.edu/Web/Mosaic/Unix/binaries/app-defaults

Existen tres archivos en este directorio: uno para computadoras con monitores a color, otro para aquellas con adaptador SGI y el último para monitores monocromáticos. Es necesario verificar los atributos por default para esta opción. Si Mosaic encuentra un archivo de recursos por default, tratará de leer las nuevas especificaciones del mismo. Sin embargo, si el archivo no existe, Mosaic trabajará con los recursos por default utilizando sus fuentes y colores originales.

La manera más segura de instalar un archivo de recursos, es transferir el original y posteriormente modificarlo de acuerdo a nuestras necesidades para no perder las propiedades originales de Mosaic.

Después que Mosaic verifica la existencia del archivo de recursos bajo el directorio */usr/openwin/lib/app-defaults*, toma los valores que son referentes al programa y que fueron especificados por el usuario. Si se usa **xrdb** para cargar recursos en un servidor X cuando establecemos una sesión, Mosaic lee los recursos del servidor. De lo contrario, buscará un archivo llamado *.Xdefaults* en el directorio de trabajo por default (home directory). Si existe, leerá los recursos que se encuentren en él.

El usuario deberá usar el comando **xrdb** de Unix para cargar los recursos de Mosaic. Cuando lo corra, debe especificar el nombre del archivo que comúnmente se llama *.Xdefaults* o *.Xresources*.

Si no está seguro de cómo el sistema maneja los recursos, el usuario puede averiguarlo en su directorio de trabajo y buscar el archivo *.Xdefaults*. Para verificar si **xrdb** está siendo usado, en el archivo *.xinitrc* debe aparecer una línea de la forma:

xrdb nombre_de_archivo

o

xrdb -load nombre_de_archivo

Si se encuentra alguna de estas líneas, se podrá poner cualquier especificación de recursos en ese archivo (nombre_de_archivo). Pero, si xrdb no está en uso, se deberá verificar si existe el archivo llamado *.Xdefaults*. De ser así, se podrán agregar todas las especificaciones de los recursos a este archivo y actualizar los cambios con el siguiente comando:

% xrdb .Xdefaults

Sintaxis de la especificación de recursos

Una aplicación X identifica sus recursos con su nombre de clase (class name). En el caso de Mosaic, el nombre de clase es *Mosaic,* las especificaciones de los recursos para el programa son todas de la forma:

*Mosaic*Nombre_del_recurso: valor*

Hay que asegurarse de incluir los dos puntos después del nombre del recurso en la especificación. Si se omiten, Mosaic silenciosamente ignora la especificación y no despliega mensajes de error para las especificaciones de recursos incorrectas. Cada especificación debe estar en diferente línea, si el último caracter es una diagonal invertida (\), la definición de ese recurso continuará en la siguiente. Se puede incluir un comentario en la definición de un recurso o al principio de una línea mediante el uso de un signo de admiración (!). A continuación presentaremos un ejemplo de cómo se podrían definir en el archivo *.Xdefaults*:

! Recursos de Mosaic
*Mosaic*homeDocument: /usr/local/home/pilar/mosaic/mitesis.html*
Mosaic verticalScrollOnRight: False*
*Mosaic*confirmExit: False*

Nótese que las letras mayúsculas en el momento de la definición de los recursos son importantes. Si se olvida enunciar algún parámetro o alguna palabra que comience con mayúscula, Mosaic por supuesto la ignorará. Cada recurso tendrá su tipo de dato (por ejemplo: cadena, entero, booleano, etc.).

Algunos recursos de Mosaic tienen banderas para comandos. Por ejemplo, el recurso *homeDocument* puede ser sustituido por *-home*, se pueden incluir recursos de Mosaic en la línea de comandos mediante la opción *-xrm*. Esta opción toma la siguiente forma:

*% Mosaic -xrm "Mosaic*Nombre_del_recurso: valor" &*

Es posible especificar sólo un valor para el recurso con la opción -xrm, entonces si se quieren colocar múltiples recursos, se necesitará utilizar la opción -

xrm para cada uno. La especificación de recursos en la línea de comandos provee una manera conveniente para probar las especificaciones de estos, antes de agregarlos en forma definitiva al archivo de recursos. Por ejemplo, se podría utilizar el siguiente comando para observar si se consigue lo deseado o no:

*% Mosaic -xrm "Mosaic*anchorColor: LimeGreen" &*

Es importante señalar que Mosaic lee los archivos de recursos únicamente cuando se inicia el sistema. Si se modifica un archivo de recursos en el momento que Mosaic se esta ejecutando, las nuevas especificaciones no toman efecto sino hasta que se reinicialice Mosaic. Si el xrdb carga los recursos en el servidor, también habrá necesidad de volver a cargar los recursos cuando se modifique el archivo *Xdefauls* mediante:

% xrdb .Xdefaults

Ejemplo del contenido del archivo .Xdefaults:

```
Mosaic*XmLabel*fontList:                    -*-helvetica-bold-r-normal-*-14-*-*-*-*-*-iso8859-1
Mosaic*XmLabelGadget*fontList:   -*-helvetica-bold-r-normal-*-14-*-*-*-*-*-iso8859-1
Mosaic*XmScale*fontList:                    -*-helvetica-bold-r-normal-*-14-*-*-*-*-*-iso8859-1
Mosaic*XmBulletinBoard*labelFontList:       -*-helvetica-bold-r-normal-*-14-*-*-*-*-*-iso8859-1
Mosaic*optionmenu.XmLabelGadget*fontList:          -*-helvetica-bold-r-normal-*-14-*-*-*-*-*-iso8859-1
Mosaic*XmPushButton*fontList:               -*-helvetica-medium-r-normal-*-14-*-iso8859-1
Mosaic*XmPushButtonGadget*fontList:         -*-helvetica-medium-r-normal-*-14-*-iso8859-1
Mosaic*XmToggleButton*fontList:  -*-helvetica-medium-r-normal-*-14-*-iso8859-1
Mosaic*XmToggleButtonGadget*fontList:    -*-helvetica-medium-r-normal-*-14-*-iso8859-1
Mosaic*optionmenu*fontList:                 -*-helvetica-medium-r-normal-*-14-*-iso8859-1
Mosaic*XmIconGadget*fontList:               -*-helvetica-medium-r-normal-*-14-*-iso8859-1
Mosaic*XmBulletinBoard*buttonFontList: -*-helvetica-medium-r-normal-*-14-*-iso8859-1
Mosaic*menubar*fontList:                    -*-helvetica-bold-o-normal-*-14-*-iso8859-1
Mosaic*XmMenuShell*XmPushButton*fontList:  -*-helvetica-bold-o-normal-*-14-*-iso8859-1
Mosaic*XmMenuShell*XmLabelGadget*fontList: -*-helvetica-bold-o-normal-*-14-*-iso8859-1
Mosaic*XmMenuShell*XmPushButtonGadget*fontList: -*-helvetica-bold-o-normal-*-14-*-iso8859-1
Mosaic*XmMenuShell*XmCascadeButton*fontList: -*-helvetica-bold-o-normal-*-14-*-iso8859-1
Mosaic*XmMenuShell*XmCascadeButtonGadget*fontList: -*-helvetica-bold-o-normal-*-14-*-iso8859-1
Mosaic*XmCascadeButton*fontList: -*-helvetica-bold-o-normal-*-14-*-iso8859-1
Mosaic*XmCascadeButtonGadget*fontList: -*-helvetica-bold-o-normal-*-14-*-iso8859-1
```

```
Mosaic*XmMenuShell*XmToggleButton*fontList: -*-helvetica-bold-o-normal-*-14-*-iso8859-1
Mosaic*XmMenuShell*XmToggleButtonGadget*fontList: -*-helvetica-bold-o-normal-*-14-*-iso8859-1
Mosaic*pulldownmenu*fontList:    -*-helvetica-bold-o-normal-*-14-*-iso8859-1
Mosaic*XmList*fontList:    -*-helvetica-medium-r-normal-*-14-*-iso8859-1
Mosaic*XmText.fontList:    -*-lucidatypewriter-medium-r-normal-*-14-*-iso8859-1
Mosaic*XmTextField.fontList: -*-lucidatypewriter-medium-r-normal-*-14-*-iso8859-1
Mosaic*TitleFont: -adobe-times-bold-r-normal-*-24-*-*-*-*-*-iso8859-1
Mosaic*Font: -adobe-times-medium-r-normal-*-17-*-*-*-*-*-iso8859-1
Mosaic*ItalicFont: -adobe-times-medium-i-normal-*-17-*-*-*-*-*-iso8859-1
Mosaic*BoldFont: -adobe-times-bold-r-normal-*-17-*-*-*-*-*-iso8859-1
Mosaic*FixedFont: -adobe-courier-medium-r-normal-*-17-*-*-*-*-*-iso8859-1
Mosaic*FixedboldFont: -adobe-courier-bold-r-normal-*-17-*-*-*-*-*-iso8859-1
Mosaic*FixeditalicFont: -adobe-courier-medium-o-normal-*-17-*-*-*-*-*-iso8859-1
Mosaic*Header1Font: -adobe-times-bold-r-normal-*-24-*-*-*-*-*-iso8859-1
Mosaic*Header2Font: -adobe-times-bold-r-normal-*-18-*-*-*-*-*-iso8859-1
Mosaic*Header3Font: -adobe-times-bold-r-normal-*-17-*-*-*-*-*-iso8859-1
Mosaic*Header4Font: -adobe-times-bold-r-normal-*-14-*-*-*-*-*-iso8859-1
Mosaic*Header5Font: -adobe-times-bold-r-normal-*-12-*-*-*-*-*-iso8859-1
Mosaic*Header6Font: -adobe-times-bold-r-normal-*-10-*-*-*-*-*-iso8859-1
Mosaic*AddressFont: -adobe-times-medium-i-normal-*-17-*-*-*-*-*-iso8859-1
Mosaic*PlainFont: -adobe-courier-medium-r-normal-*-14-*-*-*-*-*-iso8859-1
Mosaic*PlainboldFont: -adobe-courier-bold-r-normal-*-14-*-*-*-*-*-iso8859-1
Mosaic*PlainitalicFont: -adobe-courier-medium-o-normal-*-14-*-*-*-*-*-iso8859-1
Mosaic*ListingFont: -adobe-courier-medium-r-normal-*-12-*-*-*-*-*-iso8859-1
Mosaic*supSubFont: -adobe-times-medium-r-normal-*-10-*-*-*-*-*-iso8859-1
```

Capítulo 5

Propuesta del proyecto

En el capítulo anterior mencionamos algunas características del software a utilizar en la realización de nuestro proyecto.

Hasta este momento hemos discutido algunos conceptos importantes así como los recursos de hardware con los que cuenta la DCAA. Por lo tanto, es hora de comenzar en sí con el diseño de nuestra propuesta, describir en qué va a consistir y para qué vamos a utilizar un Kiosco de Información.

5.1 Sistema de Información integrado en un Kiosco

Como mencionamos en la Introducción, la inquietud sobre la realización de un proyecto tan ambicioso surge en lo que antes era *Departamento de Apoyo a Sistemas Unisys* (ahora Coordinación Técnica) gracias al Ing. Fernando Baz García. La propuesta del sistema comienza desde el año de 1994 a raíz de conjuntar herramientas disponibles en ese entonces e iniciar una etapa en la que el objetivo principal es la solución de problemas primero dentro de la Universidad y porqué no; llegar inclusive al Gobierno Federal e Iniciativa privada.

La idea original sobre el proyecto, es la instalación de terminales (similares a la red de cajeros automáticos de los bancos) dentro del campus universitario, conectadas a la RedUnam (aprovechando esta infraestructura) y ofrecer un servicio

de caracter informativo a la comunidad universitaria y público en general. Sin embargo, esa idea ha ido madurando conforme se desarrolló el sistema, ahora aplicándola a la solución de problemas reales tal como lo mostramos en el capítulo 6.

Las terminales consisten básicamente de los siguientes elementos:

- Una estación de trabajo (en este caso una SparcStation 5, aunque no necesariamente).
- Una pantalla *Touchscreen*.
- Teclado.
- Un mueble que contiene los elementos anteriores y da el aspecto de Kiosco.

El usuario puede llegar al kiosco, pedir información acerca del campus, institutos y otras dependencias de la UNAM a través de un ambiente gráfico amigable (Mosaic) utilizando la pantalla *touchscreen*. El sistema lo lleva prácticamente de la mano, el usuario nunca se entera que la información que él ve, se puede encontrar en alguna otra máquina dentro de la red. Características con más detalle se presentan en la siguiente sección.

El tener las terminales conectadas a la red presenta las ventajas de poder compartir los recursos entre las diferentes máquinas que la componen. Recursos tales como archivos, bases de datos y dispositivos periféricos.

¿ Porqué utilizar una estación de trabajo ?

Es importante señalar que aunque el proyecto se desarrolló en una estación de trabajo SparcStation 5 de Sun Microsystems, el sistema podría implantarse en

cualquiera que corra el sistema de ventanas X-Window y por lo tanto, ejecute Mosaic.

En nuestro caso, el equipo disponible en ese momento fue la *SparcStation* pero bien podría migrarse a otra estación de trabajo, lo que presentamos aquí es el **prototipo del Kiosco**.

Se utiliza una estación de trabajo debido a la capacidad de procesamiento y almacenamiento masivo de imágenes, video y sonido. Si pensamos en una computadora personal, estaríamos muy limitados por la capacidad de procesamiento; la velocidad que se requiere para el manejo de los diversos tipos de datos (imagen y video a color, sonido y texto principalmente) es enorme, por lo que hasta ahora, sólo las estaciones de trabajo pueden soportar la cantidad de operaciones que se requieren manipular en sistemas como el nuestro.

Por otra parte, Mosaic únicamente está disponible para 3 tipos de plataformas (como lo mencionamos en el capítulo anterior), éstas son: Windows para PC's, X-Window para estaciones de trabajo y Mac con su propio sistema operativo. Por lo tanto, debemos elegir algún equipo con cualquiera de los tres ambientes gráficos anteriores. Particularmente, utilizamos la estación de trabajo *SparcStation 5* porque es un equipo con el que cuenta la DCAA, tenemos la ventaja de que ya existía y por lo tanto, en él se desarrolló el sistema.

5.2 Características del equipo

En esta sección describiremos las características principales del equipo y herramientas de software que utilizamos en el proyecto.

Características de la SparcStation 5

La SparcStation 5 es un modelo de estación de trabajo creado por Sun Microsystems, entre sus características se encuentran las siguientes:

- 32 MB de memoria base, tiene un total de ocho slots disponibles, un SIMM de 32 MB en un slot y deja siete vacantes.
- 1 GB de capacidad de disco interno.
- procesador microSPARC-II de 110 MHz.
- 3 slots Sbus de entrada/salida de 32 bits.
- entrada para par trenzado y AUI.
- monitor de 17 pulgadas color, con tamaño de imagen de aproximadamente 328x242 mm. Con una definición horizontal de 1280 puntos y vertical de 1024 puntos.
- Sistema Operativo solaris 2.3 edición II o solaris 1.1.1 versión B o posteriores como software de sistema .
- para la conectividad en red soporta Token Ring, interfaz serie, FDDI, ATM, Ethernet e ISDN.
- manejo de distintos medios en red, como video (SunVideo) y cámara de video (SunCamera).
- dos puertos seriales.
- un puerto paralelo.
- un puerto SCSI.
- soporta cuatro dispositivos internos, dos discos, una unidad de disco flexible y un drive de CD-ROM.

Integrando todos estos elementos, el equipo se ve como el de la Figura 5.1.

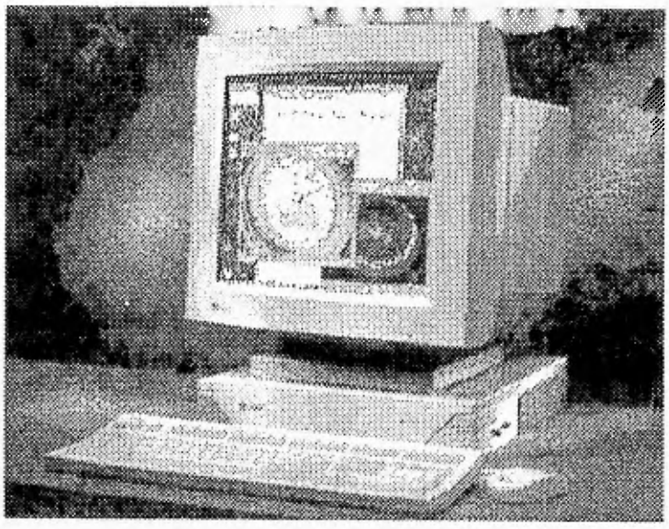


Figura 5.1 SparcStation 5

Como se puede apreciar en la Figura 5.1, se trata de un equipo relativamente pequeño en lo que respecta al CPU, existe una familia de este tipo de estaciones de trabajo (SparcStation 5) que han sido diseñadas especialmente para el desarrollo de aplicaciones Multimedia. El equipo que utilizamos en la realización del Kiosco, es el básico.

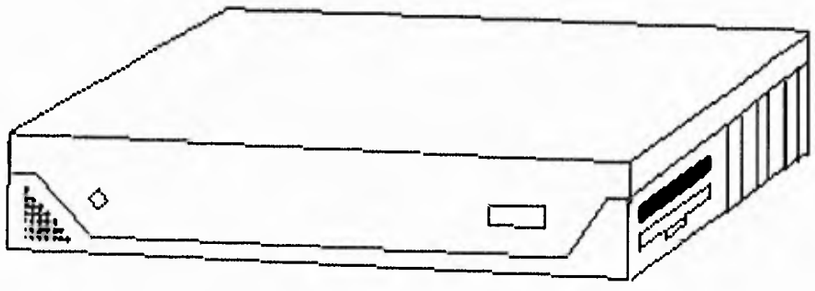


Figura 5.2 CPU de la SparcStation 5

Entre sus aplicaciones se encuentran las siguientes:

- CAD 2-D y 3-D.
- Diseño arquitectónico.
- Diseño Gráfico.
- Imágenes médicas entre otras.

Un equipo adicional que viene incluido con la SparcStation 5 es una pequeña cámara de video, conocida como *SunCamera*. Es una cámara a color para equipos con sistemas de video tales como los de Sun Microsystems Computer Corporation (SMCC) para sistemas SparcStation. Entre las características de hardware, tiene un dispositivo de lente fijo acoplado por carga (CCD), alimentado por 12 volts DC con salida de adaptador AC, la cámara soporta una conexión a un puerto de entrada de video. La lente tiene una longitud focal de 5.6 mm y ángulos visualizadores de 42 grados horizontal y 32 vertical. Tiene una contraventana electrónica automática con velocidad de 1/60 a 1/15000 segundos, con balance de imagen, sincronización interna, balance de blancos y ajuste automático de velocidades para obtener calidad en la imagen. Opera en niveles bajos de luz con lo que se puede obtener una imagen efectiva. La *SunCamera* en conjunto con otros productos como la tarjeta *SunVideo* provee aplicaciones multimedia como video conferencias, video correo o creación de documentos incorporando video clips para presentaciones. La Figura 5.3 muestra la *SunCamera*.

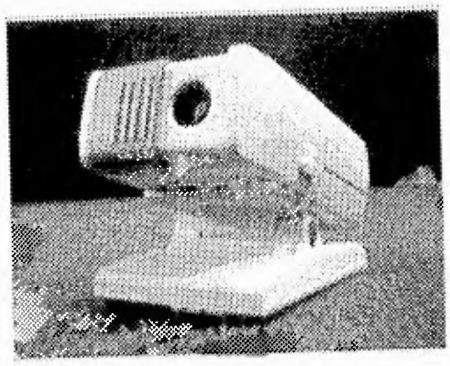


Figura 5.3 SunCamera

Entre las herramientas que utilizamos en el desarrollo del Kiosco, incluimos aquellas que nos permiten realizar multimedia con Mosaic y que pueden correr en la estación de trabajo donde realizamos el proyecto (SparcStation 5). Por ejemplo, aquellas aplicables en la grabación de videos, imágenes y sonido. También incluimos una pequeña explicación de su uso, para posteriormente generar los archivos necesarios y hacer la llamada a ellos utilizando hipertexto en lenguaje HTML (sección 4.6). Estas herramientas pueden complementarse con otras pertenecientes a otros equipos de cómputo incluyendo computadoras personales.

Los programas llamados *Snapshot* y *Audiotool* forman parte del software que acompaña al sistema operativo de la estación de trabajo SparcStation 5 (Solaris en sus versiones 2.3 y 2.4). Por otra parte, *SunVideo* es un producto comercial (de Sun Microsystems) que se vende por separado e incluye: software, la tarjeta de video y una pequeña cámara (de la cual ya hablamos) donde podemos grabar secuencias de video.

Pero también existen en la Red Internet, herramientas de software que forman parte del dominio público y nos permiten manipular imágenes. Tal es el caso de *XV* que incluimos en esta sección.

Es interesante saber que el software para realizar multimedia no es exclusivo de una máquina, gracias a la variedad de formatos de archivos que existen hoy en

día, podemos exportarlos de una plataforma de hardware a otra; realizar cambios en los documentos originales y después devolverlos a sus dispositivos de almacenamiento iniciales o bien, conservarlos en otra parte, y eso es algo que realizamos constantemente sobre todo en el manejo de imágenes.

El subsistema de video de SUN (SunVideo)

Cada vez es más común encontrar recursos de video en el ambiente de casi todas las computadoras de hoy en día. Video conferencias, imágenes sintetizadas, video en vivo, animación, así como otras opciones que han sido creadas con el fin de cubrir las necesidades del usuario.

Parte del cambio en el desarrollo de nuevas tecnologías de video es la necesidad de comprimir imágenes. Trabajarlas sin comprimir, resulta poco práctico principalmente por la demanda de procesadores rápidos que éstas requieren, gran cantidad de espacio en disco para su almacenamiento y su difícil manejo en ambiente de red. Como ejemplo, podemos mencionar que una imagen requiere de más de 900 kbytes de espacio de memoria para una imagen de 24 bits de formato RGB de 640 x 480 pixeles de resolución.

La otra parte del cambio lo constituye la velocidad de compresión y descompresión de las imágenes. La necesidad de estas tareas en tiempo real es un requerimiento muy importante para aplicaciones en video conferencias. El subsistema de video de SUN o *SunVideo* logra cubrir estos requerimientos gracias a su tarjeta de video de compresión de imágenes, la cual ofrece múltiples estándares de compresión (ver Figura 5.4).

SunVideo es un sistema de captura y compresión de imágenes en tiempo real. La tarjeta y su software capturan, digitalizan y comprimen señales de video de diferentes dispositivos como cámaras y discos. Las imágenes comprimidas pueden

ser almacenadas en disco, transmitidas a través de la red o ser descomprimidas para desplegarlas en la pantalla de una estación de trabajo.

Este subsistema es un elemento esencial para la implantación de video conferencias y aplicaciones en multimedia. La tarjeta es completamente digital, por lo que los datos pueden ser recuperados y procesados por el CPU a niveles mayores de las 30 imágenes por segundo.

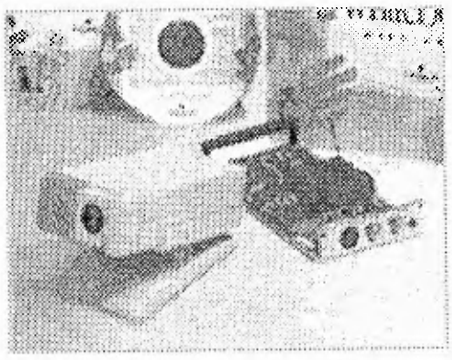


Figura 5.4 Sistema SunVideo

SunVideo está diseñado para trabajar con aplicaciones que utilizan las funciones de la plataforma de bibliotecas de imágenes conocida como XIL. Por sí misma, XIL nos ofrece funciones para el procesamiento de imágenes, así como software para su compresión y descompresión, también nos brinda facilidades para acelerar la ejecución de las operaciones más utilizadas.

La tarjeta de video de SUN soporta las siguientes técnicas de compresión:

- CellB
- JPEG
- MPEG-1
- UYVY

SunVideo se encargará de *capturar el video* y de comprimirlo; no se encargará de descomprimirlo. Las aplicaciones del software de XIL contienen funciones que lo descomprimen en el host. Por lo tanto, los niveles de descompresión dependerán del rendimiento del CPU.

Cabe señalar que el subsistema de video no ofrece soporte directo para audio. Las aplicaciones que lo requieren, tales como video conferencias, pueden utilizar las utilerías para manipular el */dev/audio*.

La siguiente tabla resume las capacidades de rendimiento de representación de imágenes dependiendo de la técnica de compresión (las unidades están en cantidad de imágenes por segundo):

Método de compresión	Resolución de la imagen 320 x 240 pixeles	Resolución de la imagen 340 x 288 pixeles
CellB	30	25
JPEG	30	25
MPEG-1 (con niveles de bit variables)	30	25
MPEG-1 (con niveles de bit constantes)	18	17
UVYV	30	25

Tabla 3 Métodos de compresión y resolución de imagen

Usos Comunes

Algunos ejemplos en los cuales el subsistema de video de SUN puede ser utilizado son:

- Video conferencia.
- Aplicaciones en Multimedia, utilizando la captura y compresión de video, entre las cuales destacan:

    1. Entrenamiento automático
    2. Presentaciones para productos y comerciales de televisión
    3. **Kioscos de información**
    4. Publicaciones electrónicas

- Visualización de programas de televisión en una estación de trabajo.
- Captura de imágenes de video para algún software.
- Procesamiento de imágenes.
- Como dispositivo de captura de imágenes fijas.

Video Conferencia

Para una video conferencia, se necesita que en las dos localidades se tengan dos tarjetas para capturar y comprimir las imágenes de video alternativamente. Estas aplicaciones envían y reciben las imágenes comprimidas y las descomprimen en tiempo real, a su vez pueden sincronizar las señales de audio y video (ver Figura 5.4).

Aplicaciones en Multimedia

Dependiendo de las necesidades de la aplicación, se puede utilizar video previamente almacenado o en tiempo real.

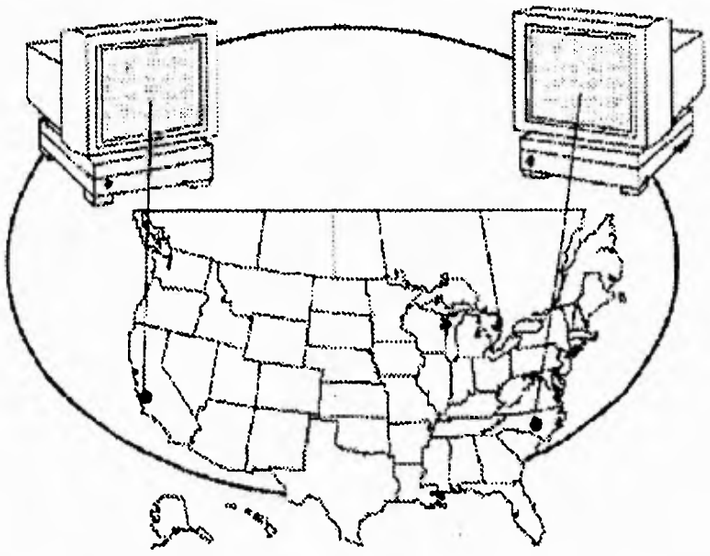


Figura 5.4 Ejemplo de una video conferencia

El video comprimido puede ser almacenado en un archivo de tal manera que pueda posteriormente ser editado e integrado en una presentación de multimedia. Estas imágenes también se pueden integrar a una **base de datos** para poder presentar la información en una forma más efectiva. Aquí presentamos algunos ejemplos:

- Fotografías de profesores para un sistema de inscripciones.
- Fotografías de objetos asegurables para una compañía de seguros.
- Autopartes de distribuidores autorizados de compañías de automóviles.
- Imágenes de hoteles y sitios de interés para los turistas.
- Imágenes de contratos y/o documentos legales para confirmación de trámites afines.

Digitalización de video

SunVideo incluye un software dentro del cual se encuentra un programa llamado *SunVideo*, al ejecutarlo muestra una ventana en la pantalla.

Desde la ventana, se selecciona el puerto de entrada y el tipo de compresión que se desea realizar y escoger si sólo se desea desplegar la imagen o si se quiere salvar en un archivo.

Ahora mencionaremos los pasos para poder digitalizar un video:

1. Conectar un dispositivo de video a alguno de los puertos de la SUN.
2. Dirigirse a un directorio en donde se tenga permiso de escritura por ejemplo:

   % cd /home/archivos_video
3. Teclear el siguiente comando:

   % /opt/SUNWits/Graphics-sw/xil/examples/test/SunVideo

Deberán aparecer las pantallas como se muestran en las Figuras 5.5a y 5.5b:

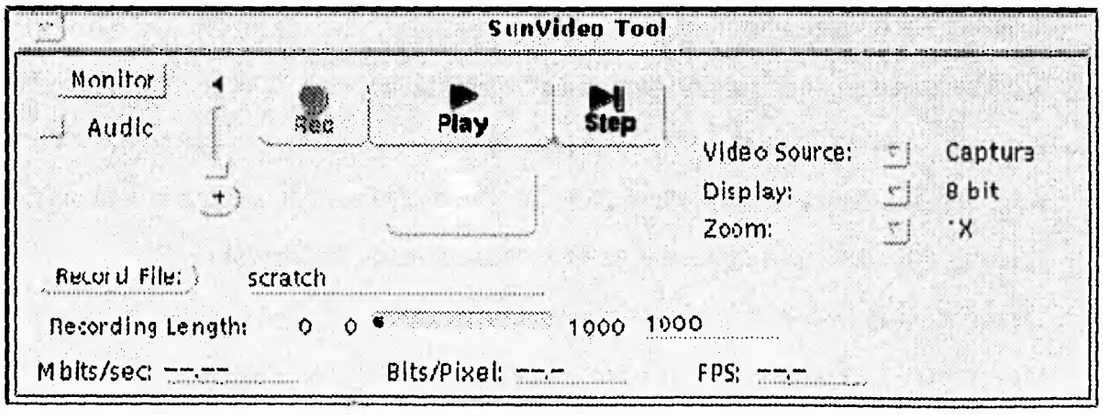


Figura 5.5a Ventana de SunVideo

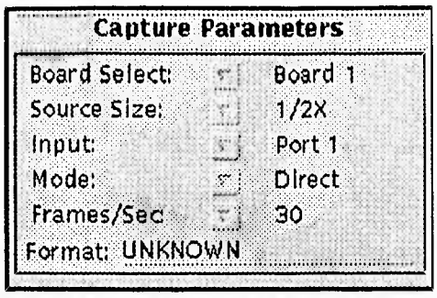


Figura 5.5b Ventanas de SunVideo

La Ventana Principal

El programa despliega la pantalla principal con los parámetros de default iniciales. Esta ventana tiene dos modos de operación: captura y reproducción. El usuario podrá de manera alternativa grabar o reproducir audio.

Para poder desplegar video desde un dispositivo como una cámara o semejante se deberán seguir los siguientes pasos:

1. Verificar en la ventana principal que la opción *capture* esté activada, si se encuentra en la opción *File* presionar el botón *Video Source* dentro de la misma ventana y después seleccionar la opción *capture*.
2. Verificar que los parámetros dentro de la opción *capture* sean los correctos para la configuración deseada, esto se logra mediante un click del *ratón* en el menú para observar las opciones disponibles. Por ejemplo, asegurarse de que la entrada está bien direccionada al puerto donde está conectado el dispositivo.
3. Hacer un click sobre el botón *play* para comenzar a desplegar la imagen. Se tiene la alternativa de cerrar o abrir la ventana de desplegado *(display)* mediante un click en el botón *Monitor* como se muestra en las Figuras 5.6 y 5.7.

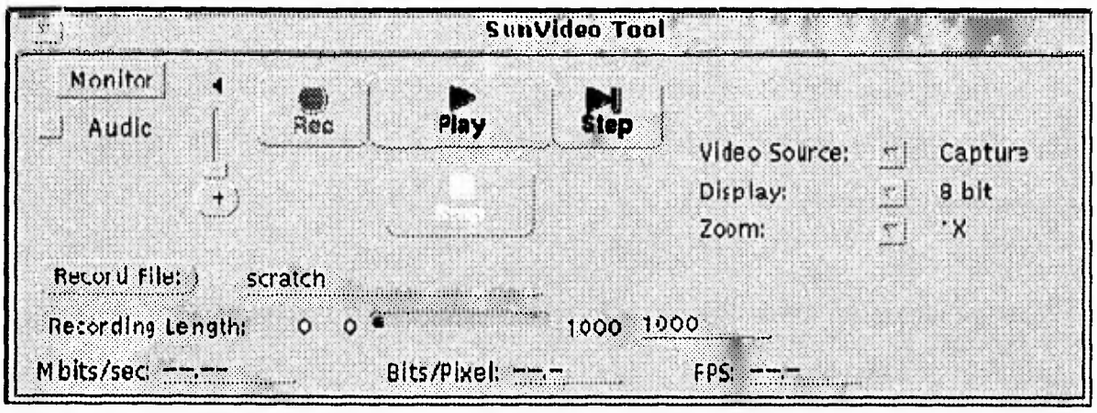


Figura 5.6 Desplegando video desde un dispositivo externo

Una vez que la imagen ha sido desplegada, se visualizará toda la información acerca del video. Por ejemplo, la resolución en pixeles de la imagen y cantidad de imágenes por segundo representadas en la pantalla.

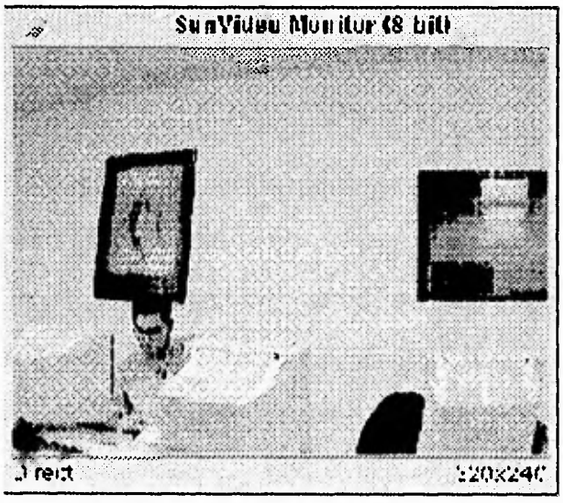


Figura 5.7 Ejemplo del "monitor" de SunVideo

Para detener el desplegado del video, presionar el botón *Stop*.

Reproducir video desde un archivo

Es posible reproducir un video desde la interfaz de ventanas o mediante la línea de comando. Para lograrlo a través de la interfaz de ventanas se deberán seguir los siguientes pasos:

1. Deberá darse un click con el ratón en el menú *Video Source* para seleccionar la opción *file* (archivo).
2. En la ventana dar un click al botón *Load* (cargar) para poder recuperar el archivo a ser reproducido.
3. Teclear la ruta y el nombre del archivo que contenga el video.
4. Presionar OK para comenzar a reproducir el video.
5. Dar un click al botón *play*. Una vez que el video se está reproduciendo se podrá detener en cualquier momento mediante el botón *stop*.

Snapshot

Esta aplicación permite capturar regiones, ventanas o la pantalla entera del monitor a color o blanco y negro. A este tipo de imágenes se les llama *raster* y pueden ser almacenadas como archivos tipo *raster* con una extensión *.rs*.

Cuando se utiliza el *Snapshot* en un monitor blanco y negro, las imágenes ahí creadas serán siempre en esa tonalidad, aunque se quieran cambiar a color con otro *snapshot* en otro monitor. Cuando se utilizan monitores a color, las imágenes serán de ese tipo, en algunos monitores se tiene la opción de poder realizar algunos *snapshots* en blanco y negro.

Snapshot trabaja en manera conjunta con el *Imagetool* para poder visualizar y manipular las imágenes.

Es posible tomar imágenes de la pantalla completa, de cualquier región rectangular que se quiera definir, o de cualquier ventana de la pantalla.

Esta aplicación es verdaderamente útil cuando se quieren obtener todo tipo de imágenes en una estación de trabajo, no importando el tipo de imagen que se encuentre en pantalla.

Para abrir la aplicación *Snapshot*, se deberá elegir dentro del menú de programas en el área de trabajo del usuario (Workspace) la opción *programs>Snapshot* (ver Figura 5.8).

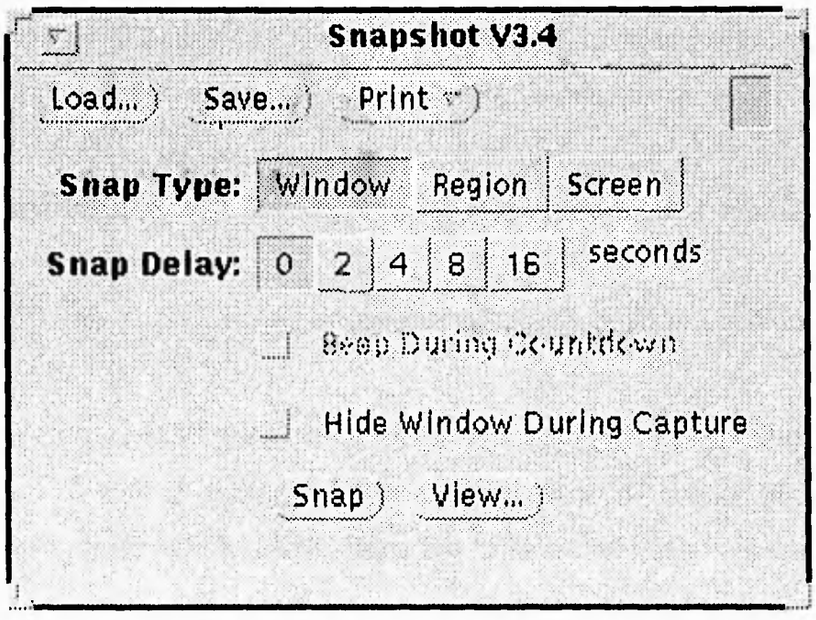


Figura 5.8 Ventana del snapshot

Tipos de Snap (Snap Type)

El *Snapshot* nos ofrece tres opciones diferentes para crear snapshots:

- Para capturar una ventana existente en la pantalla (*Window*).
- Para capturar una región definida por el usuario (*Region*).
- Para capturar por completo la pantalla (*Screen*).

Nota: El *snapshot* no captura el cursor o puntero del *ratón*.

Lapsos de captura del Snapshot (Snap Delay)

Snapshot tiene un reloj que se puede utilizar para seleccionar el número de segundos entre el tiempo de captura de la imagen. Este período es especialmente útil cuando se están capturando figuras de menús que se quieran desplegar después de empezar a tomar el *snapshot*. Una vez que el número de segundos ha sido seleccionado, comenzará un tono de *beep* mientras la captura se realiza, una vez que termine sonará un tono mayor el cual indicará que la captura ha finalizado.

Escondiendo la pantalla de Snapshot durante la captura

Algunas veces se querrá tomar una imagen de una larga porción de la pantalla por lo que es posible esconder la ventana del snapshot durante la captura de esa imagen. Esto se logra con un click del *ratón* en la opción *Hide Window During Capture* de la ventana del *snapshot* para que se oculte en el proceso de captura de una región o una pantalla.

Capturando ventanas

Es posible obtener *snapshots* de ventanas siguiendo los siguientes pasos:

1. Asegurarse de que la ventana que se quiere capturar esté completamente visible. Si la ventana está parcialmente oculta por alguna otra ventana, esa ventana que está ocultando a la imagen también aparecerán en el *snapshot.*

2. Dar un click en la opción *Snap* de la ventana del *snapshot.*

3. Ajustar el tiempo de captura en segundos para realizar la captura de la imagen.

4. Si se desea esconder la pantalla del *snapshot* durante la captura de la imagen, seleccionar la opción apropiada en la ventana escondida durante el ajuste del proceso de captura como se indicó anteriormente.

5. Mover el puntero del *ratón* hacia la parte superior de la ventana que se quiere capturar y posteriormente seleccionarla para comenzar con la captura. Cuando el puntero indicador del *ratón* está en la parte superior de una ventana, entonces la ventana será capturada en forma completa, incluyendo los bordes.

Cuando el *snapshot* ha sido concluido, aparecerá un letrero en la parte inferior de la pantalla diciendo: *snap succeeded*, indicando que la captura de la ventana ha terminado, y a continuación se podrá salvar esa ventana previamente capturada, en un archivo con el formato que más se apegue a nuestras necesidades.

Capturando regiones de pantalla

Se pueden tomar *snapshots* de cualquier área rectangular simplemente utilizando la opción correspondiente a la opción *Region* del *Snap Type* en la ventana del *Snapshot.*

El siguiente ejemplo nos muestra cómo tomar un *snapshot* de una región:

1. Seleccionar la región en la pantalla a ser capturada, y verificar que ésta no esté encimada con otras ventanas.

2. Dar un click a la opción *Region* para indicar que se desea hacer un snapshot de una región.

3. De ser necesario, ajustar el reloj para que se escuchen los tonos de *beep* durante el proceso de captura de la región.

4. Dar un click a la opción *Snap*.

5. Mover el puntero del *ratón* hacia una esquina del área a ser capturada, después marcarla sin despegar el botón del *ratón* hasta el área deseada. Posteriormente, aparecerá un rectángulo que indica los límites de la región especificada.

6. Una vez que la región está definida, dar un click a *adjust* para comenzar con la captura automática de la región.

Snapshot de la pantalla

Es posible tomar un *snapshot* de toda la pantalla siguiendo los siguientes pasos:

1. Asegurarse de que la opción de capturar pantalla esté activada en la ventana del *snapshot* (*Screen*).

2. Dar un click *hide window During Capture*. Se podrá omitir este paso en el caso de que se quiera incluir a la ventana del *snapshot* dentro de la misma pantalla.

3. Dar un click a *Snap* para comenzar con la captura de la pantalla, entonces la pantalla del *snapshot* desaparecerá mientras se realiza la captura y reaparecerá automáticamente una vez finalizada o en el caso que se haya elegido el reloj y un intervalo de segundos para llevar a cabo este paso, al momento de terminar los tonos de *beep*, reaparecerá la ventana del *snapshot* automáticamente.

Visualizando una imagen capturada

Para visualizar un snapshot:

Dar un click al botón *view* del *ratón*.

A continuación aparecerá el siguiente mensaje: *Starting Image Tool* en la esquina inferior izquierda de la ventana. Esta acción automáticamente desplegará el *Image Tool* y con ello, la figura o imagen que previamente fue capturada (ver Figura 5.8).

En ese momento podemos darle a la imagen el formato que más nos convenga. Por ejemplo, la figura anterior nos muestra el *image tool* con la imagen previamente capturada. Para guardarla, nos vamos a la opción *File* y aparece un menú, en ese momento tecleamos el nombre del archivo con el que reconoceremos nuestra imagen y seleccionamos el formato adecuado.

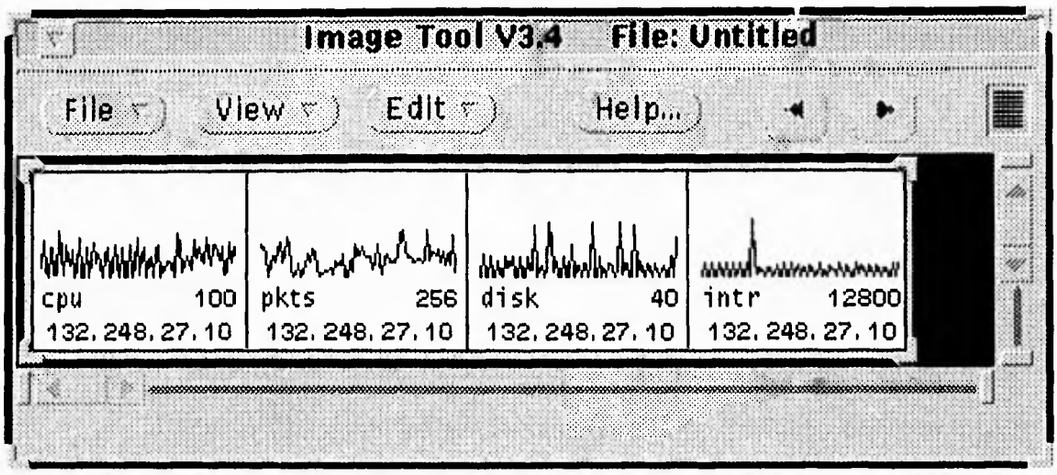


Figura 5.8 Imagen capturada por SnapShot y vista desde ImageTool

XV

Es una herramienta que nos sirve para desplegar una gran variedad de formatos de imágenes en estaciones de trabajo. Se pueden tener todo tipo de opciones disponibles para una imagen, que va desde ponerla de papel tapiz en nuestra pantalla, cambiar tamaño y tonalidades, invertirla, darle una inclinación en los grados que el usuario desee y en general una serie de opciones que a continuación se describirán con la ayuda de pantalla que nos aparece cuando entramos a XV (ver Figura 5.9).

Podemos observar que existen una serie de opciones disponibles para poder editar y modificar nuestra imagen en diferentes formatos y formas.

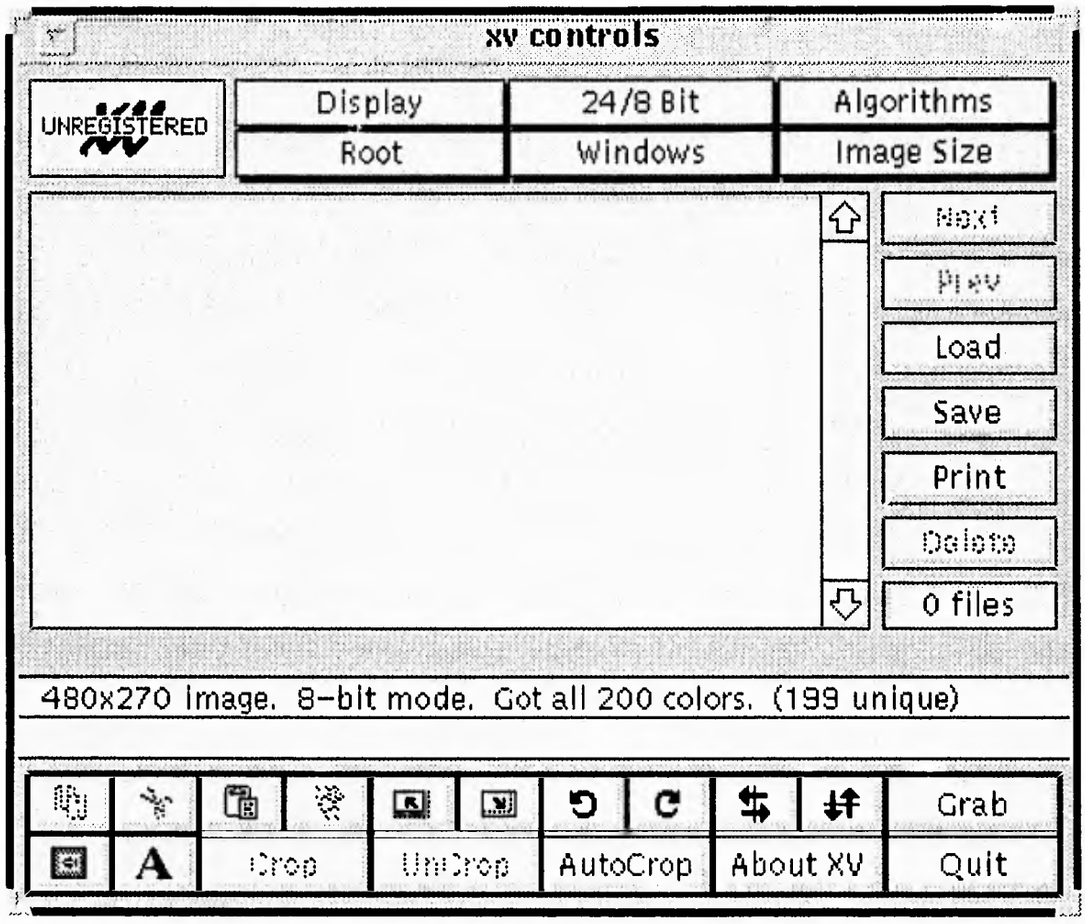


Figura 5.9 Controles de XV

En la ventana superior se encuentra la opción *Display*, ésta nos permite desplegar la imagen que nosotros estemos editando en diferentes tonalidades, que pueden ir desde tonos suaves hasta una completa resolución de color. Al lado derecho, se encuentra la opción de *24/8 Bit*, esta opción permite convertir a la imagen que se esté editando a colores de 8 ó 24 bits dependiendo de la resolución que se quiera obtener. Junto a esta opción se encuentra la de *Algorithms*, la cual nos permite darle automáticamente diferentes vistas y escalas de resolución para brindar a la imagen un tono especial diferente a su original y dentro de esa misma, podemos rotar la figura en la opción *copy rotate* y podemos establecer el número de grados

que deseamos girar a nuestra figura. Bajo la opción Display, se encuentra *Root*, la cual se encarga de dar opciones de diversificación de la imagen, por ejemplo, que se ponga de fondo en toda la pantalla, o que ponga un papel tapiz de ladrillos y enmedio de la pantalla quede la figura, o haga una serie de combinaciones simétricas con la figura a lo largo de toda la superficie de la pantalla. Junto a esta opción se encuentra la de *Window* que como su nombre lo indica, nos despliega una serie de ventanas relativas a la figura o al ambiente del sistema. Por ejemplo, se puede tener la alternativa de abrir la opción de desplegar la ventana donde se pueda visualizar la información de la imagen, de cuántos bytes ocupa, de la resolución en pixeles y su formato; también se puede abrir una ventana para editar comentarios acerca de la figura, y dentro de esta misma opción se encuentra una herramienta muy versátil llamada *color edit*, que nos brinda todas las herramientas para darle manualmente todas las tonalidades y cambios de color a la misma imagen. Junto a esta opción se encuentra la de *Image Size*, que nos sirve para ajustar el tamaño de nuestra imagen a nuestra conveniencia. Al extremo derecho de nuestra pantalla de *xv controls* se encuentran todas las opciones disponibles para manipular a esa imagen como archivo y a su vez a otros archivos, como cargar, imprimir, guardar, etc.

Es importante recordar que cuando se está guardando un archivo, es posible hacerlo con otro formato para que en un dado caso se ajuste a las necesidades de otro paquete que no reconozca su formato original. Por ejemplo, Word para Windows no reconoce archivos tipo GIF, pero con XV un archivo se puede transformar de GIF a BMP y de esa manera ser reconocido por Word para Windows.

En la parte inferior de la pantalla, se encuentran opciones de atributos adicionales para la imagen. Por ejemplo, editar un texto o girarla 180º o 90º en forma directa, darle un acercamiento automático o alejarla para que se vea más pequeña que su tamaño original. En general, se podría decir que con XV en cuestión

de imágenes, se pueden hacer las tareas básicas de conversiones de formatos, cambios de color, tamaño, etc.

Touchscreen

Aunque el *touchscreen* que utilizamos en el proyecto ya se contaba en la DCAA, es conveniente también mencionar algunas tecnologías para seleccionar la adecuada.

Con la proliferación de las computadoras, más y más diseñadores de sistemas están aprovechando las ventajas de la tecnología de *touchscreen* principalmente por su facilidad de uso y velocidad de respuesta. De acuerdo a Dataquest, una fábrica de pantallas *touchscreen* ubicada en San José California (Estados Unidos), las ventas anuales de estos dispositivos llegaron en 1994 a 150 millones de dólares, con una estimación en ventas de 300 millones de dólares para 1998.

Actualmente existe una variedad de modelos para casi cualquier tipo de aplicación y 5 tipos principales de tecnología en *touchscreen*. Es responsabilidad del diseñador balancear los requerimientos del sistema con la tecnología de *touchscreen* apropiada.

Los cinco tipos básicos son capacitivo, resistivo, de ondas de superficie, de búsqueda infrarroja y de sensado de fuerza. Cada uno cuenta con sus ventajas y desventajas y la mayoría de los vendedores está de acuerdo en que ninguna tecnología por sí misma es perfecta para todas las aplicaciones.

Para seleccionar la tecnología más apropiada es esencial examinar cómo funciona cada una de ellas.

Touchscreen resistivo

El *touchscreen* de resistencias consiste en una capa de vidrio formada para llenar la superficie de desplegado. La superficie de vidrio está recubierta de una capa transparente conductiva y después cubierta con una capa de plástico la cual también está cubierta con material conductivo, a su vez, ayuda a darle mayor consistencia y protección a la pantalla. En el interior, esta pequeña capa conductiva está finamente separada del vidrio mediante pequeños círculos, cada uno más delgado que una milésima de pulgada. Cuando un usuario toca la pantalla, la superficie conductiva de plástico hace contacto con la capa de vidrio. El controlador detecta el contacto y automáticamente calcula unas coordenadas tanto en el eje X como el Y.

Las principales aplicaciones de esta tecnología está en restaurantes, hospitales y manufactura principalmente porque son resistentes a la humedad y al polvo. También, se pueden operar con manos que utilicen guantes. Sin embargo, esta tecnología no es tan eficiente como otras. La Figura 5.10 muestra el esquema básico de un *touchscreen* resistivo.

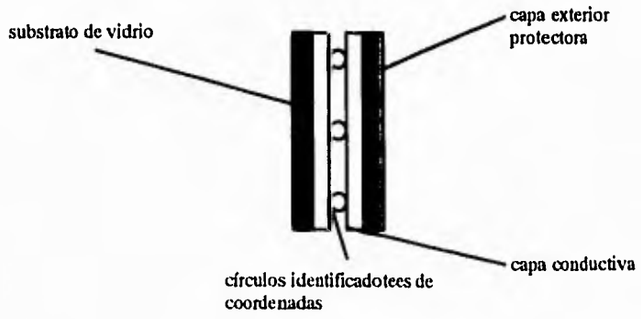


Figura 5.10 esquema de un touchscreen resistivo

Touchscreens capacitivos

Los *touchscreens* capacitivos utilizan una capa de vidrio cubierta con una capa muy delgada de metal sobre la superficie de la pantalla. Al contacto de la misma, el usuario crea una carga capacitiva con el campo eléctrico, dibujando una pequeña cantidad de corriente en el punto de contacto.

La corriente que fluye de cada esquina de la pantalla es proporcional a la distancia del dedo del usuario. De esta manera, los niveles de la corriente son medidos por los controladores y utilizados para localizar el contacto de la pantalla. Los *touchscreens* capacitivos son generalmente más brillantes y más claros que los resistivos, pero no tanto como los de ondas de superficie o los de sensado de fuerza.

Tal como los *touchscreens* resistivos, los capacitivos también pueden ser sellados para prevenir la penetración de agua o polvo al sistema interno. Sin embargo, la desventaja que presenta este tipo de tecnología es que no se pueden utilizar con guantes o con algún otro material no-conductivo. Este tipo de dispositivos se pueden encontrar en juegos de video, para dar información al público en general de cualquier tema en una terminal y para puntos de venta en ciertos establecimientos.

Los touchscreens *capacitivos* a su vez están sujetos a descalibrarse por las condiciones del ambiente y por las propiedades conductivas de los usuarios, las cuales generalmente son diferentes; o se puede presentar el caso en que el usuario por un descuido toque la pantalla junto con una superficie de metal y con el tiempo la pantalla junto con sus controladores de corriente se descalibren.

Los touchscreens *capacitivos* son a su vez muy vulnerables a los rasguños, los cuales pueden destruir la capa conductora y causar que el sistema resulte inoperable.

La Figura 5.11 muestra el esquema básico de un *touchscreen* capacitivo:

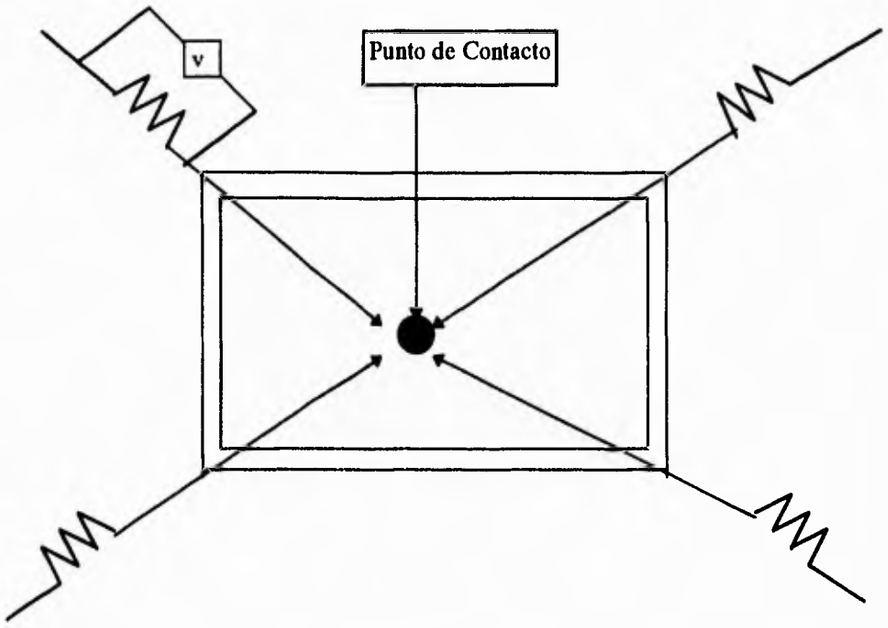


Figura 5.11 esquema del touchscreen capacitivo

Touchscreen de ondas de superficie

Un *touchscreen* de ondas de superficie consiste en una capa de vidrio muy clara. Cada eje en la capa tiene un transductor piezoeléctrico transmisor y receptor, y una serie de franjas reflectoras. El controlador del *touchscreen* manda ondas constantes de 5 Mhz a los transmisores transductores, los cuales convierten las señales en ondas de superficie. Las ondas de superficie son ondas mecánicas que se propagan en la superficie de materiales como el vidrio. Las franjas reflectoras diversifican el calor a través de la superficie de la pantalla. Cuando un dedo toca la

pantalla, una porción de la onda se absorbe. El resultado del cambio en la señal recibida es analizada por el controlador y digitalizada en coordenadas X y Y. Asimismo, un tercer eje Z es también determinado por el controlador para medir la cantidad de la señal que fue absorbida.

La tecnología de superficie de onda apareció por primera vez en 1987 y continua produciéndose actualmente como el único *touchscreen* que no necesita capas de plástico, solamente existe vidrio entre el usuario y la imagen, este tipo de tecnología ofrece mayor durabilidad y claridad que el capacitivo y el resistivo. La onda de superficie es una tecnología muy confiable, ya que está basada en la medida del tiempo. Como resultado, no depende de factores externos como la temperatura y humedad que siempre están cambiando. Gracias a su material que es completamente hecho de cristal, no es vulnerable en su superficie en comparación con los de plástico, y está libre de riesgos como sería un rasguño en su superficie. Estas ventajas permiten que se encuentren en sitios como supermercados, centros de actividad turística y tiendas. Sin embargo, este tipo de *touchscreen* no se recomienda para lugares en donde contaminantes líquidos entren en regular contacto con la pantalla, ya que causan su irremediable inoperabilidad.

Touchscreens de búsqueda infrarroja

La tecnología de búsqueda infrarroja (IR), trabaja con el principio de un haz de luz infrarroja invisible para la vista humana. Este haz de luz se produce por una serie de leds emisores de luz infrarroja y una serie de fototransistores en el otro extremo de la pantalla, creando una malla de luz a través de toda la pantalla. Cuando un objeto interfiere con esos haces de luz, automáticamente los fototransistores detectan la ausencia de luz en esa región y transmiten la señal que identifica las coordenadas X y Y de la región en que hubo contacto con la pantalla.

No existe ninguna capa de ningún material cubriendo los sistemas de contacto infrarrojo, lo cual significa que el usuario retiene la calidad de la imagen original. Este tipo de *touchscreen* es muy vulnerable al vandalismo y a lugares donde exista poca seguridad, ya que no cuentan con alguna capa protectora.

Desafortunadamente, este tipo de *touchscreens* son muy susceptibles a falsos contactos, a consecuencia de que detectan todo tipo de cuerpos que interfieran con las ondas de luz, algunas veces detectan contactos que no se tienen la intención de hacer o partículas de polvo que accidentalmente caen en la superficie y provocan falsos contactos, por lo que llegan a ser ineficientes hasta que son nuevamente limpiados. A diferencia de otras tecnologías, estas pantallas no pueden ser fácilmente programadas para ignorar contaminantes que se pudieran situar en su superficie.

A continuación se muestra la Figura 5.12 con el esquema básico de un *touchscreen* de detección infrarroja:

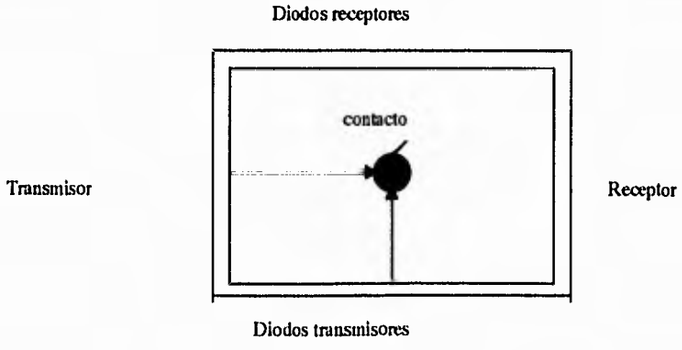


Figura 5.12 Esquema de un touchscreen de detección infrarroja

Touchscreen de sensado de fuerza

En sistemas *touchscreen* de sensado de fuerza se coloca un monitor sobre una plataforma especialmente diseñada, que es la que le da al monitor su sensibilidad al contacto. La plataforma consiste en un plato que descansa sobre unos resortes con capacidad de movimiento en todas sus direcciones. Cuando la pantalla es tocada, la fuerza aplicada causa un suave desplazamiento entre la parte superior del plato y la base de la pantalla. Sensores internos miden en ese momento el nivel de fuerza ejercido que causó este movimiento y los lectores son utilizados para calcular la localización del dedo del usuario.

Tal como los *touchscreens* de búsqueda infrarroja, no existe una capa de ningún material cubriendo a la pantalla, es por eso que la calidad original de la pantalla se mantiene constante. Los *touchscreens* de sensado de fuerza son más fáciles de integrar con los monitores existentes que la mayoría de los sistemas *touchscreen* restantes. El usuario solamente tiene que colocar su monitor sobre la plataforma especial de este sistema, realizar una calibración de rutina y entonces el monitor se convierte en un *touchscreen* de sensado de fuerza. A su vez, estos sistemas representan una buena alternativa para aquellos usuarios que quieran cambiar todo su sistema convencional a uno de este tipo, además de que puede trabajar con una gran variedad de monitores que no pesen más de 90 libras.

La desventaja de este tipo de *touchscreen* es su calibración, la cual es una tarea muy compleja que consume mucho tiempo. Y si surge la necesidad de elevar un poco el monitor, una nueva calibración se requerirá. En adición a esto, debido a los cálculos tan complejos que se requieren en la tecnología de sensado de fuerza, el tiempo de respuesta del sistema es relativamente lento. Esto también es menos ventajoso cuando se requiere por ejemplo, arrastrar un ícono o algo semejante.

En conclusión, la tecnología que se elija dependerá de la aplicación que se le quiera dar y a su peculiar conjunto de requerimientos. Esto se puede observar en la siguiente tabla:

Parámetros	Resistivo 5 cables	Resistivo 4 cables	Capacitivo	Ondas de superficie	Búsqueda Infrarroja	Sensado de fuerza
Velocidad de respuesta	rápido 10 ms	rápido 10-20 ms	rápido 10-15 ms	rápido 20 ms	moderado 20-35 ms	lento 150-250 ms
Porcentaje de resolución de pantalla en función de imagen original	75 %	75 %	85 %	92 %	100 %	100 %
Resistencia a rasguños	Resiste-te, sin falla después de fallas visibles	Sensible a daños en su pantalla.	Resistente pero con tendencia a fallas debido a rasguños invisibles	Imprevisto	Extrema-damente resistente	Extrema-damente resistente
Resolución de contacto	Alto 300/pulg	Alto 25-100 /pulg	Alto 100/pulg	Medio 30/pulg	Bajo 8/pulg	Medio 40/pulg
Sujeto a falsos con-tactos	No	No	No	No	Si	No
Costo Total	Bajo	Bajo	Bajo	Bajo	Bajo	Alto

tabla 4 Comparación de tecnologías de touchscreens

Por último, el *touchscreen* que vamos a utilizar en el kiosco de información es de tipo capacitivo, dadas las características del fabricante. Esto no quiere decir que tiene que ser necesariamente de esta tecnología pero, dados los 5 tipos básicos de tecnología en *touchscreen*, el más recomendable es éste.

5.3 Desarrollo

En la sección 5.1 mencionamos la propuesta del proyecto que presentamos en la presente tesis. Ya habíamos incluido una descripción de Mosaic en el capítulo 4, también conceptos relacionados con este software.

Mencionamos las características del equipo en cuanto a hardware se refiere, asimismo, incluimos herramientas que utilizamos en la generación de los archivos que necesitamos para generar los documentos que nuestro sistema debe tener. En esta sección hablaremos en sí, del desarrollo del proyecto.

Primero, comenzaremos presentando la información necesaria para la instalación de Mosaic y como fue posible adaptarlo a un modo kiosco que es exactamente lo que necesitamos.

En el capítulo 4, hablamos un poco acerca del servidor HTTP, el cual es necesario instalar para la transferencia de documentos hipertexto dentro de la red, aquí veremos como instalarlo y configurarlo en forma adecuada para que éste sea lo más seguro posible.

5.3.1 Instalación y Configuración del cliente Mosaic

Como se mencionó en el capítulo 4, para la instalación de un cliente Mosaic como visualizador de documentos hipertexto, se puede hacer vía servidores *ftp* anónimos.

Básicamente hay 2 formas de llevar a cabo la instalación:

1. Obtener el código binario previamente compilado que se encuentra en el servidor ftp anónimo *ftp.ncsa.uiuc.edu* (como se describe en la sección 4.3).
2. Obtener el código fuente y compilarlo localmente, también realizando una conexión al servidor especificado en el inciso anterior.

Si se escoge el primer método, lo único que se tiene que hacer es una conexión ftp al servidor *ftp.ncsa.uiuc.edu* y después cambiarnos al directorio */Mosaic/Unix/binaries*. El siguiente ejemplo nos muestra el cambio de directorio una vez que la conexión al servidor ha sido establecida.

```
ftp> cd /Mosaic/Unix/binaries
250 CWD command successful.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 26
drwxr-xr-x  6 12873   wheel     2048 Apr  5 13:24 .
drwxr-xr-x  6 12873   wheel     2048 Dec 23  1994 ..
-rw-r--r--  1 12873   other      169 Oct 10  1994 .index
drwxr-xr-x  2 15220   202       2048 Apr 13 01:44 2.5
drwxr-xr-x  2 101     10        2048 Jun 23 16:08 2.6b
drwxr-xr-x  2 12873   wheel     2048 Aug 22  1994 app-defaults
drwxr-xr-x  5 15220   202       2048 Mar 12 04:13 old
226 Transfer complete.
```

El lector puede notar que aparece una serie de directorios, aunque se observe que existe un directorio llamado 2.6b (es la versión 2.6 beta), el que nos interesa es el 2.5. En él, se encuentra Mosaic 2.5 precompilado para varias arquitecturas de máquinas tal como se muestra a continuación:

```
ftp> cd 2.5
250-Please read the file README-2.5
250- it was last modified on Sat Apr  1 09:37:59 1995 - 92 days ago
250-Please read the file README-2.5-solaris
250- it was last modified on Thu Apr 13 01:40:07 1995 - 80 days ago
250 CWD command successful.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 49722
drwxr-xr-x  2 15220   202        2048 Apr 13 01:44 .
drwxr-xr-x  6 12873   wheel      2048 Apr  5 13:24 ..
-rwxr-xr-x  1 15220   202      845660 Mar 12 03:58 Mosaic-alpha13-2.5.Z
-rwxr-xr-x  1 15220   202      760451 Mar 12 03:59 Mosaic-alpha30-2.5.Z
-rwxr-xr-x  1 15220   202     1953997 Mar 12 03:59 Mosaic-dec-2.5.Z
-rwxr-xr-x  1 15220   202     2193623 Mar 12 09:24 Mosaic-hp-2.5.Z
-rwxr-xr-x  1 15220   202     2478729 Mar 12 04:00 Mosaic-ibm-2.5.Z
-rwxr-xr-x  1 15220   202     1367411 Mar 12 04:00 Mosaic-indy-2.5.Z
-rwxr-xr-x  1 15220   202     1632311 Mar 12 04:01 Mosaic-linux-2.5.Z
-rwxr-xr-x  1 15974   202     2753415 Mar 31 13:54 Mosaic-linux-static-2.5.Z
-rwxr-xr-x  1 15220   202     1493021 Apr  1 09:34 Mosaic-sgi-2.5.Z
-rwxr-xr-x  1 15220   202     2451643 Mar 12 04:01 Mosaic-solaris23-2.5.Z
-rwxr-xr-x  1 18381   202     2444387 Apr 13 01:42 Mosaic-solaris24-2.5.Z
-rwxr-xr-x  1 15220   202     2531137 Mar 12 04:02 Mosaic-sun-2.5.Z
-rwxr-xr-x  1 15220   202     2537161 Mar 12 04:03 Mosaic-sun-lresolv-2.5.Z
-rw-r--r--  1 15220   202        1681 Apr  1 09:37 README-2.5
-rw-r--r--  1 18381   202         828 Apr 13 01:40 README-2.5-solaris
226 Transfer complete.
```

A finales del año pasado, cuando la versión 2.4 de Mosaic fue oficialmente liberada en el NCSA, ya se tenía disponible la versión 2.5 beta (recuerde que la versión beta es una oficialmente no liberada) y desde entonces teníamos pensado trabajar con ella, por la principal razón de que a partir de esta versión, Mosaic puede operar en modo kiosco (veremos más detalles de esto más adelante). Mosaic en su

versión 2.5 fue oficialmente liberado en los primeros meses de este año (en abril ya se contaba con ella) y desde entonces, comenzamos a trabajar con la misma.

El siguiente paso es transferir a nuestra máquina local, el archivo binario correspondiente. En este caso, recuerde que estamos trabajando en una SparcStation 5 con sistema operativo Solaris 2.3. No debemos olvidar que para los archivos binarios, el comando *ftp* debe operar en el modo binario (*binary*).

```
ftp> bin
200 Type set to I.
ftp> get Mosaic-solaris23-2.5.Z
local: Mosaic-solaris23-2.5.Z remote: Mosaic-solaris23-2.5.Z
200 PORT command successful.
150 Opening BINARY mode data connection for Mosaic-solaris23-2.5.Z
(2451643 bytes).
226 Transfer complete.
2451643 bytes received in 35.37 seconds (67.69 Kbytes/s)
ftp> quit
221 Goodbye.
```

En este momento contamos con el archivo binario del cliente Mosaic en la máquina local. Observe que el archivo está comprimido, por lo que será necesario utilizar el comando *uncompress* de Unix y cambiar los permisos de ejecución. Antes de correrlo, se debe definir la variable de ambiente LD_LIBRARY_PATH de la siguiente forma:

```
% setenv
LD_LIBRARY_PATH=/usr/openwin/lib:/usr/lib:/opt/SUNWmotif/lib:/opt/SUNWits
/Graphics-sw/xil/lib
```

la cual contiene un path (ruta) donde se buscarán las bibliotecas que intervienen en el ambiente gráfico del sistema, si es que se necesitan. Una vez hecho esto, al correrlo en la consola de la estación de trabajo o una terminal X, veremos la pantalla de Mosaic con su correspondiente "home page" que tiene por default (ver sección

4.1). Más adelante veremos como cambiar ciertos parámetros de configuración para tener nuestra aplicación en modo kiosco y el "home page" que deseemos.

Para el segundo método, necesitamos obtener el código fuente de Mosaic de forma similar a la anterior, en este caso tenemos que dirigirnos al directorio */Mosaic/Unix/source.*

```
ftp> cd /Mosaic/Unix/source
250 CWD command successful.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 5988
drwxr-xr-x  4 12873   wheel     2048 Jun 22 11:41 .
drwxr-xr-x  6 12873   wheel     2048 Dec 23  1994 ..
-rw-r--r--  1 12873   other      206 Aug 22  1994 .index
drwxr-xr-x  9 15220   202       2048 Mar 12 06:48 Mosaic-2.5
-rw-r--r--  1 15220   202    1110223 Mar 12 06:46 Mosaic-src-2.5.tar.Z
-rw-r--r--  1 101     10     1945511 Jun 22 11:38 Mosaic-src-2.6b3.tar.Z
drwxr-xr-x  2 15220   202       2048 Mar 12 04:08 old
226 Transfer complete.
```

El lector puede notar que solo hay un archivo conteniendo el código fuente. Si se pregunta ¿porqué?, la respuesta es simple. Recuerde cuando hablamos del sistema de ventanas X-Window como GUI del sistema operativo UNIX en las estaciones de trabajo, X-Window ofrece muchas ventajas en un ambiente de red. Las aplicaciones que se realizan en este ambiente gráfico, se basan en bibliotecas (biblioteca Xlib) o cajas de herramientas (toolkit). El sistema X-Window proporciona un conjunto estándar de rutinas en lenguaje C las cuales se encuentran en la biblioteca Xlib. Si una aplicación corre en este ambiente, puede hacerlo en todas las estaciones de trabajo que tengan X-Window no importando la arquitectura de la máquina, hablamos entonces de aplicaciones portables entre sistemas.

Por otra parte, C es un lenguaje de programación de propósito general que ofrece como ventajas economía de expresión, control de flujo y estructuras de datos

modernos y un rico conjunto de operadores. Además, C no es un lenguaje de "muy alto nivel" ni "grande", y no está especializado en alguna área específica de aplicación. Originalmente, C fue diseñado para el sistema operativo UNIX y Dennis Ritchie (uno de los investigadores que diseñaron UNIX) lo implantó sobre el mismo en una DEC PDP-11. El sistema operativo, el compilador y esencialmente todos los programas de aplicación de UNIX están escritos en lenguaje C, el cual no está ligado a ningún hardware o sistema en particular y es fácil de escribir programas que corran sin cambios en casi cualquier máquina que maneje C. En 1983, el *American National Standards Institute* (ANSI) estableció un comité cuyos propósitos eran producir "una definición no ambigua del lenguaje C independientemente de la máquina", cuidando la conservación de su espíritu. El resultado es el estándar ANSI para el lenguaje C, el cual formaliza construcciones sugeridas, especifica una biblioteca estándar, con un conjunto extensivo de funciones para realizar la entrada y salida, la administración de memoria, la manipulación de cadenas y tareas semejantes.

Una vez contestada la pregunta, el siguiente paso es transferir el archivo *Mosaic-src-2.5.tar.Z*.

```
ftp> get Mosaic-src-2.5.tar.Z
local: Mosaic-src-2.5.tar.Z remote: Mosaic-src-2.5.tar.Z
200 PORT command successful.
150 Opening BINARY mode data connection for Mosaic-src-2.5.tar.Z (1110223 bytes).
226 Transfer complete.
1110223 bytes received in 14.83 seconds (73.13 Kbytes/s)
ftp> quit
221 Goodbye.
```

Nuevamente, tenemos que descomprimir el archivo con ayuda del comando *uncompress*, el cual nos genera un archivo en formato *tar*. Para ello, debemos utilizar el comando *tar* de Unix como lo muestra el siguiente ejemplo:

```
tzetzal 18:> uncompress Mosaic-src-2.5.tar.Z
tzetzal 19:> tar xf Mosaic-src-2.5.tar
tzetzal 20:> ls -la
total 6678
drwxr-xr-x  9 ricardo soporte    1024 Jul  2 17:05 .
drwxrwxrwx 13 root    root       6656 Jul  2 17:04 ..
-rw-r--r--  1 ricardo soporte   65724 Mar 12 06:29 CHANGES
-rw-r--r--  1 ricardo soporte    4623 Jan 10 18:02 COPYRIGHT
-rw-r--r--  1 ricardo soporte    5261 Mar 12 06:35 FEATURES
-rw-r--r--  1 ricardo soporte    8958 Jan 11 20:34 Makefile
-rwxr-xr-x  1 ricardo soporte    8970 Jan 11 02:46 Makefile.alpha
-rwxr-xr-x  1 ricardo soporte    9038 Jan 11 02:46 Makefile.dec
-rwxr-xr-x  1 ricardo soporte    9049 Jan 11 02:46 Makefile.hp
-rwxr-xr-x  1 ricardo soporte    9016 Jan 11 02:46 Makefile.ibm
-rwxr-xr-x  1 ricardo soporte    9916 Mar 12 01:23 Makefile.indy
-rwxr-xr-x  1 ricardo soporte    9109 Mar 12 06:40 Makefile.linux
-rwxr-xr-x  1 ricardo soporte    9922 Feb 17 03:59 Makefile.sgi
-rwxr-xr-x  1 ricardo soporte    9378 Feb 17 03:30 Makefile.solaris
-rwxr-xr-x  1 ricardo soporte    8958 Jan 11 02:46 Makefile.sun
-rw-r--r--  1 ricardo soporte 3194880 Jul  2 17:04 Mosaic-src-2.5.tar
-rw-r--r--  1 ricardo soporte    4065 Mar 12 06:38 README
-rw-r--r--  1 ricardo soporte     644 Jan 10 18:02 XKeysymDB
-rw-r--r--  1 ricardo soporte    6465 Jan 10 18:02 app-defaults.color
-rw-r--r--  1 ricardo soporte    6464 Jan 10 18:02 app-defaults.color-sgi
-rw-r--r--  1 ricardo soporte    5687 Jan 10 18:02 app-defaults.mono
drwxr-xr-x  2 ricardo soporte     512 Mar 12 01:30 auth
-rw-r--r--  1 ricardo soporte     752 Jan 10 18:02 documents.menu
drwxr-xr-x  2 ricardo soporte     512 Mar 12 01:30 libXmx
drwxr-xr-x  2 ricardo soporte    1024 Mar 12 01:30 libdtm
drwxr-xr-x  2 ricardo soporte    1024 Mar 12 01:31 libhtmlw
drwxr-xr-x  2 ricardo soporte     512 Mar 12 01:30 libnet
drwxr-xr-x  2 ricardo soporte    2048 Mar 12 01:30 libwww2
drwxr-xr-x  4 ricardo soporte    1024 Mar 12 01:31 src
tzetzal 21:>
```

El código fuente está escrito en lenguaje C, así que debemos tener instalado el compilador en la máquina local. Por otra parte, utilizamos también la herramienta de Unix llamada *make* la cual es muy útil cuando tratamos de compilar grandes programas como Mosaic. El programa *make* trabaja sobre un archivo llamado Makefile, podemos observar que el código fuente viene acompañado con varios Makefile (con extensiones alpha, dec, IBM, solaris por mencionar algunos). El

contenido del Makefile son instrucciones para el compilador de C. Por ejemplo, imagine que tiene que compilar un programa, pero necesita incluir una infinidad de bibliotecas tanto de C como del sistema de ventanas conocidas como Xlib (en el caso de Unix, las bibliotecas no siempre se encuentran en los directorios que el archivo Makefile contiene). Al compilar, representa una pérdida de tiempo el tener que teclear la orden con todas las bibliotecas y si existe un error, hay que corregirlo y repetir la operación. Por lo tanto, es más cómodo agilizar la orden teniéndola en un archivo y simplemente invocarlo para que automáticamente se lleve a cabo la compilación, esa es la principal función de *make* en Unix.

Por último, pareciera ser que eso es todo, correr el programa make con la opción *-f* (para cambiar el nombre del archivo Makefile por otro, no necesariamente se tiene que llamar así) y dando como argumento el archivo Makefile apropiado (en nuestro caso Makefile.solaris). Sin embargo, los problemas los encontramos aquí precisamente, los errores se refieren generalmente a rutas de archivos (bibliotecas) incorrectos en nuestra máquina local, a veces las bibliotecas no se encuentran instaladas en nuestro sistema. En ocasiones será necesario y con mucho cuidado, eliminar ciertos nombres de archivos que no necesitamos, esto nos ayudará a eliminar algunos errores (mas no todos).

Los programadores experimentados no tendrán problema alguno en este tipo de compilaciones, pero si usted va a compilar el programa por primera vez, se encontrará con algunos detalles. De hecho, no todos los programas que se encuentran en la Red Internet y al compilarlos, corren a la primera.

Ventajas de tener disponible el código fuente:

1. Es mejor contar con el código fuente puesto que el programa compilado será más seguro que la versión precompilada. En la red Internet podemos encontrarnos

programas compilados sin costo alguno pero, éstos pudieran generar "hoyos de seguridad" en nuestro sistema.

2. Podemos modificar los programas fuentes y modificarlos para realizar la aplicación a nuestro gusto. Por ejemplo, cambiar íconos, mensajes, etc.
3. Podemos insertar el código de una aplicación totalmente ajena al programa fuente e interactuar con ella.

En general, contar con los programas fuentes, podemos modificar y cambiar totalmente el aspecto original del programa. Sin embargo, el lector no debe olvidar que debe tomar en cuenta el archivo llamado COPYRIGHT por los derechos reservados si es que desea distribuir su nuevo software.

Compilación de Mosaic

Ahora, surge la pregunta ¿Cuál opción elegir?. Por un lado tenemos el programa precompilado de Mosaic, y por el otro, podemos generar nuestro propio Cliente con características deseadas. El compilar el código fuente de Mosaic fue una tarea que surgió en la Coordinación Técnica (recuerde que en este lugar se desarrolló el proyecto), a partir de adaptar el visualizador Mosaic a un Modo Kiosco, eso fue a principios del año de 1995. En ese entonces, trabajábamos con la versión 2.4, la cual ya mencionaba los nuevos cambios respecto a las versiones anteriores y la posibilidad de configurar al visualizador en modo Kiosco a partir de la versión 2.5. Pero nosotros no podíamos esperar, no sabíamos cuanto tiempo tardaría en ser liberada oficialmente en el NCSA la versión 2.5 (que era la que necesitábamos), así que se comenzó con el trabajo de la compilación. La idea era eliminar todos los botones de la página de presentación de Mosaic y dejar únicamente, aquellos indispensables. Por ejemplo, el usuario dentro del Kiosco no

tiene porqué hacer conexiones externas y mucho menos comenzar a utilizar el menú sin saber lo que está haciendo, esto nos provocaría muchos problemas los cuales evitaríamos, eliminando del código fuente aquellas partes que hicieran referencia a los botones y al menú principal. De esta forma, se procedió a consultar en la lista de correo *Grupo de Administración y Seguridad en equipos Unix* (gasu) a la que estamos suscritos, solicitando una posible solución a nuestro problema. La respuesta llegó del Matemático Gabriel Ciriaco López Walle (que en estos momentos se encuentra haciendo una Maestría en la Universidad de Illinois en Estados Unidos), y aquí mostramos una copia del mail recibido.

From root Wed Feb 15 17:44 CST 1995
>Return-Path: <gasu@ds5000.dgsca.unam.mx>
Received: from ds5000.dgsca.unam.mx
by tzetzal.dcaa.unam.mx; Wed, 15 Feb 1995 17:44:37 +0600
Received: by ds5000.dgsca.unam.mx (5.65/DEC-Ultrix/4.3)
id AA00730; Wed, 15 Feb 1995 17:38:44 -0600
Date: Wed, 15 Feb 1995 17:38:44 -0600
Message-Id: <9502152337.AA08058@sp72.csrd.uiuc.edu>
Errors-To: diego@ds5000.dgsca.unam.mx
Reply-To: gasu@ds5000.dgsca.unam.mx
Originator: gasu@listas.unam.mx
Sender: gasu@ds5000.dgsca.unam.mx
Precedence: bulk
From: gabriel@acm.org (Gabriel C. Lopez-Walle)
To: Multiple recipients of list <gasu@ds5000.dgsca.unam.mx>
Subject: Re: Ayuda con Mosaic...
X-Listprocessor-Version: 6.0c -- ListProcessor by Anastasios Kotsikonas
X-Comment: Lista del Grupo de Administracion y Seguridad en Unix
X-Mailer: ELM [version 2.4 PL24]
Content-Type: text
Content-Length: 5612
Status: RO
X-Status:

>
> Hola administradores!!
>
> Siento mucho enviar este mensaje en esta lista. De antemano pido
> disculpas.
>

tiene porqué hacer conexiones externas y mucho menos comenzar a utilizar el menú sin saber lo que está haciendo, esto nos provocaría muchos problemas los cuales evitaríamos, eliminando del código fuente aquellas partes que hicieran referencia a los botones y al menú principal. De esta forma, se procedió a consultar en la lista de correo *Grupo de Administración y Seguridad en equipos Unix* (gasu) a la que estamos suscritos, solicitando una posible solución a nuestro problema. La respuesta llegó del Matemático Gabriel Ciriaco López Walle (que en estos momentos se encuentra haciendo una Maestría en la Universidad de Illinois en Estados Unidos), y aquí mostramos una copia del mail recibido.

From root Wed Feb 15 17:44 CST 1995
>Return-Path: <gasu@ds5000.dgsca.unam.mx>
Received: from ds5000.dgsca.unam.mx
by tzetzal.dcaa.unam.mx; Wed, 15 Feb 1995 17:44:37 +0600
Received: by ds5000.dgsca.unam.mx (5.65/DEC-Ultrix/4.3)
id AA00730; Wed, 15 Feb 1995 17:38:44 -0600
Date: Wed, 15 Feb 1995 17:38:44 -0600
Message-Id: <9502152337.AA08058@sp72.csrd.uiuc.edu>
Errors-To: diego@ds5000.dgsca.unam.mx
Reply-To: gasu@ds5000.dgsca.unam.mx
Originator: gasu@listas.unam.mx
Sender: gasu@ds5000.dgsca.unam.mx
Precedence: bulk
From: gabriel@acm.org (Gabriel C. Lopez-Walle)
To: Multiple recipients of list <gasu@ds5000.dgsca.unam.mx>
Subject: Re: Ayuda con Mosaic...
X-Listprocessor-Version: 6.0c -- ListProcessor by Anastasios Kotsikonas
X-Comment: Lista del Grupo de Administracion y Seguridad en Unix
X-Mailer: ELM [version 2.4 PL24]
Content-Type: text
Content-Length: 5612
Status: RO
X-Status:

>
> Hola administradores!!
>
> Siento mucho enviar este mensaje en esta lista. De antemano pido
> disculpas.
>

> Alguno de ustedes que tiene Mosaic sabe como desactivar los menus
> de arriba de la ventana y los botones de la parte de abajo, a manera de
> que solamente Mosaic despliegue home pages y nada mas????
>
> Creo que en la version 2.5 de Mosaic tendria una opcion -kiosk
> pero notengo documentacion al respecto.
>
> Cualquier ayuda es bien recibida.
>
> Muchas Gracias y Feliz 14 de Febrero en 15.
>

Hola, mira, si quieres la version 2.5 la puedes obtener de ftp.ncsa.uiuc.edu:/Mosaic entre otros muchos lugares.

Una manera de hacer lo que tu quieres, que supongo es para simular alguna aplicacion, es quitar las partes que crean esas ventanas en el codigo fuente.

Para hacer esto necesitas estar seguro que mosaic compila bien en tu maquina, ya que estes seguro de eso entonces busca en el archivo src/gui-menubar.c y borra todas las partes que se parezcan a esta:

```
{"File",     'F', NULL, NULL, file_menuspec},
{"Navigate", 'N', NULL, NULL, navi_menuspec},
{"Options",  'O', NULL, NULL, opts_menuspec},
{"Annotate", 'A', NULL, NULL, anno_menuspec},
{"Help",     'H', NULL, NULL, help_menuspec},
```

asegurate de dejar toda la declaracion de la estructura y tambien lo que dice:

```
/* Dummy submenu. */
 {NULL,     NULL, NULL, NULL, NULL},
 {NULL},
```

de tal manera que te quedaria algo asi:

```
static XmxMenubarStruct menuspec[] =
{
/* Dummy submenu. */
 {NULL,     NULL, NULL, NULL, NULL},
 {NULL},
};
```

y asi en cada parte que aparezca algo con "Navigate" p.ej.

Para quitar los botones de abajo, en el achivo src/gui.c tienes que borrar donde se declara ese submenu, si solo borras todas las definiciones de los botones te evitarias muchos problemas, para eso borra todo esto:

```
  /* Children of win->button_rc. */
 {
   win->back_button = XmxMakePushButton
    (win->button_rc, "Back", menubar_cb, mo_back);
   win->forward_button = XmxMakePushButton
    (win->button_rc, "Forward", menubar_cb, mo_forward);
   win->home_button = XmxMakePushButton
    (win->button_rc, "Home", menubar_cb, mo_home_document);
   if (!(Rdata.kiosk || Rdata.kioskNoExit))
    {
    if (!Rdata.simple_interface)
     win->reload_button = XmxMakePushButton
      (win->button_rc, "Reload", menubar_cb, mo_reload_document);
    if (!Rdata.simple_interface)
     win->open_button = XmxMakePushButton
      (win->button_rc, "Open...", menubar_cb, mo_open_document);
    win->save_button = XmxMakePushButton
     (win->button_rc, "Save As...", menubar_cb, mo_save_document);
    win->clone_button = XmxMakePushButton
     (win->button_rc, "Clone", menubar_cb, mo_clone_window);
    if (!Rdata.simple_interface)
     win->new_button = XmxMakePushButton
      (win->button_rc, "New Window", menubar_cb, mo_new_window);
    }
   if (!Rdata.kioskNoExit) {
    win->close_button = XmxMakePushButton
     (win->button_rc, "Close Window", menubar_cb, mo_close_window);
   }
 }
```

y voila!!!, lo demas debe de funcionar porque cada botoncito es una "entidad" distinta, lo cheque en una sparc con SunOS4.1.3 y compilo bastante bien asi que suerte, se ve medio feon, pero funciona.

Lo que si es muy importante es que compiles el mosaic en tu maquina y compile bien, pera despues hacer los cambios y que no haya problemas.

Muchos saludos y suerte.

Gabriel C. Lopez Walle gabriel@acm.org
Center for Supercomputing Research and Development
University of Illinois at Urbana-Champaign

Gracias al Ing. Edgar Valencia Figueroa y al Sr. Fernando Vargas quienes laboran también en la Coordinación Técnica, lograron compilar el código fuente y generar el archivo ejecutable. Claro, después de varios intentos.

Hasta ese momento, teníamos Mosaic compilado en modo Kiosco, pero surgió un nuevo problema. Aunque no aparecen los botones en la pantalla, se pueden activar cada uno de ellos, desde el teclado. Por ejemplo, observe en la Figura 4.1 los botones de la parte inferior, "Back" se activa con la tecla "b", "Forward" con la tecla "f", "Clone" se activa con la tecla "c", "New Window" con la tecla "n", "Save As..." se activa con "s", etc. Por lo tanto, el usuario accidentalmente podría teclear algún caracter "prohibido" y meterse en problemas. La solución para este problema, era modificar nuevamente el código fuente tratando de eliminar aquella parte donde se contemplan las teclas "b", "f", "c", etc., de forma similar a la eliminación de los botones.

Esto nos llevaría tiempo puesto que el código fuente está formado por varios archivos de algunas miles de líneas. Pocos días después, la versión 2.5 de Mosaic fue liberada oficialmente. Nuestro trabajo se vio reducido porque sabíamos que a partir de esta versión, Mosaic puede operar en modo kiosco, así que lo probamos y funcionó sin problema alguno, ya no fue necesario modificar el código fuente de la versión 2.4. Sin embargo, ya sabíamos compilar Mosaic e hicimos lo mismo con la versión 2.5 (nos basamos en los cambios hechos en la versión 2.4).

Obviamente, nuestros primeros intentos por compilar los fuentes de Mosaic 2.5, fueron sin éxito alguno. Los errores básicamente se refirieron a rutas de archivos incorrectas o alguna biblioteca que no se tenía instalada en el sistema,

redefinición de variables, eliminación de rutinas no necesarias, etc., y todo dentro del archivo *Makefile* (hasta ahora no habíamos tocado el código fuente en C). Por lo tanto, el proceso de compilación puede llevar de algunos minutos a varios días.

Finalmente, después de múltiples intentos, logramos obtener el código ejecutable a partir de la compilación del código fuente. Es una gran satisfacción personal ver el mensaje de bienvenida "***Welcome to NCSA Mosaic." cuando la compilación no tuvo error alguno.

La Figura 5.13 muestra algunos mensajes que aparecen al compilar el programa. No mostramos la compilación completa porque nos llevaría páginas enteras de este capítulo, lo que el lector puede observar es el resultado de una segunda compilación (que es mucho más sencilla que la primera porque previamente se generó el código objeto), en esta ocasión, únicamente estamos ligando el código objeto para generar el archivo ejecutable.

La compilación del programa tomará algunos minutos cuando es la primera vez, y el archivo ejecutable que se genera, se llama *Mosaic* encontrándose bajo el directorio *src/*.

Una vez que Mosaic compila correctamente en nuestra máquina, podemos proceder a modificar el código fuente. Aunque ya no es necesario eliminar los botones y desactivar el menú, tenemos la posibilidad de traducir los mensajes de inglés a español y eso fue precisamente lo que hicimos.

Con ayuda de la versión 2.4 de Mosaic compilada, procedimos a realizar los cambios necesarios en la versión 2.5. En este caso, se pudo traducir los mensajes de tal forma que el usuario pueda observar los letreros del visualizador en español.

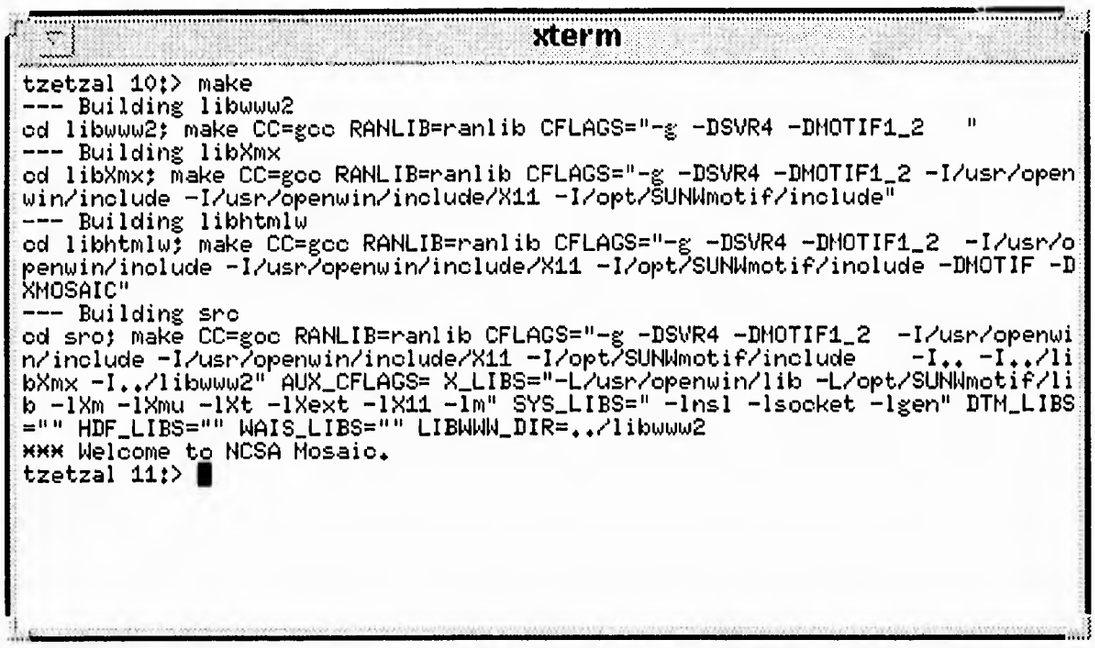


Figura 5.13 Compilación de Mosaic

Por otra parte, en el directorio donde reside el código fuente, existe un subdirectorio llamado *pixmaps*. En él, hay 25 imágenes que representan el logotipo de Mosaic a manera de ícono, el formato de las mismas es XPM de 64x64 pixeles. En su conjunto, estas imágenes logran el efecto que el usuario vé cuando Mosaic trabaja, por ejemplo, realizando la transferencia de algún archivo. El logotipo que se encuentra en la parte superior derecha de Mosaic (un mundo) realiza movimientos dando la apariencia que el "mundo gira", pero en realidad lo que observa, es una secuencia de imágenes que logran hacer ese efecto.

Así, con ayuda del equipo de Multimedia que pertenece a la Coordinación Técnica, logramos cambiar el logotipo de Mosaic. En este caso, se realizó una animación en el paquete 3D-STUDIO y se salvaron las 25 imágenes que se necesitaban. Una vez hecho esto, tuvimos que convertir cada una al formato XPM para posteriormente ponerlas en el directorio *pixmaps* con los nombres de los

archivos anteriores. Después, modificamos el nombre de la estructura a cada uno que define la imagen.

```
static char *icon1[] = {
```

Esto es, reemplazamos el nombre de la imagen original por *icon1*, *icon2*, . . . , *icon25* en cada archivo.

Mosaic modo kiosco

Una vez que se tiene el programa Mosaic compilado (en cualquiera de las dos formas mencionadas anteriormente), para invocarlo simplemente se hace por su nombre. Sin embargo, también podemos especificar opciones para que el programa tenga algún cambio en su aspecto. Este es el caso de la versión 2.5, que a partir de ésta podemos correr Mosaic desde la línea de comandos con la opción *-kioskNoExit*. Al hacerlo, veremos la página de presentación tal como se muestra en la Figura 5.14.

Dentro del directorio de trabajo por default del usuario que corre Mosaic, se puede crear un archivo llamado *.Xdefaults* (ver sección 4.7), en él, se especifican algo que en Unix se maneja mucho, y son los recursos X. Cada uno de ellos, representa un parámetro diferente y podemos definir por ejemplo, el "home page" de default, tipos de letra para los encabezados en HTML, algún directorio temporal, comando de impresión, tamaño por default de la ventana de Mosaic, etc., tal como se mencionó en el Capítulo 4.

Ejemplo del archivo .Xdefaults utilizado para el kiosco:

```
Mosaic*defaultAuthorName: Ing. Edgar Valencia Figueroa
Mosaic*defaultHeight: 875
Mosaic*defaultWidth: 1145
Mosaic*personalTypeMap: .mailcap
```

```
Mosaic*printCommand: relp -P BLUE4850 -c 1 -1
Mosaic*xtermCommand: xterm -fg white -bg blue
Mosaic*Header2Font: -itc-*-*-*-*-*-35-*-*-*-*-*-*-*
Mosaic*Header1Font: -b&h-*-*-*-*-*-45-*-*-*-*-*-*-*
```

El lector puede notar que Mosaic modo kiosco elimina el menú que aparece en la parte superior, así como los botones de la ventana original (compare la Figura 4.1 con la Figura 5.14), de tal forma que sólo nos permite avanzar y regresar entre páginas y regresar al "home page" de default. Tampoco hay forma de salir directamente del programa, el usuario no puede hacer uso de las demás opciones que Mosaic presenta. De esta forma, evitamos que navegue en URL's no permitidos, que cambie la configuración de la ventana, imprimir y en general, evitarle hacer uso de las opciones que no tiene porqué usarlas.

En el archivo .Xdefaults modificamos el tamaño de la ventana de Mosaic con las siguientes opciones:

```
Mosaic*defaultHeight: 875
Mosaic*defaultWidth: 1145
```

Al correr el programa, ocupará toda la pantalla del monitor de la estación de trabajo. El objetivo es que el usuario pueda observar todo el ambiente gráfico y manipularlo con el tacto utilizando el *touchscreen*. Sobre Mosaic no tendrá control alguno, únicamente podrá avanzar entre páginas, porque tampoco tiene que salir del programa debido a que el servicio de información es permanente.

Por otra parte, tenemos especificadas las siguientes opciones:

```
Mosaic*Header2Font: -itc-*-*-*-*-*-35-*-*-*-*-*-*-*
Mosaic*Header1Font: -b&h-*-*-*-*-*-45-*-*-*-*-*-*-*
```

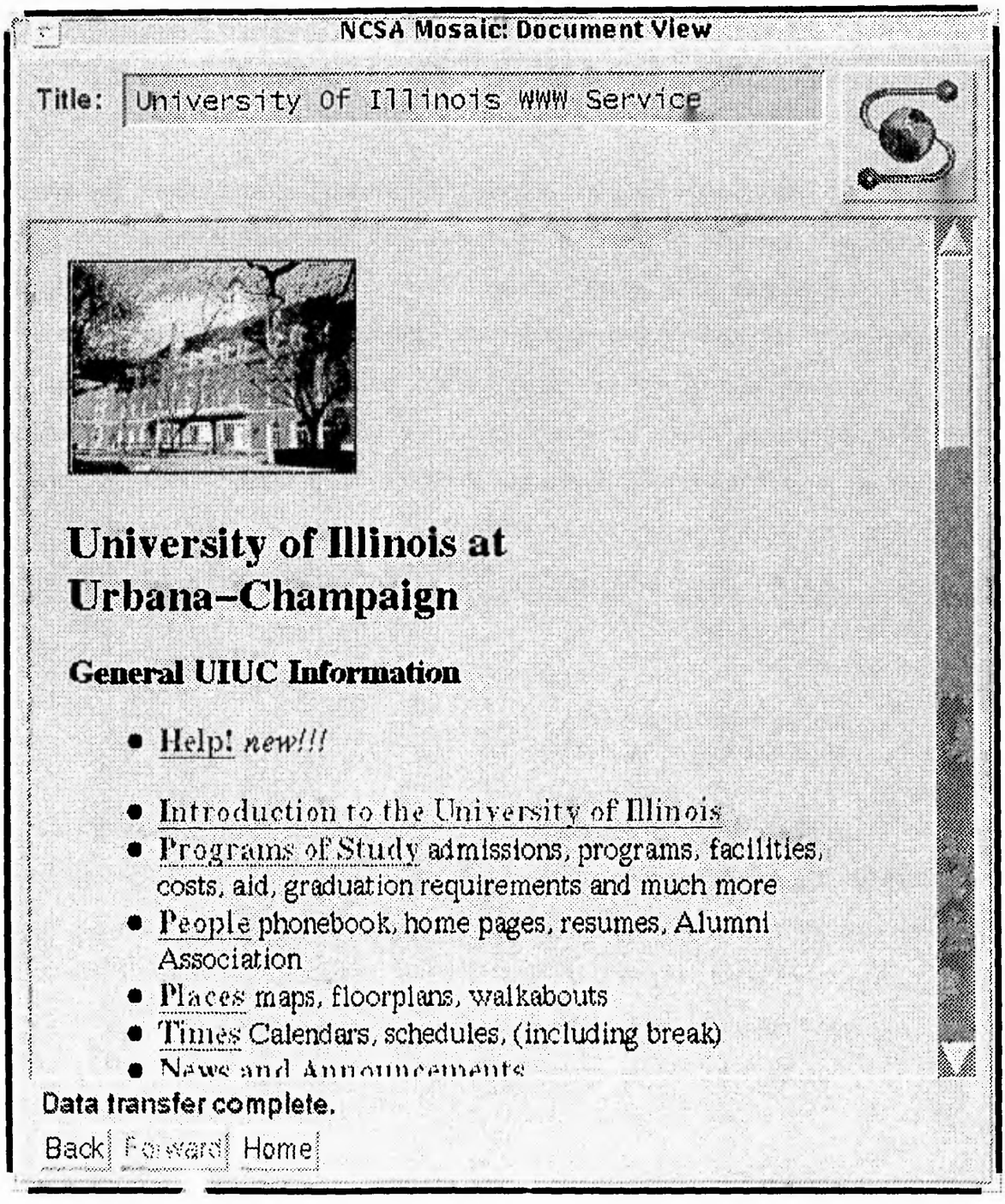


Figura 5.14 Mosaic modo Kiosco

Recuerde que en HTML podemos manejar 6 tamaños de encabezados (del 1 al 6, donde 1 es el más grande). El tamaño por default de los mismos no es suficiente para que el usuario pueda observar lo que Mosaic presenta. Por lo tanto, ampliamos el tamaño de la letra para que los encabezados tipos 1 y 2 aparezcan más grandes y puedan ser observados a mayor distancia.

Una vez que tenemos Mosaic modo kiosco y las modificaciones necesarias en el archivo *.Xdefaults*, nos resta únicamente cambiar el "home page", es más fácil hacerlo desde la línea de comandos con la opción *-home URL*.

El URL no necesariamente tiene que ser un documento que se encuentre en otra máquina, se puede especificar un archivo local. El comando resultante para ejecutar Mosaic con los cambios hechos en el archivo .Xdefaults, es el siguiente:

```
% Mosaic -kioskNoExit -home http://xel-ha.dcaa.unam.mx/home/kiosko/html/home_page.html
```

De esta forma, tenemos el ambiente gráfico de nuestro Kiosco, el cual se puede manipular desde otra estación de trabajo o una PC inclusive. Esto se logra haciendo una conexión remota a nuestro servidor, y definir desde la línea de comandos la variable de ambiente DISPLAY (en la sesión remota), de la siguiente forma:

```
% setenv DISPLAY 132.248.63.249:0.0
```

la cual direcciona la salida del ambiente gráfico a la consola de la estación de trabajo cuya dirección IP es 132.248.63.249 (ésta es la del Kiosco, pero puede especificarse otra e incluso la de una Terminal X). Por un lado, podemos tener una sesión en otra estación de trabajo o en una computadora personal operando como terminal, cada vez que invoque a Mosaic o alguna aplicación gráfica, el despliegue se llevará a cabo en la consola de la estación de trabajo con dirección IP 132.248.63.249 no importando que la sesión original tenga o no capacidades gráficas. Por otro lado, el

servidor con dirección IP 132.248.63.249 debe autorizar el despliegue de ventanas en su consola, lo cual se hace con el comando **xhost** de la forma:

```
% xhost +132.248.27.10
```

donde autoriza a la estación de trabajo con dirección IP 132.248.27.10, el despliegue gráfico de aplicaciones. Así, puedo ejecutar Mosaic desde una sesión remota y la ventana se verá en la consola del Kiosco.

5.3.2 Instalación y Configuración del servidor http

Mosaic puede ser utilizado en forma local, en este caso, no es necesario instalar el servidor http. Simplemente se generan los documentos en hipertexto y las ligas apuntan a los URL's locales. Sin embargo, si deseamos compartir nuestra información con la comunidad de la World Wide Web a través de Mosaic, es necesario instalar el servidor de transferencia de hipertexto (http).

De forma similar a la instalación de Mosaic, para el servidor http existen dos caminos para hacerlo.

1. Obteniendo la versión precompilada o
2. Compilar el código fuente

Si se elige la primera opción, debemos hacer una conexión al servidor ftp anónimo en la dirección *ftp.ncsa.uiuc.edu* y cambiarnos al directorio */Web/httpd/Unix/ncsa_httpd*. Veremos algo como lo siguiente:

```
ftp> cd /Web/httpd/Unix/ncsa_httpd/current
250-Please read the file README
250-  it was last modified on Thu Jun 22 19:28:09 1995 - 11 days ago
250 CWD command successful.
```

```
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 10712
drwxr-xr-x  2 19056   wsstaff    2048 Jun 22 19:10 .
drwxr-xr-x  7 12873   wheel      2048 Jun 22 19:29 ..
-rw-rw-r--  1 19056   wsstaff    3368 Jun 22 19:28 README
-rwxr-xr-x  1 19056   wsstaff  113507 Jun 23 12:04 httpd_aix3.2.5.Z
-rw-r--r--  1 19056   wsstaff  377073 Jun 23 12:04 httpd_aix3.2.5.tar.Z
-rwxr-xr-x  1 19056   wsstaff   85261 Jun 23 12:04 httpd_hpux9.0.5.Z
-rw-r--r--  1 19056   wsstaff  304644 Jun 23 12:04 httpd_hpux9.0.5.tar.Z
-rwxr-xr-x  1 19056   wsstaff  209641 Jun 23 12:04 httpd_irix4.0.5.Z
-rw-r--r--  1 19056   wsstaff  580755 Jun 23 12:04 httpd_irix4.0.5.tar.Z
-rwxr-xr-x  1 19056   wsstaff  108231 Jun 23 12:04 httpd_irix5.2.Z
-rw-r--r--  1 19056   wsstaff  333993 Jun 23 12:05 httpd_irix5.2.tar.Z
-rwxr-xr-x  1 19056   wsstaff   62597 Jun 23 12:05 httpd_linux.Z
-rw-r--r--  1 19056   wsstaff  694175 Jun 23 12:05 httpd_linux.tar.Z
-rwxr-xr-x  1 19056   wsstaff  133238 Jun 23 12:05 httpd_osf3.0.Z
-rw-r--r--  1 19056   wsstaff  379027 Jun 23 12:05 httpd_osf3.0.tar.Z
-rwxr-xr-x  1 19056   wsstaff   70743 Jun 23 12:05 httpd_solaris2.3.Z
-rw-r--r--  1 19056   wsstaff  303047 Jun 23 12:05 httpd_solaris2.3.tar.Z
-rwxr-xr-x  1 19056   wsstaff   70727 Jun 23 12:05 httpd_solaris2.4.Z
-rw-r--r--  1 19056   wsstaff  303213 Jun 23 12:05 httpd_solaris2.4.tar.Z
-rw-r--r--  1 19056   wsstaff  139070 Jun 23 12:05 httpd_source.tar.Z
-rwxr-xr-x  1 19056   wsstaff   72327 Jun 23 12:05 httpd_sunos4.1.3.Z
-rw-r--r--  1 19056   wsstaff  298136 Jun 23 12:05 httpd_sunos4.1.3.tar.Z
-rwxr-xr-x  1 19056   wsstaff  217343 Jun 23 12:05 httpd_ultrix4.0.Z
-rw-r--r--  1 19056   wsstaff  606639 Jun 23 12:05 httpd_ultrix4.0.tar.Z
226 Transfer complete.
ftp> quit
221 Goodbye.
```

En este caso, debemos obtener el archivo apropiado. Por ejemplo, para una SparcStation 5 con sistema operativo Solaris 2.3, debemos transferir en modo binario, el archivo *httpd_solaris2.3.tar.Z* de forma similar como transferimos el código precompilado de Mosaic. Nuevamente tenemos que utilizar el comando *uncompress* y *tar* de Unix.

A diferencia del código de Mosaic, el de http es más amigable para la compilación, por lo que el usuario no tendrá mayores problemas en esta tarea. Simplemente haga una llamada al programa *make* de Unix bajo el directorio /src y la

compilación se llevará a cabo generando un archivo binario llamado *httpd* que se encuentra bajo el mismo directorio.

Configuración del servidor httpd

Los archivos de configuración del servidor httpd, reflejan la forma en como se va a comportar dentro de nuestro sistema. Existen 3 archivos principales que el usuario debe editar.

El primero es el archivo llamado *conf/httpd.conf* que viene acompañado con el código fuente. Cada archivo de configuración se compone de directivas (una directiva por línea) y comentarios opcionales que comienzan con el signo #.

Ejemplo:

```
# Este es el archivo de configuracion principal. Aqui
# puedes indicar directivas en base a las caracteristicas deseadas
# en tu sistema. Vee el URL http://hoohoo.ncsa.uiuc.edu/ para instrucciones.
# No sigas las instrucciones sin entender que hacen
# Si tienes alguna duda, puedes obtener la documentacion al respecto

# ServerType. Esta directiva puede ser inetd o standalone.
ServerType standalone

# Port: El numero de puerto en que el servidor escuchara peticiones
# Deberas correrlo inicialmente como root.
Port 80

# Si deseas que httpd corra como un usuario y grupo diferente, debes cambiar las
# siguientes directivas.
# User/Group: El nombre (or #numero) del usuario/grupo
User http
Group staff

# ServerAdmin: Tu direccion electronica, donde se pueden reportar problemas del
#servidor
ServerAdmin edgar@tzetzal.dcaa.unam.mx
```

```
# ServerRoot: El directorio raiz del servidor, donde residen los archivos de
# configuracion, errores y auditoria.
ServerRoot /bases/http/

# ErrorLog: Nombre del archivo donde se almacenan errores del servidor
# El directorio logs se encuentra bajo raiz especificado con la directiva anterior
ErrorLog logs/http_error.log

# TransferLog: Nombre del archivo que registra los accesos a nuestro servidor
TransferLog logs/accesos.log

# PidFile: Este archivo contiene el pid con el que esta corriendo el servidor
PidFile logs/httpd.pid

# Archivo de configuracion de acceso global. Relativo a ServerRoot
AccessConfig conf/access.conf

# Archivo de configuracion de recursos:
ResourceConfig conf/srm.conf

# Archivo de configuracion de tipos, es decir, como maneja las extensiones
# de archivos.
TypesConfig conf/mime.types

# ServerName te permite colocar el nombre de tu maquina, el cual sera valido para
# los clientes de la Web
ServerName tzetzal.dcaa.unam.mx
```

Podemos observar que el archivo anterior es el de configuración general, no define permisos ni protege directorios. Por otra parte, este archivo corresponde a un servidor instalado en otra máquina llamada tzetzal.dcaa.unam.mx cuya arquitectura corresponde a una estación de trabajo SparcStation 10. Lo que sucede, es que en la SparcStation 10 se generan los documentos HTML correspondientes a todas las aplicaciones en otros proyectos que se desarrollan en la Coordinación Técnica de la DCAA. Es por ello que existen URL's residentes en esta estación de trabajo, y uno de ellos apunta hacia el "home page" del kiosco, el cual es un documento HTML que existe en la estación de trabajo del kiosco, donde también se instaló y configuró un servidor http de manera similar. En el Apéndice A, se muestran los archivos de configuración correspondientes al servidor http del Kiosco.

La ventaja de tener documentos HTML en otra estación de trabajo dentro de la red, es que se pueden tener varios Kioscos actuando como clientes accediendo a la misma aplicación. Si alguno falla, la aplicación puede ser accedida por los demás clientes de la red puesto que el servidor se encuentra en un lugar seguro como lo es un Centro de Cómputo. Recuerde que nuestra intención es que los Kioscos se instalen en lugares públicos.

El segundo archivo de configuración es el de recursos, llamado *conf/srm.conf*. También se compone de directivas como se muestran a continuación:

```
# Con este archivo, defines un directorio de documentos HTML para tu servidor http
# Puede ver la documentacion en el URL http://hoohoo.ncsa.uiuc.edu/docs/tutorials/
# DocumentRoot: Esta directiva especifica el directorio raiz de tus documentos html
# Por default, todas las peticiones son tomadas de este directorio.
DocumentRoot /www

# UserDir: El nombre del directorio donde residen los documentos html de los usuarios
# en general. Este se encuentra bajo /www.

UserDir html_docs

# DirectoryIndex: Si no se especifica el URL al hacer una conexion a este servidor, por
# default se tomara el que contiene esta directiva.

DirectoryIndex p-dgsca/DCAA.html

# Define iconos por default
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 define el icono para archivos que no se pueden desplegar como una imagen
DefaultIcon /icons/unknown.xbm

# AccessFileName: El nombre de archivo donde residen nombres y claves de usuarios
# cuando protegemos directorios
AccessFileName .htaccess
```

```
# DefaultType: es el tipo por default para documentos que el servidor va a manejar
DefaultType text/plain

# Alias: Simplemente define alias a directorios
Alias /icons/ /bases/http/icons/
Alias /www/ /

# ScriptAlias: Alias que define el nombre del directorio donde residen los scripts
# del usuario o los CGI's
ScriptAlias /cgi-bin/ /bases/http/cgi-bin/
```

El archivo de configuración para el acceso a ciertos directorios de nuestro servidor se llama *conf/access.conf*. Si se desea que una máquina o un conjunto de máquinas (en base a direcciones IP) sean las únicas que puedan acceder nuestros documentos HTML, se pueden especificar aquí.

```
# access.conf: Configuracion de acceso global al sistema
# La documentacion en linea se encuentra en el URL http://hoohoo.ncsa.uiuc.edu/
<Directory /bases/http>
Options Indexes FollowSymLinks
</Directory>

# Este directorio permite ser accedido por todos los clientes de la Web.
<Directory /www>

# La siguiente directiva puede ser "None", "All" o "FollowSymLinks" dependiendo de
# las caracteristicas de tu directorio. En nuestro caso, el servidor seguira ligas simbolicas
# bajo este directorio.
Options Indexes FollowSymLinks

# AllowOverride. Esta directiva controla permisos de acceso sobre algun directorio.
# Puede ser "None" (nadie entra) o "All" (todos pueden entrar).
AllowOverride All

# Controles que pueden usarse para dar permiso de acceso a ciertas maquinas.
<Limit GET>
order allow,deny
allow from all
</Limit>
</Directory>

# Tu puedes especificar otros directorios de tu sistema que sean accedidos
```

```
# Modificacion para la configuracion de tzetzal.
# Existen directorios que debes proteger por ejemplo: home, usr, etc y otros
<Directory /home>
AllowOverride None
Options None
</Directory>

<Directory /opt>
AllowOverride None
Options None
</Directory>

<Directory /usr>
AllowOverride None
Options None
</Directory>

<Directory /etc>
AllowOverride None
Options None
</Directory>

<Directory /tmp>
AllowOverride None
Options None
</Directory>

<Directory /www/sistemas>
AllowOverride None
Options None
</Directory>

<Directory /www/p-dgsca>
AllowOverride None
Options None
</Directory>

<Directory /www/presenta-beca>
AllowOverride None
Options None
</Directory>

<Directory /www/presenta-dgsca>
AllowOverride None
Options None
</Directory>
```

Una vez compilado los programas que generan el archivo *httpd* y modificado adecuadamente los archivos de configuración para nuestro sistema, debemos correr el programa para que quede como servidor. Esto se puede hacer por dos caminos dependiendo del valor de la directiva ServerType:

- Bajo los servicios de *inetd* (Internet Services Daemon)
- Standalone

Para correr nuestro servidor bajo los servicios de red (inetd), se debe editar el archivo */etc/services* y agregar la siguiente línea:

```
http        80/TCP
```

Usted puede sustituir el valor del puerto 80 por otro. Ahora hay que editar el archivo */etc/inetd.conf* agregando la siguiente línea:

```
http  stream  tcp  nowaitnobody  /bases/http/httpd httpd</PRE>
```

Donde la ruta */bases/http/httpd* puede cambiarse dependiendo del directorio donde resida el archivo binario *httpd*.

El siguiente paso es matar y levantar el proceso asociado a *inetd* dentro del sistema (para hacer válidos los nuevos cambios de los archivos de configuración) utilizando la siguiente instrucción:

```
% kill -HUP <pid_de_inetd>
```

Para saber el número de proceso asociado a *inetd*, puede hacer uso del comando *ps* y *grep* de Unix. Por ejemplo:

```
% ps -fea | grep "inetd"
```

. . . y eso sería todo, el servidor está corriendo.

Si se escoge correrlo como *Standalone*, simplemente se hace la llamada al archivo ejecutable con las opciones *-d* y *-f*. Ejemplo:

```
% /bases/http/httpd -d/bases/http -f/bases/http/conf/httpd.conf
```

donde *-d* es el nombre del directorio donde reside el archivo ejecutable *httpd* y *-f* es la ruta del archivo de configuración del cual ya hablamos (httpd.conf).

La forma más usual de correr el servidor es como *Standalone* puesto que, si se llegase a hacer alguna modificación en la configuración del mismo o simplemente darlo de baja, no tendríamos que bajar los servicios de red (en el caso de inetd) para hacerlo e interrumpir el trabajo de los demás usuarios.

A partir de la instalación del servidor (en cualquiera de las dos formas), podemos ofrecer a la comunidad de la Internet, el servicio de transferencia de hipertexto. Usuarios de otras máquinas pueden conectarse a la nuestra para recuperar información y visualizarla con programas como Mosaic.

5.3.3 Generación de documentos Hipertexto

En el capítulo 4 (sección 4.6) explicamos algunas de las etiquetas que forman parte de HTML versión 2.0, ahora pasaremos a su aplicación para generar los documentos en hipertexto necesarios para el sistema de información del Kiosco. El hipertexto nos ayuda a formar ligas entre un documento y otro, ya sea mediante texto o imágenes. A continuación presentamos un ejemplo de lo que sería el "Home Page" del sistema o la primera página de presentación del mismo.

```
<TITLE>  KIOSCO DE INFORMACION </TITLE>
<pre>
<h1>
  Universidad Nacional Aut&oacute;noma de M&eacute;xico
</h1>
</pre>
```

En este caso se está representando el título del documento en hipertexto y el primer encabezado.

```
<pre>
                                        <IMG SRC="/home/Mosaic/images/unamplac.3.gif">
</pre>
```

Aquí se está llamando a una imagen para que sea incluida al momento de ser visualizado el documento. En este caso, la figura es el escudo de la UNAM (Universidad Nacional Autónoma de México).

```
<pre>
<H2>
    Sea usted bienvenido a la red de kioscos de informaci&oacute;n
</H2>
Para continuar hacia abajo, deslizar la barra con el dedo.
<p>
<img src="http://xel-ha.dcaa.unam.mx/home/Mosaic/images/line.blue.gif">
</pre>
```

En este caso se está desplegando un letrero de bienvenida, con un tipo de letra más grande, que como ya se explicó anteriormente, se logra con las etiquetas <H2> y </H2>. Posteriormente se despliega una línea azul divisoria; esta línea (que a su vez es una figura reconocida con el nombre de archivo asociado *line.blue.gif*) se está mandando llamar desde otra máquina.

```
<h1>A trav&eacute;s de este kiosco usted podr&aacute; accesar a servicios tan diversos
como: </h1>
```

Después se despliega otro letrero, sólo que con un tipo de letra más grande que el representado por <H2>.

```
<p>
<h2>
<img src="http://xel-ha.dcaa.unam.mx/home/Mosaic/images/punto-rojo1.gif">
<a href="/home/kiosko/html/Dependencias.html">
Descripci&oacute;n de Dependencias y Escuelas de la UNAM</a><p>

<img src="http://xel-ha.dcaa.unam.mx/home/Mosaic/images/punto-rojo1.gif">
<a href="http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/Inicio2.html">
Inscripci&oacute;n a cursos</a><p>
</h2>
```

En esta parte se tienen dos ligas al mismo tiempo que se despliegan los letreros, el usuario sabe que están apuntando a otros documentos puesto que el texto está subrayado y en color azul. En este caso, se llama localmente al documento en HTML que se encuentra en la ruta:

/home/kiosko/html/Dependencias.html

Cuando el usuario toque la frase con el tacto, se desplegará el documento al que se está apuntando. Vale la pena recordar que el URL al que se hace referencia, se puede encontrar en alguna otra máquina de la misma red. En este caso, se hace de la siguiente forma:

http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/Inicio2.html

Donde tenemos la posibilidad de conectarnos a otro servidor con los mismos servicios de http, esa liga puede estar apuntando a un archivo de video, sonido o texto, de esta manera, podríamos crear ligas hacia servidores en cualquier parte del mundo. La liga que representa la inscripción a cursos, será explicada con mayor

detalle en el siguiente capítulo. Por ahora, simplemente hacemos referencia a un documento donde aplicaremos *formas* y CGI's para interactuar con Sybase como manejador de Base de datos.

```
<pre>
<img src="/home/Mosaic/images/line.blue.gif">
</pre>

<h3>
Para cualquier comentario o sugerencia, por favor comunicarse con los
<a href="/home/kiosko/html/Responsables.html">responsables</a> del sistema:
kioscos de informaci&oacute;n.<p> </h3>
```

Por último, se encuentra este apartado donde hacemos referencia a otro documento hipertexto que nos dá información acerca de los responsables del sistema.

El documento que hemos venido explicando, se podrá visualizar con *Mosaic*, solo que muchas veces por la dimensión del mismo, el usuario tendrá la necesidad de deslizar la barra de corrimiento la cual se encuentra en su extremo derecho, para ir recorriendo la extensión del documento sin mayor problema.

Todas las ligas que se estén realizando con documentos hipertexto, se podrán identificar principalmente porque están subrayadas, y a su vez serán la conexión hacia otros documentos que estén asociados a las mismas.

Los documentos hipertexto tienen la posibilidad de ser editados en cualquier editor antes de ser llamados por Mosaic, de esta forma, la elaboración de este tipo de documentos se vuelve más versátil, y su manejo por lo tanto se vuelve más sencillo.

Para la generación del resto de los documentos de nuestro sistema de información, utilizamos la consola de la estación de trabajo y una terminal gráfica X (X terminal). Es mucho más cómodo trabajar en ventanas porque se agiliza el diseño de las páginas. Por otro lado, el sistema Unix provee 2 editores (**ed** y **vi**), sin

embargo, para aplicaciones de este tipo, utilizamos uno llamado **pico** (que se encuentra disponible en la Red Internet y puede ser encontrado con ayuda de un servidor archie) que es mucho más amigable para los usuarios.

Un fragmento de este documento en hipertexto, desde Mosaic se vería tal como lo muestra la Figura 5.15.

El contenido del archivo /home/kiosko/html/Dependencias.html se muestra a continuación y un fragmento del mismo, visualizdo desde Mosaic, se presenta en la Figura 5.16.

```
<title>
Dependencias y escuelas de la UNAM.
</title>

<pre>
<h1>
Dependencias y Escuelas de la UNAM.
</h1>
<img src="/home/Mosaic/images/rectoria2.gif">


        Para continuar deslizar la barra con el dedo.
<p>
<p>
<img src="/home/Mosaic/images/line.blue.gif">

                                <img src="/home/Mosaic/images/line.blue.gif">
<p>
<p>
</pre>

<h2>
<img src="/home/Mosaic/images/punto-rojo1.gif">
<a href="http://www.super.unam.mx/dgsca">
Direcci&oacute;n General de Servicios de C&oacute;mputo Acad&eacute;mico</a><p>

<img src="/home/Mosaic/images/punto-rojo1.gif">
<a href="http://tzetzal.dcaa.unam.mx/presenta-dgsca/DCAA.html">
Direcci&oacute;n de C&oacute;mputo para la Administraci&oacute;n
Acad&eacute;mica</a><p>
</h2>
```

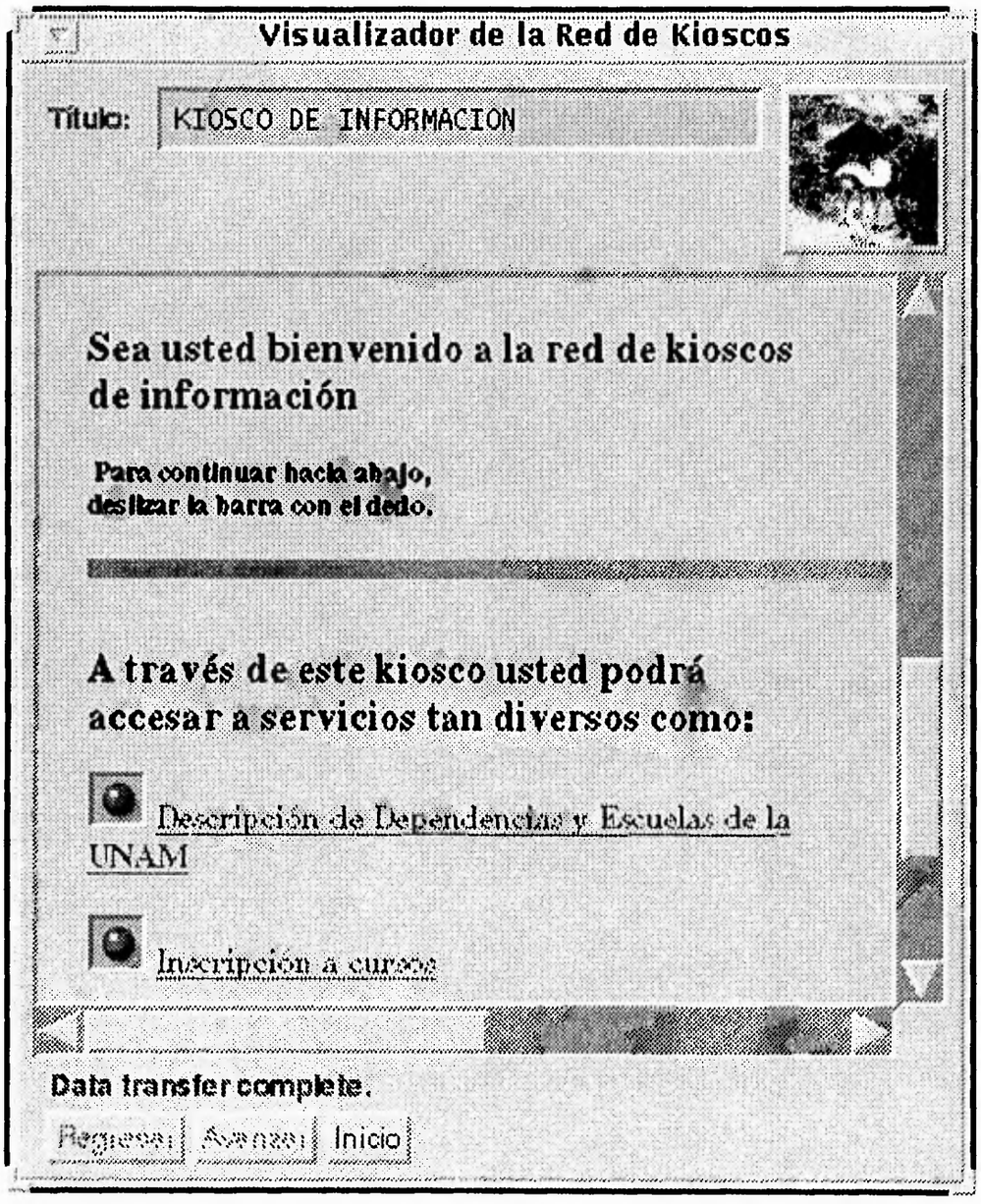


Figura 5.15 "Home Page" del kiosco

```
<pre>
<p>
<p>
<img src="/home/Mosaic/images/line.blue.gif">
                              <img src="/home/Mosaic/images/line.blue.gif">
<p></pre>
```

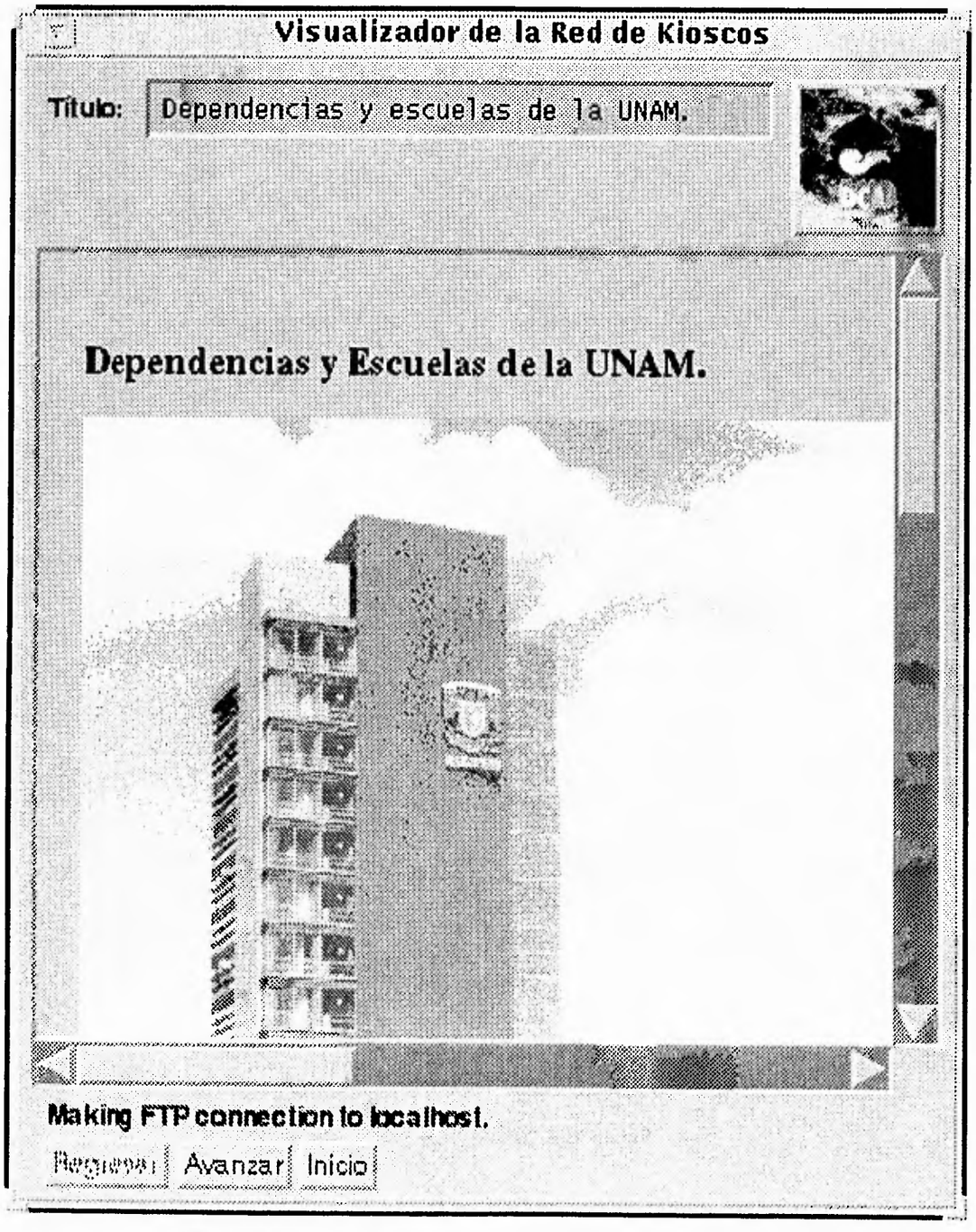


Figura 5.16 Representación de la página de Escuelas y Dependencias

El lector puede notar que existen las siguientes ligas:

```
<a href="http://www.super.unam.mx/dgsca">
Direcci&oacute;n General de Servicios de C&oacute;mputo Acad&eacute;mico</a><p>
```

```
<a href="http://tzetzal.dcaa.unam.mx/presenta-dgsca/DCAA.html">
Direcci&oacute;n de C&oacute;mputo para la Administraci&oacute;n
Acad&eacute;mica
```

La primera, apunta al "home page" de la DGSCA, cuando el usuario toque la frase: Dirección General de Servicios de Cómputo Académico **automáticamente Mosaic hará la conexión al servidor www.super.unam.mx sin que el usuario se percate de ello; él, no logrará diferenciar a simple vista entre las transacciones locales y remotas, se verá algo como lo muestra la figura 5.17.**

De la misma forma, cuando toque la frase: Dirección de Cómputo para la Administración Acadéamica **se hará una conexión a otro servidor cuyo "home page" se muestra a continuación:**

```
<title>Presentacion de la DCAA</title>
<pre>
<h1>
 Universidad Nacional Aut&oacute;noma de M&eacute;xico
</h1>

<img align=middle src="../images/DCAA.gif">

<h2>
  Direcci&oacute;n General de Servicios de C&oacute;mputo Acad&eacute;mico

 Direcci&oacute;n de C&oacute;mputo para la Administraci&oacute;n Acad&eacute;mica
</h2>

<HR>


<h1>
            Bienvenido a la
      Direcci&oacute;n de C&oacute;mputo para la
        Administraci&oacute;n Acad&eacute;mica
                                        (D C A A)
</h1>
```

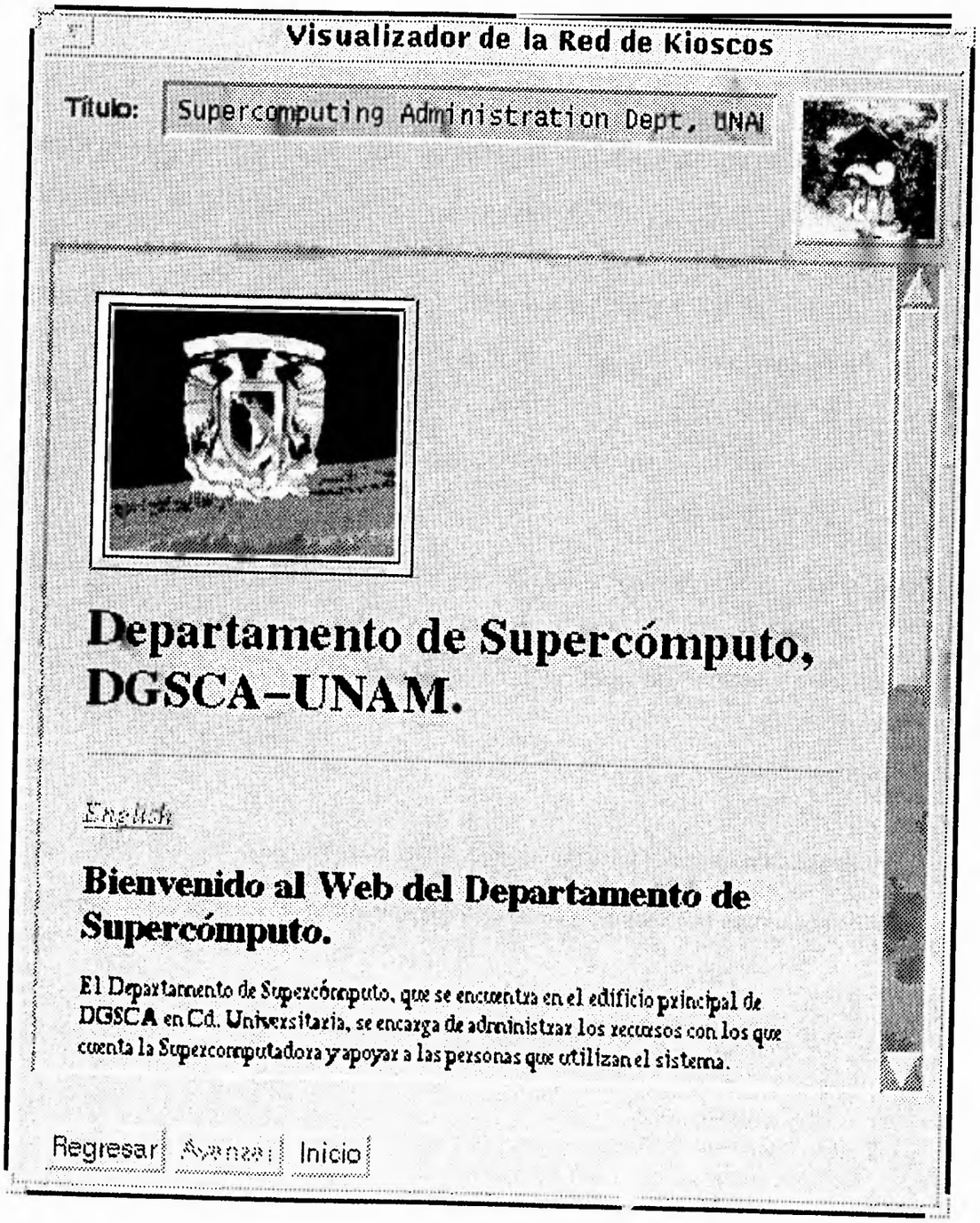


Figura 5.17 "home-page" de la DGSCA

```
<h3>
```
La Dirección de Cómputo para la Administración Académica (DCAA) proporciona servicios de cómputo y desarrollo de sistemas de información para la administración académica de las dependencias de la Universidad Nacional Autónoma de México, así como de las instituciones externas con las que se establecen convenios; a la vez que se supervisa el cumplimiento de las normas y procedimientos relativos a los servicios que se prestan, a fin de lograr un uso racional de los recursos de cómputo con que se cuenta.

```
<a
href="http://tzetzal.dcaa.unam.mx/sonidos/dcaa.act.2.au"><img
src="../images/sonidito2.gif"></a>
```

La Dirección de Cómputo para la Administración Académica es una dependencia fundamentalmente de servicios, los cuales son proporcionados a través de tiempo de máquina, aplicaciones en equipos especiales, desarrollo de sistemas de información y asesorías. Cuenta con un componente académico muy importante, ya que en gran parte sus actividades están dedicadas a la formación de recursos humanos y la difusión de la cultura informática.

```
<a
href="http://tzetzal.dcaa.unam.mx/sonidos/dcaa.act.beca.au"><img
src="../images/sonidito2.gif"></a>
```

Estos dos tipos de acciones están dirigidas, principalmente, a las Dependencias Usuarias y a un grupo sobresaliente de Becarios de la UNAM, que integran un <a
href="http://tzetzal.dcaa.unam.mx/presenta-beca/Inicio.html">proyecto de Becas que la DCAA viene desarrollando con mucho éxito desde hace varios años.

Para atender estas actividades y demás requerimientos de cómputo que las dependencias de la UNAM reclaman en el desempeño de sus funciones de administración académica, la DCAA est&aacute organizada de la siguiente manera:

```
<img src="../images/orgdcaa.gif">
<a
href="http://tzetzal.dcaa.unam.mx/presenta-dgsca/SFRH.html"><img
align=middle src="../images/rechum2.gif"></a>
<a
href="http://tzetzal.dcaa.unam.mx/presenta-dgsca/SS.html"><img align=middle
src="../images/sist.gif"></a>
```

```
<a
href="http://tzetzal.dcaa.unam.mx/presenta-dgsca/CT.html"><img align=middle
src="../images/prospc.gif"></a>                    <a
href="http://tzetzal.dcaa.unam.mx/presenta-dgsca/CO.html"><img align=middle
src="../images/a12.gif"></a>                    <a
href="http://tzetzal.dcaa.unam.mx/presenta-dgsca/CA.html"><img align=middle
src="../images/admin.gif"></a>                    <a
href="http://tzetzal.dcaa.unam.mx/presenta-dgsca/DPE.html"><img align=middle
src="../images/interope.2.gif"></a>
<img src="../images/tsfrh.gif">        <img
src="../images/tss.gif">        <img
src="../images/tctri.gif">              <img
src="../images/tco.gif">          <img
src="../images/tsa.gif">            <img
src="../images/tdpe.gif">
```

Teniendo como personal a 42 administrativos, 13 técnicos académicos, 50 becarios, varios prestadores de servicio social, 4 jefes de área, 3 coordinadores, 2 subdirectoras y una directora, haciendo un total de más de 100 personas.

```
<HR>

<pre>
                                        <img align=middle src="../images/act.if">

                                            Directora de la DCAA <p>
</pre><HR>
```

Visto desde Mosaic, el documento anterior se observará como lo muestra la Figura 5.18.

El procedimiento para generar las demás páginas de presentación es similar a los ejemplos expuestos anteriormente. A partir de aquí, generamos y creamos los restantes documentos en hipertexto, los cuales utilizaron la misma técnica de elaboración, pudiendo inclusive incluir en los mismos, ligas hacia archivos de sonido, imágenes o video que realizan enlaces hacia otras páginas de Mosaic, no importando si éstas residen en forma local o remota, siempre y cuando se especifique la ruta de manera adecuada y el servidor esté activo en el momento que

se invoque su servicio. De esta manera, podemos compartir nuestras páginas y archivos con otros servidores de la red.

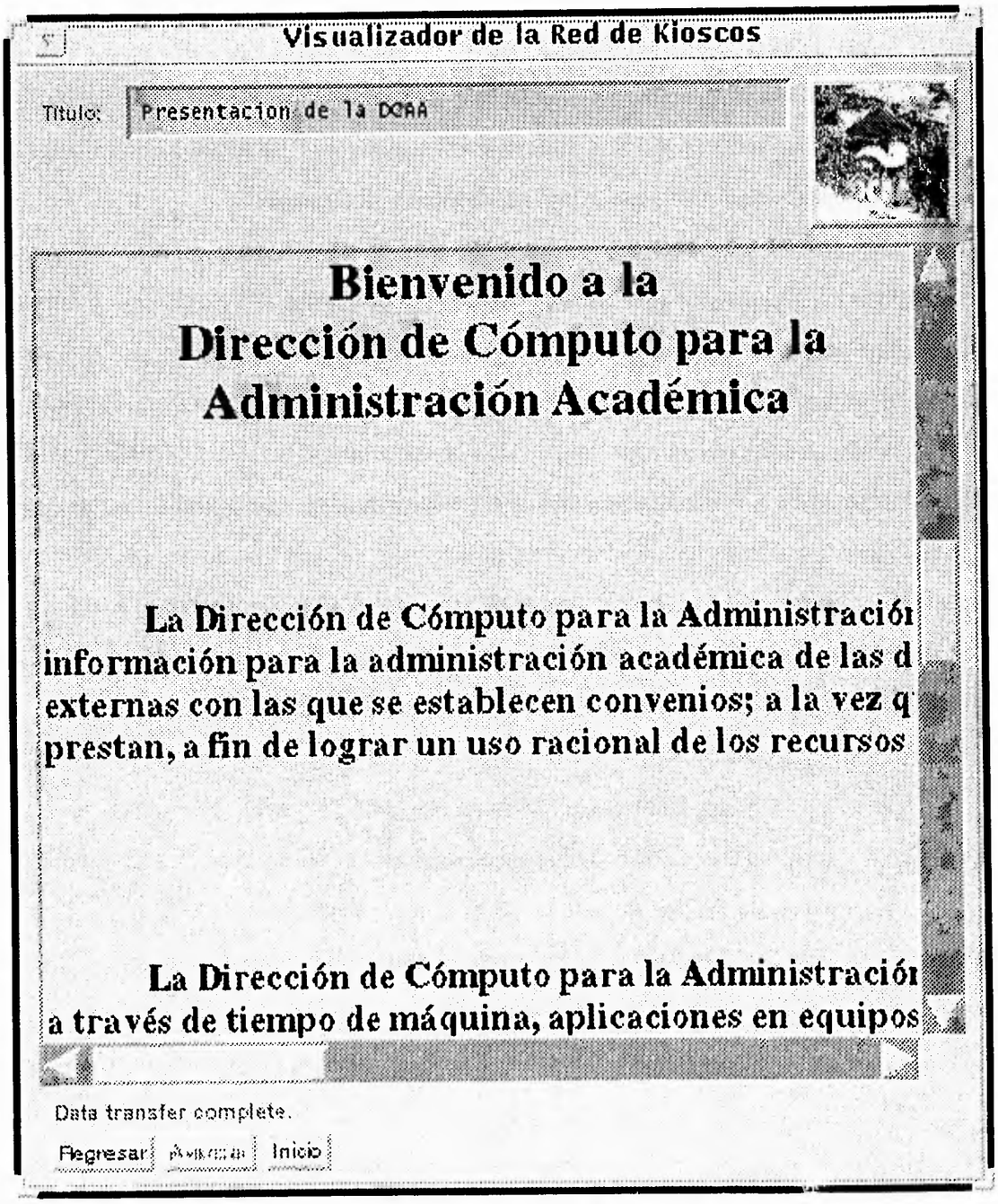


Figura 5.18 "home page" de la DCAA

5.4 Integración del Kiosco

Los documentos en hipertexto generados para nuestra aplicación en multimedia utilizando Mosaic como Interfaz de Usuario Gráfica, pueden compartirse con otros visualizadores gracias al servidor http que fue instalado, el cual permite transferir hipertexto a otras máquinas dentro de la Red Internet.

Hasta este momento, hemos generado los documentos en HTML incluyendo las imágenes, sonido y video. Mosaic fue configurado para el tipo de aplicación que necesitamos. Nos resta probarlo con la pantalla *touchscreen* y por último, instalar la estación de trabajo en un mueble. De esta forma, nuestro Kiosco estará completo.

Cabe mencionar que la estación de trabajo donde se desarrolló el proyecto, estará conectada a la RedUnam, con el objetivo primero, que nuestro "home page" pueda ser accedido desde otras máquinas de la red (incluso computadoras personales) y acceder a otras páginas de presentación de la Web en la misma, y segundo, en el siguiente capítulo veremos como poder interactuar con una Base de Datos remota que se localiza en otra máquina para aplicarlo a un sistema de inscripciones a cursos.

Esta es la etapa final de la primera fase del proyecto. El *touchscreen* que utilizamos, forma parte de uno de los productos que la Compañía Cromasoft S.A de C.V. distribuye, fue adquirido por la DCAA hace algunos meses. El proveedor se encargó de instalar el software que permite configurarlo y ponerlo en uso. Una vez hecho esto, nuestro trabajo fue la calibración del mismo así como definir ciertos parámetros que nos permitieran modificar su funcionamiento. Por ejemplo, Mosaic ocupa toda la pantalla del monitor, pero sigue siendo una ventana. Si un usuario logra tocar los bordes de la misma, puede moverla logrando con ello, cerrar la ventana y desechar nuestra aplicación. Esto es algo que también debemos evitar.

Para ello, aprovechamos una ventaja que el software del *touchscreen* ofrece: activar una región de la pantalla para el usuario. De esta forma, evitamos que el puntero en la pantalla se mueva a los bordes de la ventana.

El software del *touchscreen* reside bajo el directorio */usr/lib/tsi*. Existe un archivo binario ejecutable llamado *setup*, el cual nos permite definir ciertos parámetros referentes a la calibración, sensibilidad, etc. Cuando se invoca, aparece una ventana como se muestra en la Figura 5.19.

Donde cada opción se refiere como sigue:

Jitter Control:

Este control puede variar sus valores entre 0 y 15. Su valor representa la distancia en pixeles a la cual el nuevo punto deberá moverse de su posición actual a la que se le indique con el puntero. Esta característica elimina el movimiento del puntero de un contacto estacionario. El valor es automáticamente devuelto a cero si el botón de *Reset Defaults* es presionado.

Press Threshold:

Este control puede variar sus valores entre 1 y 15. Su valor se refiere al nivel de sensibilidad de presión a la cual se quiere que el touchscreen responda, siendo 15 el máximo valor. Cuando la presión aplicada al touchscreen excede a la especificada por el *press threshold*, se generará el equivalente a un click del *ratón*. De lo contrario, se producirá un movimiento del puntero del mismo en la pantalla. El valor que se le especifique a este botón de control, deberá de ser por lo menos uno mayor que el que se especifique al botón de *Release Threshold*. *Press Threshold* tendrá automáticamente un valor de 5 si el botón de *Reset Defaults* es presionado.

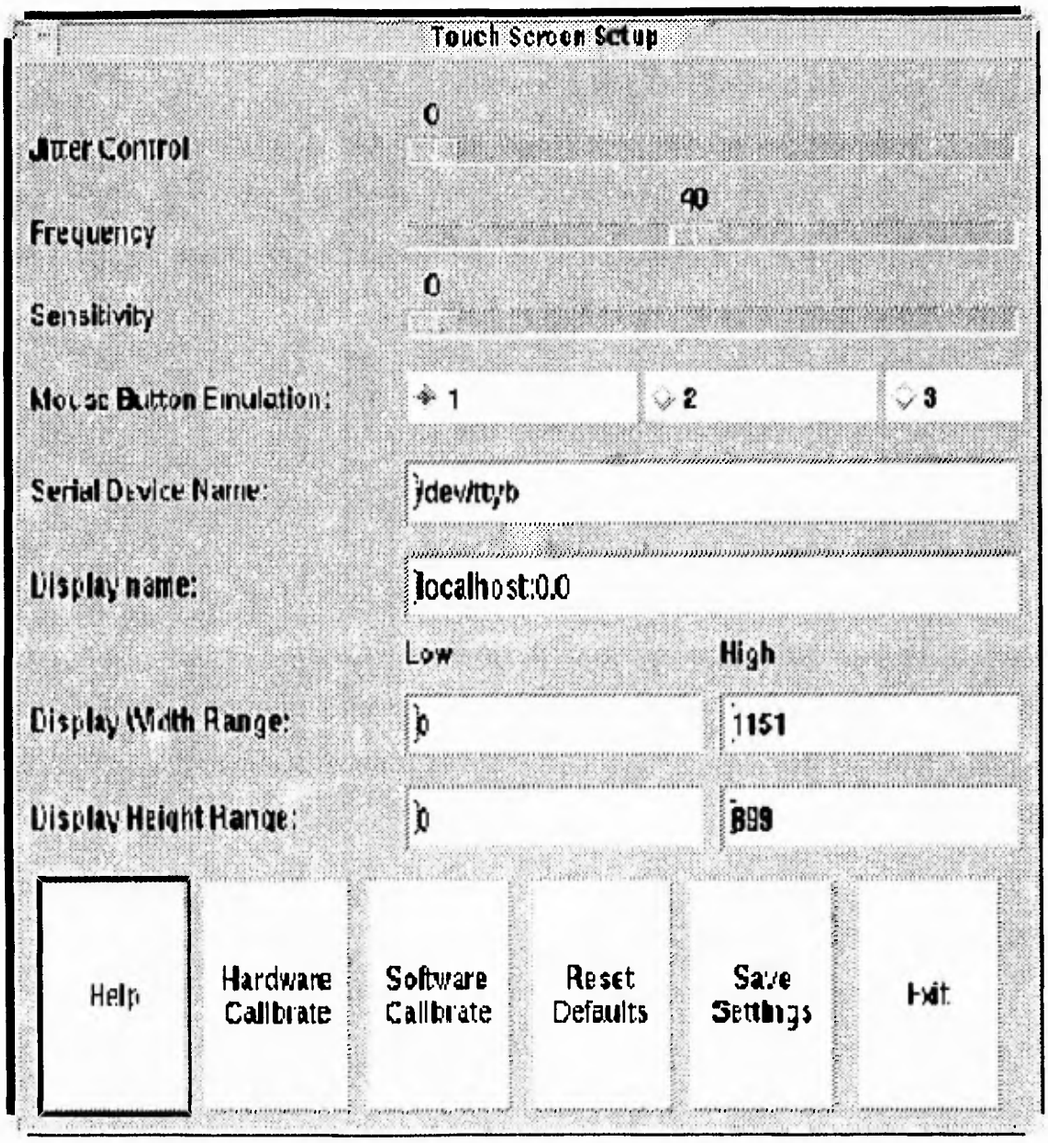


Figura 5.19 Ventana de configuración del touchscreen

Release Threshold:

Este control puede variar sus valores entre 0 y 14. Después de que una presión sobre la pantalla fue aplicada, este botón nos va a ayudar a detectar si la

presión aplicada está por debajo de la especificada por el valor de *press threshold*, es por eso que el valor de este botón deberá ser por lo menos un valor menor que el de *press threshold*. El botón *Realease Threshold* tendrá automáticamente el valor de 2 si el botón de *Reset Defaults* es presionado.

Mouse Emulation:

Para emular el botón 1 del *ratón* seleccionar 1. Para emular el botón del *ratón* seleccionar 2. Para emular el *botón* 3 del ratón seleccionar 3.

Serial Device Name:

Este campo representa el nombre del dispositivo del puerto serial al cual el controlador del *touchscreen* está físicamente conectado. Usualmente el nombre del dispositivo será algo como */dev/tty??*. Este valor será colocado si el botón de *Reset Defaults* es presionado.

Display Name:

Este campo representa el nombre del servidor al cual el control del puntero será aplicado. Normalmente este campo está representado con el nombre de la estación de trabajo o la terminal a la cual el *touchscreen* está conectado. Este valor aparecerá automáticamente si el botón de *reset* es presionado.

Display Width Range (Low):

Este campo representa el valor más bajo de pixel que puede tomar el puntero. Usualmente es cero. Este valor se ajustará automáticamente a 0 si el botón de *reset defaults* es presionado.

Display Width Range (High):

Este campo representa el máximo más alto de pixel que puede tomar el puntero. Usualmente su valor es uno menor que el máximo del tamaño de la pantalla.

Help Button:

Desplegará estas instrucciones en inglés.

Software Calibrate Button:

El botón de calibración de software desplegará una pantalla con esquinas la cual mediante flechas nos ayudará a seleccionar el área donde el *touchscreen* vaya a responder a las especificaciones, misma que no se tomará en cuenta si se presiona el botón de *Cancel*.

Reset Defaults Button:

Este botón anulará todas las entradas predeterminadas en la configuración, las cuales no tomarán efecto hasta que hayan sido salvadas y aplicadas previamente.

Save Settings Button:

Este botón salvará y aplicará la presente configuración en un archivo llamado *cap.config*.

Exit Button:

Sale del programa de configuración.

En realidad, el uso de este software básico, no es nada difícil. Si se cometen errores, se repite la operación hasta que el *touchscreen* quede configurado como se

desea. Note que hay una sección donde podemos definir una región de la pantalla válida para el touchscreen.

Después, se corre el programa llamado *xtouch*, el cual activa la pantalla para ser manipulada con el tacto. Ahora, estamos listos para comenzar las pruebas con Mosaic.

Primero, se debe correr el programa *xtouch* (que activa la pantalla), luego se corre Mosaic, y de ahí en adelante, podemos utilizar el tacto para manipular el ambiente gráfico. Cada frase subrayada y en color azul, representa una liga hacia otro documento. El usuario, puede tocarlas con los dedos haciendo el equivalente al click del *ratón*. De esta forma, puede navegar en Mosaic sin enterarse que la información que visualiza, puede encontrarse localmente (en el kiosco) o en alguna otra máquina dentro de la Red.

El mueble

Por último, nos resta diseñar el mueble (ver Figura 5.20) que contendrá a la estación de trabajo. Este diseño lo realizó el Sr. Víctor Javier Arrieta Rosas (también labora en la Coordinación Técnica en el área de Difusión). Puede notar que el mueble tiene un teclado con el objeto de que la aplicación en el kiosco sea interactiva con el usuario. En realidad, no se siguió algún criterio en este diseño, únicamente se contempló que fuera lo más atractivo posible.

Con esto terminamos el prototipo de nuestro Kiosco. El mueble está hecho de madera y resta encontrar un lugar adecuado donde debe operar. Inicialmente, estará instalado en la Dirección de Cómputo para la Administración Académica, ofreciendo el servicio de información referente a esta dependencia. Por ejemplo, calendario de cursos, departamentos que existen, proyectos desarrollados y posteriormente el servicio de Inscripción a Cursos.

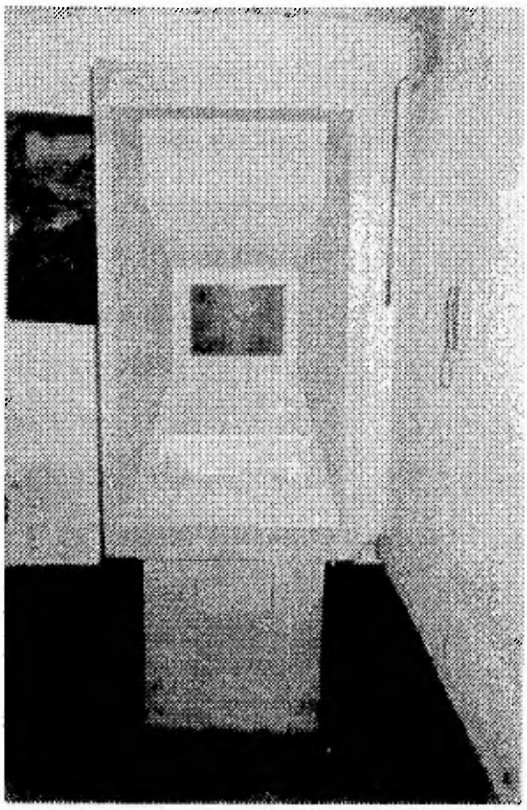


Figura 5.20 El mueble

5.5 Pruebas realizadas

Las pruebas a nuestro sistema son una parte muy importante para garantizar el buen funcionamiento del mismo. El objetivo que se pretende se puede resumir en 3 puntos:

1. La prueba es un proceso de ejecución de un programa con la intención de descubrir algún error.

2. Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
3. Una prueba tiene éxito si se descubre un error no detectado hasta entonces.

Las pruebas a realizar se pueden dividir en dos:

1. Una prueba a los documentos en hipertexto
2. Otra al servidor http

Primero, Mosaic puede operar en forma local, no es necesario estar conectado a la Red para poder generar documentos en hipertexto y mucho menos, tener instalado el servidor http. Se pueden realizar pruebas generando archivos conteniendo ligas hacia otros (también locales) especificando el URL y la instrucción en HTML de la siguiente forma:

```
<a href="file://localhost/directorio/archivo.html"> Apunta a otro archivo </a>
```

Obviamente, cuando Mosaic busque el archivo especificado, lo hará en la máquina local. Si lo encuentra, lo despliega, de lo contrario enviará un mensaje de error indicando que no existe. Note que el URL no hace referencia a un servidor http, simplemente se indica como un archivo local tal como se mostró en el ejemplo anterior.

Ahora, si se desea crear un "home page" y compartirlo con la comunidad de la Web, es necesario instalar el servidor http. Y en este caso, todas las ligas deberán apuntar a un URL de la siguiente forma:

```
<a href="http://tzetzal.dcaa.unam.mx/directorio/archivo.html"> Apunta a otro documento </a>
```

Pareciera que eso es todo, sin embargo, no es así. Una vez que generamos las páginas de presentación necesarias para el Kiosco, se comete un error muy común. En el URL se debe especificar el nombre completo de la máquina (se puede utilizar la dirección IP también), en ocasiones se omite el dominio. Por ejemplo:

```
<a href="http://tzetzal/directorio/archivo.html"> Apunta a otro documento </a>
```

Donde Mosaic no tendrá problema alguno al recuperar el documento si la máquina donde está instalado (llamada "tzetzal") pertenece al mismo dominio. Pero, si algún visualizador en alguna otra máquina tratara de hacer lo mismo y no pertenezca al dominio, tendría problemas puesto que no encontraría a "tzetzal". Para ello, es necesario utilizar nombres completos de los servidores al momento de especificar los URL's. Este tipo de errores fueron los más comunes que se encontraron en la etapa de pruebas. Donde, solicitábamos a alguien (por ejemplo DGSCA y CECAFI de la Facultad de Ingeniería) que tratara de acceder a nuestro "home page" desde Mosaic, los errores encontrados nos los notificaban y realizábamos las correcciones necesarias. En general, para cada documento HTML, fueron verificadas las ligas que apuntan hacia otros documentos y de ser necesario, corregir los URL's.

Por otra parte, Mosaic no es capaz de detectar errores en un documento en hipertexto. Si olvidé cerrar una etiqueta, Mosaic no lo va a notificar, por lo que hay que tener cuidado con esos detalles porque nos pueden meter en problemas. El simple hecho de no cerrarla puede causar que todo el texto restante aparezca de un solo tipo de letra (si se olvidó cerrar una etiqueta de encabezado por ejemplo) aunque las demás etiquetas se especifiquen correctamente.

El siguiente paso fueron las pruebas realizadas al servidor http. Aquí, fue necesario determinar los niveles de seguridad de nuestro servidor instalado.

En la sección 5.3.2 ya habíamos mencionado algo acerca de esto, nos referimos a los archivos de configuración de nuestro servidor. Cuando se instala por primera vez en alguna máquina, no se toman en cuenta que los servicios del mismo van más allá de la simple transferencia de hipertexto. Por ejemplo, yo podría acceder a directorios importantes del sistema y transferir información sin que el administrador se de cuenta. Recuerde cuando hablamos del archivo llamado *conf/access.conf*, en él, yo protegía ciertos directorios para no permitir el acceso a los clientes de la Web y de esta forma evitar que pudieran entrar a los mismos y ver la información del sistema y de los usuarios. Entre los directorios que se muestran en ese archivo se encuentran:

/home	Directorio de trabajo de los usuarios
/etc	Residen archivos de configuración del sistema
/opt	Almacena los programas de aplicación tales como manejadores de bases de datos, etc.

Si no se tienen protegidos los directorios adecuadamente, algún usuario en otra máquina dentro de la red y usando Mosaic, puede ver la información que contienen, representando con ello un grave problema de seguridad.

Puede observar que el archivo de configuración *access.conf* descrito en la sección 5.3.2 tiene una directiva llamada **DocumentRoot /www**. Aquí, se define un directorio a partir del cual, los clientes como Mosaic deben buscar sus archivos. Por ejemplo, si yo especifico la siguiente liga dentro en un archivo en HTML:

```
<a href="http://tzetzal.dcaa.unam.mx/pruebas/home.html"> Prueba 2 </a>
```

Cuando Mosaic busque el archivo "/pruebas/home.html", realmente lo va a hacer a partir de /www, esto es, el servidor *http* va a presentarle a Mosaic el directorio raíz como /**www** y no al directorio /. De lo contrario, puede entrar a todos los directorios y archivos bajo / (raíz), recuerde que el Sistema Operativo Unix tiene una organización arborescente para los archivos, donde / es la raíz y a partir de él, se desprenden los demás directorios. Por lo tanto, el visualizador no podrá acceder a otro directorio o archivo que no se encuentre bajo /www.

Al mismo tiempo, decidimos que todos los documentos en HTML generados para páginas de presentación, deben residir en el directorio /www cuyo contenido será únicamente archivos tipo imagen, sonido, video e hipertexto, controlando así la seguridad de nuestro sistema.

NetScape

Mosaic es actualmente el examinador de documentos de la Web más popular. Se ha convertido en un estándar para el manejo y recuperación de la información. En el año de 1994, se liberó la primera versión, se llama **NetScape**. Este programa, al ser un producto del mercado, tiene un costo y como tal, no se cuenta con el código fuente en la Red Internet. Sin embargo, en los últimos meses se ha difundido en todo el mundo abarcando incluso las computadoras personales, existe una versión de dominio público disponible en la Red Internet.

Como características relevantes destacan: una eficiencia mayor que Mosaic y puede interpretar HTML+ (o HTML plus). El lenguaje HTML+ es una variante del HTML versión 2.0, incluye un mayor número de cambios que se pueden hacer en un documento en hipertexto. Por ejemplo, se puede ajustar el tamaño de una imagen acorde al tamaño de la ventana que ocupa NetScape en la pantalla, existe un mayor

número de tamaños de letras, interpreta imágenes tipo JPEG, entre otras cosas. Todo esto, actualmente Mosaic no lo soporta.

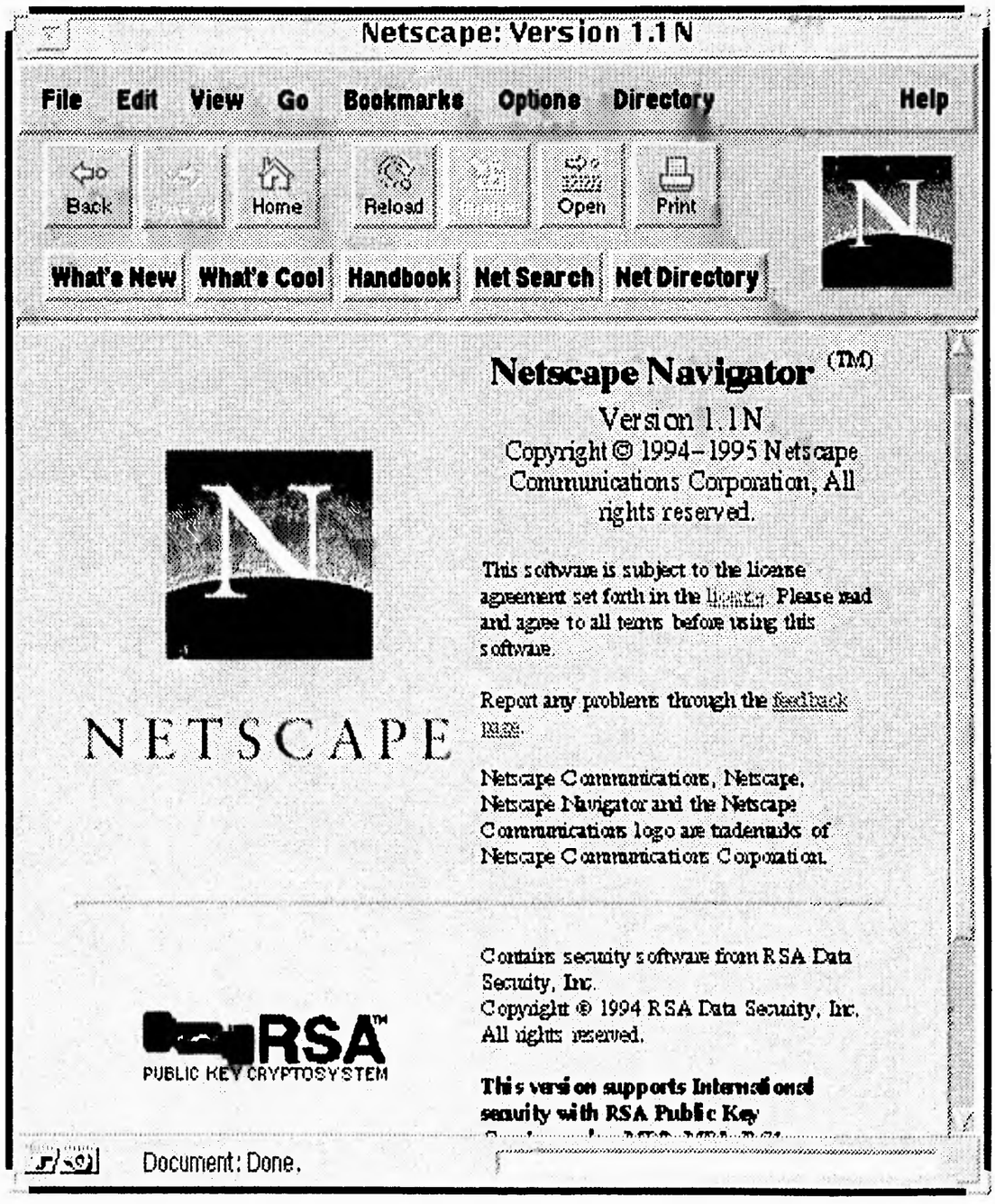


Figura 5.21 Netscape

Por último, los documentos generados en HTML para el sistema de información fueron probados desde NetScape (ver Figura 5.21) sin encontrar problema alguno. Únicamente observamos el comportamiento del sistema, puesto que no fue necesario realizar algún cambio en la configuración del servidor http. Cuando se corre NetScape, la ventana es muy parecida a la de Mosaic y simplemente verificamos el buen funcionamiento del sistema.

Capítulo 6

Caso de Aplicación: Sistema de Inscripciones a cursos de la DCAA

En el capítulo anterior, describimos el resultado del desarrollo de nuestro sistema. Hasta este momento, únicamente hemos incluido los elementos básicos de Mosaic en el *Kiosco de Información*. Sin embargo, el proyecto no termina ahí. Es cierto que en estos momentos tenemos una aplicación en multimedia, el usuario ya puede llegar al kiosco, comenzar a navegar en el ambiente gráfico y visualizar información con ayuda de la pantalla touchscreen.

Ahora, hemos pensado en un caso de aplicación orientado a la solución de un problema real. Queremos mostrar que con Mosaic no sólo podemos realizar multimedia (que son los casos más prácticos), ya que tenemos la posibilidad de desarrollar interfaces para interactuar con bases de datos importantes que se encuentran en algunas máquinas dentro de la red Internet.

Este capítulo, presenta la segunda fase del proyecto que presentamos como tema de tesis, el *Kiosco de Información* originalmente no contempla un sistema de inscripciones. Sin embargo, cuando terminamos la primera fase, nos dimos cuenta que nuestro producto sería aplicable a la solución de diversos problemas tal como lo discutiremos en las Perspectivas de Desarrollo (Capítulo 8).

Hasta ese momento, sabíamos de la existencia de *formas* en HTML, aunque pensar en un sistema de inscripciones completo manejando archivos (que son los casos más aplicables de CGI's), era una idea aún sin futuro en el Kiosco. Sin embargo, no tardó en surgir la propuesta de la posible aplicación de Bases de Datos en el sistema (a través de clientes y CGI's), así que las primeras pruebas de consulta de información a una Base de Datos en Sybase desde Mosaic, se realizaron en febrero de 1995 con gran éxito. En una semana, la idea se concretó e inmediatamente comenzamos a trabajar en el proyecto que dio como resultado el sistema que presentamos en este capítulo.

6.1 Introducción

Es indiscutible la importancia que los manejadores de bases de datos han adquirido en los últimos años, los cuales nos permiten manipular grandes cantidades de información en forma confiable. Con ayuda de ellos, se han implantado sistemas de información orientados a solucionar diversos problemas. Como un ejemplo, tenemos el sistema de inscripciones de los alumnos de la Facultad de Ingeniería.

El proceso de inscripción a cursos representa un gran problema para toda dependencia educativa, en algunos casos se tienen sistemas eficientes que automatizan las tareas y procesos de inscripción. En otros, se continua con el llenado de formas manualmente y no se tiene el control y metodología adecuados para garantizar la integridad y seguridad de la información. La Dirección de Cómputo para la Administración Académica no es la excepción, en los últimos meses, esta dependencia ha incrementado su labor docente y con ella, las necesidades de automatizar las tareas del proceso de inscripciones a cursos cada semestre.

Actualmente, la DCAA no cuenta con un sistema de información para administrar las inscripciones de los alumnos, los procedimientos se llevan a cabo en forma manual y como consecuencia, se llegan a cometer errores que podrían evitarse con un adecuado sistema que automatice esos procedimientos.

Para empezar, tenemos por un lado al Kiosco de información a partir del cual, el alumno se podría inscribir. Como será un servicio permanente, no importa la hora y fecha en que lo haga. Por otro lado, tenemos la versión comercializada de Mosaic (NetScape), la cual opera en una computadora personal con Windows de Microsoft y donde el administrador del sistema, podría programar el calendario de cursos y en general, realizar las tareas de administración. Utilizando la versión 1.1 de NetScape, el administrador estaría accediendo a una Base de datos que se encuentran en una estación de trabajo SparcStation 10. En las siguientes secciones discutiremos con más detalle las características que va a tener el sistema, por ahora vamos a comentar la operación del mismo a grandes rasgos.

Nuestra propuesta la podemos resumir en dos puntos:

- Primero, el alumno llenará su solicitud de inscripción a algún curso desde el Kiosco, esto se logra usando *formas* en HTML que interpreta Mosaic. Los datos capturados por el usuario se envían a una base de datos la cual se encuentra en otra máquina en alguna parte de la red.

- Segundo, el administrador del sistema puede controlar los procedimientos de inscripciones desde una PC inclusive, utilizando Netscape como *front end*. Puede manipular la información que el alumno proporcionó en el punto anterior.

La DCAA a pesar de ser una dependencia perteneciente a la Dirección General de Servicios de Cómputo Académico (DGSCA), cuenta con aulas y equipo de cómputo para impartir cursos. Actualmente los usuarios se inscriben directamente en la DGSCA, aunque algunos cursos estén programados para impartirse en la DCAA.

6.1.1 Importancia de las Bases de Datos

En el desarrollo de la ingeniería de programación fueron incorporándose nuevos modelos para el manejo de la estructura de los datos y el desarrollo de aplicaciones de tal forma que conforme el tiempo transcurre, los programas y archivos parecen no ser la mejor técnica para obtener resultados, cada vez se busca una respuesta más rápida y con ella, un mejor control en el manejo de la información. Los sistemas basados en los esquemas de Bases de Datos ofrecen grandes ventajas para el manejo sencillo de la información, garantizando la integridad y seguridad de la misma, así como permitir que sea compartida. Los manejadores de archivos tienden a tener inconsistencia en los datos, dificultad para el acceso a los mismos, aislamiento y se pueden tener usuarios múltiples sin control, sin olvidar los problemas de seguridad. Todos estos problemas pueden evitarse con un manejador de Bases de Datos, que en general, nos provee de herramientas para la administración adecuada de la información.

La evolución de los manejadores de Bases de Datos ha permitido crear nuevas formas para recuperar información, pero tal vez lo más interesante es como interactuan con otros programas e inclusive con otros manejadores. Actualmente éstos incluyen en sus versiones, la programación de clientes para interactuar con otros ambientes, algunos incluyen la programación orientada a objetos, permitiendo incrementar el poder que por sí solos tienen.

Como cualquier manejador, antes de realizar cualquier tarea, se requiere de un estudio previo del sistema a desarrollar. Este estudio consiste en elaborar la normalización de las Base de Datos, ya que si esto no se realiza, se puede tener el manejador más poderoso, pero el diseño de la Base de Datos puede no ser eficiente, por lo que no se obtendrá el rendimiento ni las respuestas adecuadas. Tanto la normalización como el diseño en general son técnicas tradicionales, pero muy importantes que ayudan a la correcta elaboración de un sistema.

Este proyecto no utiliza al 100% las bases de datos porque no es su objetivo, pero ya que las incluimos, es necesario definir algunos conceptos que no podemos excluir.

Las interfaces entre Mosaic y Bases de datos en SYBASE que presentamos en este capítulo, están escritas en lenguaje C. Utilizamos SYBASE porque es un manejador de Base de Datos con el que se cuenta en la DCAA, algunos proyectos de esta dependencia han sido desarrollados en él. Es un producto que fue adquirido hace algunos meses, así que continuamos con nuestra filosofía en cuanto a aprovechar recursos de cómputo existentes para aplicarlos a sistemas específicos.

Sybase como manejador de bases de datos, trabaja en un sistema operativo como lo es UNIX, donde el usuario puede desarrollar su sistema e inclusive programar sus propias interfaces con otros ambientes.

Sybase y UNIX forman parte del software de desarrollo utilizado en esta fase del proyecto, donde tendremos a Mosaic como *front end.*

El sistema de inscripciones a cursos de la DCAA pretende ser un sistema que pueda ser utilizado por cualquier persona y desde cualquier lugar, ya que puede trabajar en red y ser accedido desde cualquier parte de la Internet. El objetivo es automatizar el proceso de inscripciones a cursos, donde la cantidad de información crece en forma rápida y se pierde el control de la misma fácilmente.

6.1.2 Planteamiento del problema

La DCAA requiere inscribir cerca de 500 alumnos cada semestre. El calendario de cursos se divide en períodos, donde cada uno de ellos tiene varios horarios y cuenta con dos aulas de cómputo.

El sistema de inscripciones a cursos de la DCAA pretende resolver los problemas que actualmente se vienen presentando en el departamento de Recursos Humanos de la Dirección de Cómputo para la administración Académica en cuanto al proceso de inscripción. Entre los cuales podemos listar los siguientes:

- Reducir el tiempo de inscripción, programación de cursos y administración en general de profesores, aulas, alumnos, etc.
- Mantener la integridad de la información.
- Obtener el control de la cantidad de información que actualmente se maneja.

Por lo tanto, se requiere un sistema que automatice las tareas del proceso de inscripción.

En la siguiente sección, el lector encontrará algunos conceptos que incluimos antes de desarrollar esta parte del proyecto. Son antecedentes que nos definen términos relacionados con las Bases de Datos. Posteriormente, tenemos una descripción general del manejador de Bases de Datos Sybase. Aquí, hacemos énfasis en la programación de clientes en lenguaje C, es necesario saber como realizar operaciones tales como insertar información en una tabla, borrarla, actualizarla, etc., desde un programa externo porque será la base para la creación de los CGI's. El modelado del sistema y la creación de la Base de Datos en Sybase es el siguiente paso. Una vez hecho esto, pasamos al desarrollo de las interfaces entre los

programas clientes y Mosaic. Incluimos ejemplos y algunas pantallas para mostrar el resultado del sistema.

6.2 Conceptos Generales de Bases de Datos

Haciendo un poco de historia, cuando se empezó a utilizar la computadora para el soporte de los sistemas de información en las empresas, se pensó que el desarrollo de estos sistemas era trivial y se cometieron una serie de descuidos, como los siguientes:

- Permitir que personas no autorizadas accedieran a los datos de la empresa.
- Perder información debido a "caídas del sistema".
- Tener valores incorrectos en los datos.
- Tener almacenada repetidas veces la misma pieza de información y perder el control sobre esta redundancia.
- Errores en el control de la concurrencia, por ejemplo, en una agencia de transporte, vender el mismo lugar a dos personas distintas.

Más adelante, la gente de sistemas se dio cuenta que la solución a todos estos problemas no era sencilla, por lo que se le dio una atención adecuada desarrollándose entonces una serie de modelos, metodologías, técnicas y herramientas que nos permitieran crear sistemas de información confiables. Uno de estos desarrollos nos lleva a los sistemas de bases de datos. Un sistema bases de datos es un sistema de información en el cual se hace uso de ciertos modelos, metodologías y técnicas que ayudan a estructurar los datos y de ciertas herramientas que nos permitan manejarlos.

En el ambiente de las bases de datos se pueden identificar dos importantes entes, los cuales aunque están íntimamente relacionados, son conceptualmente distintos, ellos son: *La Base de Datos y el Sistema Manejador de Bases de Datos (DBMS).*

Bases de Datos: En la definición de las Bases de Datos entran varios conceptos que posteriormente serán explicados por separado. Existen también muchas definiciones sobre este concepto, aquí trataremos de dar solo algunas de ellas.

"Es una colección estructurada y organizada lógicamente de datos interrelacionados, almacenados juntos sin redundancia innecesaria, para servir a múltiples y variadas aplicaciones y para permitir cambiar los requerimientos de la información (independencia física y lógica)".

"Una base de datos contiene todo lo que cualquiera quiere conocer, está organizada de tal forma, que cualquiera puede encontrar alguna pieza de información cuando la necesite".

"Una colección útil y bien organizada de información no sólo tiene los datos, sino también se sabe que se tiene y se pueden encontrar cuando se necesiten".

Independientemente de todas las definiciones, el objetivo de la tecnología de las bases de datos es el siguiente:

- Crear una independencia física y lógica de los datos.
- Habilidad de compartir datos.

- **Redundancia limitada.**
- **Habilidad para relacionar.**
- **Integridad.**
- **Flexibilidad de acceso.**
- **Seguridad.**
- **Rendimiento y eficacia.**
- **Control y administración.**
- **Ser capaces de evolucionar.**
- **Ser accesibles a múltiples aplicaciones.**

A continuación describiremos algunos conceptos básicos de palabras muy utilizadas en Bases de Datos.

Entidad: Persona, lugar, cosa, evento o concepto del cual se registra información.

Ejemplo: Alumno
Profesor

Atributo: Algo que caracteriza a una entidad. También son llamados campos de datos.

Entidad	Atributos
Alumno	No. de cuenta nombre licenciatura semestre

Dato: Valor de los campos de datos. Están contenidos en cada campo, pueden ser cuantitativos o descriptivos, dependiendo de la manera en que los campos describen a la entidad.

Un dato aislado no tiene significado.

Entidad	Atributos	Datos
Alumno	No. de cuenta	8736940-1
	nombre	Pilar Valeriano
	licenciatura	32
	semestre	10

Campo clave (campo llave): Algunos campos tienen la propiedad de que, al reconocer el valor tomado por un campo particular de la entidad, es posible identificar los datos de otros campos de esa entidad. Es posible que existan 2 o más campos que identifiquen de manera única una entidad, a esto se le llama prospecto ya que pueden convertirse en campo clave (o más conocido como campo llave).

Registro: Colección de datos tomados por campos, relacionados y almacenados en algún medio (papel, memoria de una computadora, dispositivo magnético de almacenamiento, etc.).

Archivo: Conjunto de registros. Por lo regular los registros que conforman a un archivo son homogéneos.

Información: Conjunto de datos relacionados con significado. El objetivo de la información es adquirir conocimiento.

"La información mueve a toda la organización"

En la actualidad, la información se ha convertido en un arma de competividad.

Características de los sistemas tradicionales

- Son aplicaciones para resolver problemas de negocio individuales (actividades aisladas).
- Se pueden justificar económicamente muy fácil.
- Estas aplicaciones tradicionales no alteraban significativamente la forma en que las empresas hacían las cosas, la rutina y las tareas repetitivas fueron automatizadas, pero no cambiadas o redefinidas, aun cuando se "sistematizó" antes de hacer la sistematización.
- El programador desarrollaba un sistema trabajando con la gente afectada: nómina, contabilidad, etc. Consecuentemente la vista de la empresa estaba sustentada por el departamento de nóminas, contabilidad, etc., frecuentemente estas vistas eran sorprendentemente limitadas e inconsistentes.

Una de las ventajas de los sistemas tradicionales es que se realizan los mismos cálculos en una forma más rápida.

Problemas de los sistemas tradicionales

- Redundancia innecesaria. En los sistemas tradicionales los mismos datos son requeridos por diferentes aplicaciones y se registran en varios archivos de forma repetida. La redundancia conduce a problemas que tienen que ver con la integridad de la información. Cuando existe redundancia, se requieren múltiples procesos de entrada, actualización e informes. Además de la redundancia de datos, es común que exista también redundancia de procesos.

- Falta de integridad. Es una de las consecuencias de la redundancia. Cuando se realizan cambios en algún campo de un archivo, se tiene que actualizar ese dato en todos los archivos que tengan el mismo campo.
- Compartimiento limitado de datos. No se puede compartir información entre diferentes archivos de datos ya que están instalados como unidades separadas.
- Disponibilidad de datos limitada. Los datos deben estar disponibles para la persona autorizada en el momento oportuno.
- Inconsistencia de datos. Desorden en el universo (entropía), hay varias versiones de los datos.
- Inconsistencia de representaciones y códigos. Por ejemplo: En una base de datos un campo CLAVE es numérico y en otro es alfanumérico o de diferente longitud de los otros campos, etc. La consistencia da lugar a que exista integridad en una base de datos.
- Falta de seguridad.
- Documentación deficiente.

Los paradigmas y herramientas nos ayudan a implantar estas propiedades en la base de datos. Sin embargo, es el profesional en computación quien aplicando sus conocimientos e ingenio tiene la responsabilidad de alcanzarlas.

Sistema Manejador de Base de Datos (DBMS)

Por otro lado tenemos la herramienta que nos va a permitir trabajar con los datos: el sistema manejador de base de datos (DBMS o DataBase Management System), el cual es un conjunto de rutinas que nos permiten definir, crear, acceder, respaldar, recuperar y administrar la base de datos garantizándonos seguridad, integridad y protección de los datos, así como de sincronizar el acceso de múltiples

aplicaciones. Es también el que nos impone cierta disciplina, tanto para modelar la base de datos como para accederla. Por seguridad, debe entenderse que sólo podrán acceder la base de datos los usuarios autorizados y cada uno verá solo la parte que le corresponde.

La integridad nos dice que los valores almacenados son correctos y que la base de datos es consistente.

La protección nos asegura que en caso de "corrupción" de la base de datos, es posible recuperar los datos correctos.

Finalmente, la sincronización controla que varias aplicaciones accedan en forma concurrente la base de datos. Cabe aclarar que las funciones de seguridad, integridad, protección y sincronización no se obtienen en forma automática, el DBMS sólo nos provee las rutinas necesarias las cuales hay que saber aplicar, siendo entonces los usuarios encargados de administrar la base de datos y de desarrollar aplicaciones los que tienen que hacer buen uso de estas rutinas para obtener los beneficios propios de un DBMS.

Metodología de Análisis de Richard Barker

Esta metodología está orientada a un análisis de sistemas de Bases de Datos y determina una serie de actividades a través de las cuales se especifican los requerimientos de la información así como de las funciones y procedimientos que operarán sobre dichos datos.

Tiene por objeto comprender las "cosas de importancia" para el sistema, la información que de ellas se requiere y cuales son las relaciones que se establecen entre ellas.

De este análisis de entidades debe obtenerse el modelo de datos (Diagrama de Entidades y Relaciones) y debe así especificarse los atributos de estas entidades, así como los volúmenes esperados de crecimiento de la Base de Datos.

Es necesario determinar los dominios de la información que intervienen en el sistema. Los dominios son conjuntos de valores con propiedades determinadas de los cuales los atributos de las entidades adquieren su valor.

Un ejemplo podría ser el "dinero", donde sus propiedades son que debe ser numérico, mayor que cero y con dos posiciones en el punto decimal, de esta manera si se tuviera una entidad "empleado" en la cual un atributo es su salario, podríamos especificar que el atributo salario pertenece al dominio del dinero.

De esta manera podemos detectar en un sistema un gran número de atributos que provienen de un dominio común, de tal forma que este dominio sea especificado sólo una vez.

Diseño de la Base de Datos

El diseño de la Base de Datos consiste en definir las estructuras de datos que procederán a construirse en la herramienta seleccionada; hablando en términos de los sistemas desarrollados bajo un ambiente de base de datos, el diseño consiste en determinar las tablas que representarán correctamente las entidades definidas durante el análisis.

El diseño debe resolver los problemas de modelar superentidades y tener relaciones determinadas por arcos, así como la definición completa de atributos.

Para llevar al cabo este diseño de tablas es necesario incorporar el término de "normalización". Podemos definir a la normalización como una técnica que nos garantiza que las entidades estén identificadas de manera única. Existe un gran número de reglas de normalización o formas normales, sin embargo, es muy común

que los diseños de datos se encuentren hasta la tercera forma normal. A continuación procedemos a describir las primeras tres formas normales.

Primera Forma Normal

Para que un registro esté en la primera forma normal es necesario eliminar a los atributos o grupos de atributos repetidos.

Segunda Forma Normal

Para que un registro se encuentre en segunda forma normal es necesario eliminar los atributos dependientes de una parte de la llave primaria.

Tercera Forma Normal

Para que un registro se encuentre en la tercera forma normal es necesario eliminar los atributos que dependan de atributos que no forman parte de la llave primaria.

Los beneficios que presenta el uso de la normalización se pueden resumir en:

- Una mayor consistencia en los datos.
- Redundancia mínima y controlada.

- Elimina dependencias no deseadas que se traducen en problemas de actualización, es decir, al insertar, modificar o borrar un renglón de una tabla puede ser necesario hacerlo en otras.

Sin embargo, no siempre es posible tener un modelo 100% normalizado (considerando sólo hasta la Tercera Forma Normal), existen situaciones en las cuales es necesario incorporar relaciones redundantes o incluso duplicar información, esto normalmente es debido a la necesidad de obtener un mejor tiempo de respuesta en los procesos o hacer un uso eficiente del hardware.

En resumen, se puede decir que en los casos en los que se requiera un mejor tiempo de respuesta y no pueda ser posible obtenerlo por tener toda la información normalizada, es válido pensar en soluciones que impliquen desnormalizar el modelo siempre y cuando éstas se documenten y se incorporen los procesos que arreglen los problemas de actualización que **de estas soluciones se deriven**.

6.2.1 SYBASE como manejador de bases de datos

SYBASE es un nombre genérico para un grupo de productos de la compañía Sybase. Los principales productos son *SQL server* y *SQL toolset*. *SQL server* es un sistema manejador de bases de datos (DBMS) que puede correr dentro de una gran variedad de sistemas operativos, incluyendo VMS, Sun Unix, OS/2 y otros, permite que los usuarios o programas de aplicación pueden acceder a las bases de datos relacionales.

El *SQL toolset* es un conjunto de interfaces programables para el *SQL server*. Se puede comunicar con cualquier *SQL server* y manipular datos mediante el lenguaje relacional de SYBASE.

El lenguaje relacional de SYBASE se llama transact-SQL; es una versión del bien conocido lenguaje relacional SQL, con numerosas extensiones para el soporte de procesamiento de transacciones. SQL es el lenguaje de bases de datos que más se maneja hoy en día en las bases de datos relacionales. Un sistema relacional es aquel donde:

a) Los datos del usuario se almacenan en forma de tablas; y
b) Los operadores generan las nuevas tablas a partir de las que ya existían. Por ejemplo, habrá un operador que recupere un subconjunto de filas a partir de una tabla, y otro que haga lo mismo con las columnas de la tabla dada anteriormente.

Bases de Datos Relacionales

Las bases de datos de SYBASE son esencialmente relacionales. Una base de datos relacional es aquella que es percibida por sus usuarios como una colección de tablas. Un ejemplo (de sustitutos y de partes de la base de datos) se muestra en las siguientes tablas:

S	S#	SNOMBRE	STATUS	CIUDAD
	S1	CARLOS	10	LONDRES
	S2	RICARDO	20	PARIS
	S3	PILAR	30	MEXICO

P	P#	PNOMBRE	COLOR	PESO	CIUDAD
	P1	Tornillo	Rojo	12	LONDRES
	P2	Tornillo	Verde	17	PARIS
	P3	Tuerca	Azul	14	ROMA

SP	S#	P#	CANT
	S1	P1	300
	S1	P2	200
	S2	P3	400
	S2	P4	100

Figura 6.1 Los sustitutos y partes de una Base de Datos (muestra de valores)

Como se puede observar, esta Base de Datos consiste de tres tablas llamadas S, P y SP.

La tabla S representa a los sustitutos. Cada sustituto tiene un número de sustitución (S#) único; un nombre (SNOMBRE) no necesariamente único, un valor de status (STATUS); y una localidad (CIUDAD). Para este ejemplo, asumiremos que cada sustituto está localizado exactamente en una ciudad.

La tabla P representa las partes. Cada una tiene un número (P#) el cual es único, y está formado por otros campos llamados COLOR, PESO y una localidad en donde se asume que cada una de esas partes está almacenada en una ciudad.

La tabla SP representa una conexión para poder unir a las otras dos tablas sin tener que ser específico en los demás campos, donde la combinación de S# y P# es 0única con respecto al conjunto de envíos que aparecen en la tabla SP.

A continuación mencionaremos algunos puntos característicos de las tablas:

- Primero, nótese que todos los valores son atómicos. Esto es, en cada fila y columna de una tabla existe siempre un valor, nunca un conjunto de múltiples valores. Las Bases de Datos Relacionales no permiten grupos de datos repetidos.

- Segundo, nótese que toda la información contenida en la Base de Datos está representada por valores de datos explícitos. Este método de representación es el único disponible en una Base de Datos Relacional. Específicamente, no existen "ligas" o punteros conectando una tabla a otra.

- Tercero, cada una de las tablas en el ejemplo anterior tiene un identificador único, una columna (o una combinación de columnas) cuyo valor en cualquier fila dada es único con respecto al conjunto de todos esos valores que aparecen en la tabla. El único identificador para la tabla S es S#; para la tabla P, es P#; y para la tabla SP es la combinación (S#,P#). Por ejemplo, los valores de la columna S# de la tabla S pueden ser utilizados para localizar con la mayor exactitud las filas de sustitución individuales dentro de la misma tabla.

El término para representar un identificador único dentro de una tabla, se le conoce como *llave primaria*, SYBASE no obliga a que en cada tabla exista una llave primaria, pero a los usuarios se les recomienda ampliamente que exista una en la práctica.

Uno se podrá preguntar porqué a la Base de Datos vista en el ejemplo se le llama "relacional". La respuesta es simple: "Relación" es sólo un término matemático para una tabla. Entonces, podemos decir que la Base de Datos del ejemplo anterior está compuesta de tres relaciones. De hecho, para este caso, tomaremos el término "relación" y "tabla" como un sinónimo. Los sistemas relacionales tienen su origen en una teoría matemática, pero entonces ¿porque no llamarlas sólo tablas?, es posible, pero también debemos de entender el término "relación". Los sistemas relacionales están basados en lo que se conoce como *el modelo relacional de datos*. Este modelo está basado en la teoría del Dr. E. F. Codd, quien planteó los principios teóricos del manejador de Bases de Datos, las cuales

hoy en día son aceptadas universalmente en el desarrollo de tecnologías de Bases de Datos, y consideradas en otros campos como la Inteligencia Artificial y el Procesamiento Natural del Lenguaje.

La Figura 6.2 muestra los términos que estaremos usando más frecuentemente (tabla, fila, columna y llave primaria). Para nuestro interés, también se nos da el término formal para cada caso.

Términos en forma relacional	**Equivalentes informales**
relación	tabla
tuple	fila
atributo	columna
llave primaria	identificador único

Figura 6.2 Terminología

El Lenguaje SQL

Este lenguaje se utiliza para formular operaciones relacionales (p.e. operaciones que definen y manipulan datos en una forma relacional). A continuación se muestra un ejemplo del como se define una tabla:

```
CREATE TABLE  S
(S#        CHAR(5)  NOT NULL CONSTRAINT S_PK PRIMARY KEY,
SNAME  CHAR (20) NOT NULL DEFAULT  'UNKNOWN',
STATUS SMALLINT  NOT NULL DEFAULT 0,
CITY       CHAR(15)  NOT NULL DEFAULT 'UNKNOWN' );
```

Como se puede observar, en el ejemplo anterior se incluye una sentencia CREATE TABLE, la cual especifica el nombre de la tabla a ser creada, los nombres

y tipos de datos de las columnas de la misma. Debemos enfatizar que la sentencia CREATE TABLE es una *sentencia ejecutable*. Inicialmente la tabla estará vacía, es decir que sólo estamos definiendo los encabezados de esa tabla, es entonces cuando podemos utilizar la sentencia INSERT para agregar datos.

Continuando con el ejemplo, una vez creada la tabla e insertado algunas filas en ella, podemos empezar a hacer uso de la misma, utilizando *sentencias de manipulación de datos de SQL*. Por ejemplo, la *recuperación de datos*, la cual en SQL se especifica mediante la sentencia SELECT.

El siguiente ejemplo nos ilustra el uso de esta sentencia:

En forma interactiva (ISQL):

```
SELECT CITY FROM S WHERE S# = 'S4'
                    RESULT:     CITY
                                ----------
                                LONDON
```

Una característica particular de la mayoría de los sistemas relacionales, incluyendo a SYBASE, es que el mismo lenguaje relacional está disponible en dos diferentes interfaces, llamadas interactiva (manejada con ISQL) e interfaz de programación.

En el ejemplo anterior se muestra una interfaz interactiva manejando ISQL. Aquí, el usuario ha tecleado la sentencia SELECT desde su terminal, y el servidor SQL ha respondido a esa solicitud mediante el desplegado del resultado en forma casi inmediata en esa terminal. Note que esta misma sintaxis podría ser utilizada en una función de lenguaje de tercera generación como un *Open Client* o en un lenguaje de cuarta generación como *APT*, en cuyo caso los resultados son devueltos a una variable de un programa.

Por lo tanto, SQL es tanto interactivo como programable. De hecho, cualquier sentencia que se introduce en una terminal, puede ser alternativamente utilizada en

un programa. Asimismo, se pueden crear tablas dentro de un mismo programa, si así lo requiere una aplicación (y si se está autorizado para realizar ese tipo de operaciones). Las sentencias SQL pueden ser utilizadas en programas escritos en cualquiera de los siguientes lenguajes: BASIC, C, COBOL, FORTRAN entre otros. El lenguaje apropiado depende del sistema operativo que se esté manejando y del hardware que se tenga disponible para hacer al programa lo más eficiente posible.

Ahora estamos en posición de comprender como SYBASE ve a el usuario. Por "usuario" entendemos al elemento que está utilizando al sistema en una terminal, como aquel que está desarrollando un programa de aplicación en lenguaje C.

El primer punto es que debemos estar conscientes de que normalmente existirán varios usuarios, de ambos tipos usando el sistema al mismo tiempo. El servidor SQL automáticamente aplicará las medidas necesarias para asegurarse de que todos estos usuarios estén protegidos uno del otro. Por ejemplo, va a garantizar que alguna actualización que pueda realizar un usuario, no va a causar un resultado incorrecto en la operación de algún otro usuario.

Las tablas, al igual que los usuarios, se catalogan en dos tipos llamadas *tablas base* y *vistas*.

- Una tabla base es una tabla "real", es decir, una tabla que físicamente existe, en el sentido que están almacenados registros y posiblemente índices físicos en uno o más archivos, que directamente representan aquella tabla almacenada. Las tablas S, P y SP en el ejemplo anterior son tablas base, y utilizaremos el término "tablas base" en lugar de tablas cuando queramos excluir a las vistas.

- En contraste, una vista es una tabla virtual, esto es, una tabla que no existe directamente en un lugar físico, pero el usuario la ve como si esta existiera. Las

vistas se pueden interpretar como una manera diferente de ver las tablas base. Además, las vistas son definidas mientras que las tablas son creadas.

Tal como las tablas, las vistas también pueden ser creadas en cualquier momento mediante la sentencia CREATE VIEW para la creación de tablas virtuales, y análogamente existe la sentencia CREATE INDEX para la creación de índices. Similarmente, las tablas, vistas e índices pueden ser eliminadas mediante el uso de sus respectivas sentencias: DROP TABLE, DROP VIEW y DROP INDEX.

La interfaz primaria del usuario al servidor SQL es el lenguaje SQL, ya hemos mencionado que SQL tiene dos tipos de usuarios y que nos da las opciones de manipulación de datos y funciones. Las principales sentencias para la definición de datos son: CREATE TABLE, CREATE VIEW, CREATE INDEX, DROP TABLE, DROP VIEW y DROP INDEX, y las principales sentencias para el manejo de datos son: SELECT, INSERT, UPDATE y DELETE.

Con la versatilidad del manejo de datos de SQL, se puede ver que es un lenguaje de selección de niveles. Por lo tanto, se puede concluir que SQL maneja mas detalles sin necesidad de especificarlos ampliamente en otro lenguaje como sería COBOL por ejemplo, y por lo tanto incrementa la productividad en materia de sistemas relacionales.

Estructura Cliente Servidor

SYBASE es un sistema con arquitectura cliente/servidor. El servidor es el DBMS (manejador del sistema de bases de datos por sus siglas en inglés: Data Base Management System) relacional. Este se encarga de proveer todas las funciones básicas para el DBMS incluyendo un soporte total para todo el lenguaje SQL como sería: definición, manipulación, seguridad e integridad de los datos. También sirve como manipulador de transacciones dando el control necesario de concurrencia de

las operaciones así como el soporte de recuperación de las mismas. En adición a esto, también da soporte al diccionario de datos de SQL. El servidor es conocido formalmente como *servidor* SQL.

Los clientes constituyen el conjunto de herramientas de SQL. En otras palabras SYBASE no es solamente un DBMS, sino que es un DBMS relacional para dar facilidades en el proceso de desarrollo de aplicaciones de bases de datos. En sí, los clientes consisten de varios subsistemas distintos. Los subsistemas de clientes más importantes se listan a continuación:

- **Data Workbench**. Administración del DBMS, herramientas y utilerías para el DBO (Dueño de la base de datos por sus siglas en inglés: Database Owner), formas (para entrada de datos), composición de consulta visual, manipulación de datos y generación de reportes (Report Workbench).

- **APT Workbench**. APT significa: "herramientas para productividad de aplicación" (*application productivity tools*), y existen varias aplicaciones que utilizan APT como edición (APT-Edit), para generación de reportes (APT-Build), y para ejecución (APT-Execute) con un lenguaje de cuarta generación conocido como APT-SQL.

- **Open Client**. Es una biblioteca con funciones 3GL (lenguaje de tercera generación) para la creación, cierre y control de conexiones al servidor SQL, enviar los datos al servidor, manipulación, definición y manejo de transacciones y sentencias; se le conoce formalmente como DB-LIBRARY.

- **Net library**. Una biblioteca con funciones 3GL para manejar todas las conexiones de red y enrutamiento.

- **APT library**. Una biblioteca con funciones 3GL para definición y edición de formas.

- **RPT library**. Una biblioteca con funciones 3GL para definición y edición de reportes.

- **ISQL**. Es un comando interactivo para atender peticiones SQL.

La división que nos da SYBASE entre los clientes y los servidores está en una flexibilidad para el manejo de sus componentes: Un cliente y los componentes de un servidor se pueden mover de un sitio a otro sin la necesidad de reescribirlos. Como podemos ver, la división de estos componentes es una parte importante de las capacidades de los procesos distribuidos de SYBASE.

SYBASE como soporte de red

Cualquier combinación de arquitecturas VAX/VMS, OS/2, UNIX, y Windows NT cliente y servidor pueden ser conectados juntamente en la misma red. Los clientes de SYBASE en cualquier sitio pueden entonces interactuar con los servidores de SYBASE. Por ejemplo, un usuario en un sitio "A" puede realizar operaciones con SQL en un sitio "B". Todo lo que tiene que hacer, es especificar el nombre del servidor "B" que va a utilizar.

Una gran variedad de redes se pueden utilizar por ejemplo: TCP/IP, DECnet, Netware, etc. Un archivo especial llamado *interfaces* se utiliza para proveer un mapa lógico del nombre del servidor a la dirección de la red, nombre de la máquina, tipo de red, etc. Un nuevo servidor se hace accesible mediante la edición del archivo y

haciendo las entradas apropiadas, asumiendo que las conexiones físicas ya se han realizado.

6.2.2 Clientes en Sybase

En las secciones anteriores, explicamos que SQL puede usarse como un lenguaje interactivo de consultas y como un lenguaje de programación. Hasta este punto, hemos mas o menos ignorado los aspectos de programación de SQL y considerado prácticamente que se usa en forma interactiva. Ahora, centramos nuestra atención en esos aspectos de programación específicamente.

El principio fundamental de la programación SQL, es que cualquier sentencia de este lenguaje que puede usarse en la terminal, puede también ser usada en un programa de aplicación.

Sybase provee dos métodos principales de interactuar con el servidor SQL a través de un programa. En esta sección, describiremos el método *interfaz de llamada de función* (Function Call Interface) utilizado por un Open Client (Programa cliente) y un componente llamado DB-Library. Los lenguajes de programación actualmente soportados por el servidor SQL usando DB-Library, son: Ada, C, Cobol, Fortran y Pascal.

¿Qué es una Interfaz de llamada de función (Function Call Interface)?

Una interfaz de llamada de función tal como un Open Client, es un conjunto de funciones 3GL compiladas, archivos fuentes (archivos "include" o "header"), y macros (en este caso, escritas en lenguaje C). Este cliente accede a un conjunto de servicios tales como los de un servidor SQL. Uno de los propósitos de definir y diseñar este tipo de funciones o módulos es ocultar la implantación exacta de sus

tareas. Típicamente el programador necesita saber sólo la manera de invocarlas, conocer su comportamiento, y preparar un programa que haga uso de ellas.

Un Open Client consiste de dos partes: Las funciones tradicionales de la biblioteca DB-Library y las más recientes, llamadas funciones de Open Client (OC-Lib). Ambas acceden las Net-Library (bibliotecas de red). Estas últimas proveen los servicios de comunicación entre cliente y servidor. Son invocadas automáticamente por los programas Cliente.

Como ya hemos mencionado, el Open Client consiste de un conjunto de funciones 3GL, residentes en una biblioteca y usadas por los programas clientes. Un programa que usa esta biblioteca tiene una estructura genérica, los pasos que sigue son los siguientes:

1. Establecer una conexión con el servidor SQL.
2. Crear una sentencia.
3. Enviarla al servidor.
4. Procesar los resultados de cada sentencia, y por último.
5. Cerrar la conexión.

Aunque hemos ignorado un número importante de características poderosas: Primero, un programa puede dividirse en un cierto número de módulos, los cuales a su vez pueden dividirse en otros nuevos. Segundo, un programa cliente puede establecer múltiples conexiones con un servidor SQL. Tercero, los pasos 2, 3 y 4 pueden repetirse varias veces antes de cerrar la conexión. Y finalmente, una secuencia de eventos completa puede repetirse más de una vez con un programa.

¿Que es una DB-LIBRARY?

Los productos con arquitectura cliente/servidor tales como SYBASE requieren un canal de comunicación entre el cliente y el servidor. Mientras los detalles de esta comunicación y protocolo son manejados por las Net-Library, el Open Client debe establecer y mantener una conexión, proveer un medio para manipular sus peticiones, y finalmente cerrar la conexión. Cada vez que un cliente establece una conexión con el servidor SQL, se crea una estructura de datos especial asociada a un número único de proceso, llamado ID (identificador de proceso). Por cada proceso, existe una estructura de datos llamada "DBPROCESS" mediante la cual se comunican el programa cliente y el servidor SQL. Regularmente contiene:

- Comandos SQL
- Resultados que regresa el servidor
- Estado de la información

La DB-Library es una API (Applications Programming Interface o interfaz para programación de aplicaciones), consiste en varias rutinas utilizadas para manejar la comunicación entre aplicaciones cliente y los servidores de SYBASE (servidor SQL y OPEN Server). Entre sus múltiples funciones se encuentran las siguientes:

- Transmitir sentencias SQL al servidor
- Recuperar los datos para el programa cliente
- Realizar conversión de tipos de datos

Todos los programas clientes deberán utilizar la DB-Library. El conjunto de interfaces cliente/servidor está compuesto por:

- DB-LIBRARY. Es el conjunto de funciones que permiten desarrollar aplicaciones a los clientes.
- Net-Library. Esta parte se encarga de adoptar los protocolos de servicio.
- Server-Library. Biblioteca del servidor, programas desarrollados como aplicaciones Open Server.
- Tabular Data Stream (TDS). Es el protocolo de comunicación entre el cliente y el servidor.

Beneficios de la DB-Library

- Provee un camino estándar y consistente para los desarrolladores de aplicaciones con los servidores de SYBASE.
- Aisla a los desarrolladores de los detalles de red y protocolos.
- Provee llamadas de funciones y definiciones estándares.
- Aisla las aplicaciones de los cambios internos del servidor SQL mediante el uso de Definiciones Simbólicas.

La DB-Library como interfaz Open Client

- El servidor SQL y el Open Server se encargan de responder y procesar todas las peticiones del cliente vía DB-Library.
- Las peticiones son enviadas en sentencias SQL al servidor.
- Los resultados se regresan a la DB-Library utilizando el protocolo TDS (Tabular Data Stream).

Los mensajes, errores y notificaciones son enviados desde el servidor a la DB-Library.

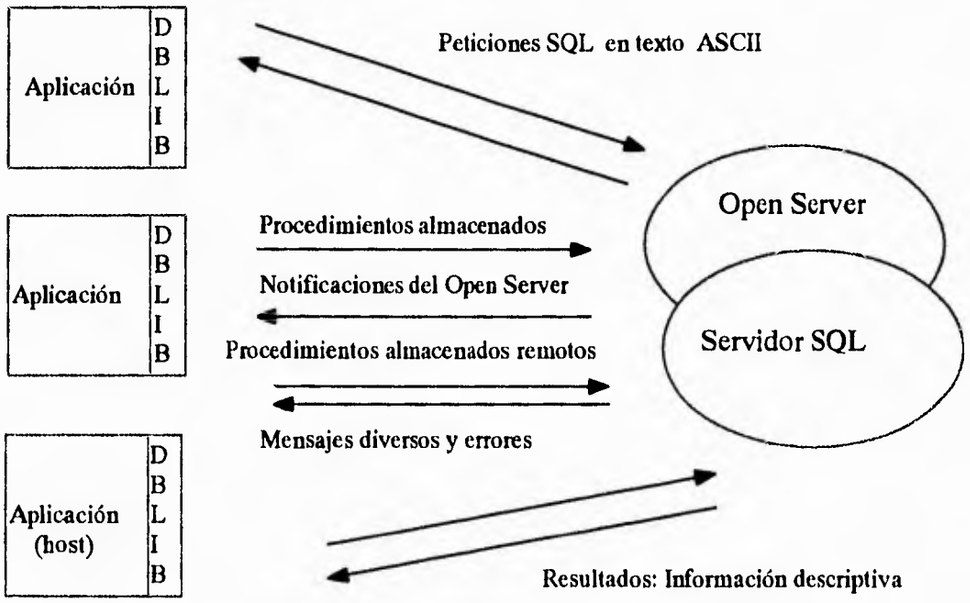


Figura 6.3 Las interfaces Cliente/Servidor de Sybase

Funciones Básicas de la DB-Library

- Funciones de Inicialización
  - Instalación de manejadores de mensajes y errores
  - Crear claves de entrada (*logins*)
  - Establecer una conexión al servidor SQL
  - Uso de Bases de Datos

- Configuración y ejecución de comandos
  - Construir comandos SQL
  - Enviarlos al servidor
  - Construir y enviar llamadas de procedimientos almacenados

- Procesamiento de resultados
  - La mayoría de las funciones caen dentro de esta categoría
  - Los resultados se procesan solos o en un buffer a un tiempo

- Diversas Rutinas
  - Obtener información acerca de los datos
  - Estado de un proceso o comando

- Funciones de limpieza general
  - Cerrar conexiones
  - Limpiar buffers y estructuras de datos

Convenciones usadas por DB-Library

- Nombres de funciones
  - Todas las funciones son de la forma **dbxxx**, y son regularmente en letras minúsculas.

- Parámetros
  - La mayoría de las funciones requieren parámetros
  - Algunas funciones aceptan el valor NULL como parámetro

- Estado de la información
  - Muchas funciones regresan un valor que puede ser verificado por el programa.

Adicionalmente, la DB-Library provee los siguientes valores definidos como estado de las funciones:

RETCODE	NO_MORE_ROWS
FALSE	MORE_ROWS
TRUE	BUF_FULL
SUCCEED	NO_MORE_RESULTS
FAIL	REG_ROW
NULL	

Por otra parte, manejan las siguientes estructuras de datos: LOGINREC y DBPROCESS, las cuales se describirán en la siguiente sección.

Comenzando con la programación

En este apartado describiremos los pasos fundamentales para la elaboración de un programa cliente en Sybase. Es importante saber usar las funciones que nos provee la DB-Library, así como tener presente un pequeño diagrama de flujo en el desarrollo de aplicaciones o programas de este tipo.

El sistema que presentado en esta tesis, se desarrolló en la versión 4.9.2 de SYBASE corriendo bajo sistema Operativo Unix Solaris 2.3, por lo que la creación de los programas cliente será en lenguaje C. Por otro lado, los mismos programas pueden correr en la versión reciente de Sybase (que es System X) sin la necesidad de modificar el código fuente.

Primero, para usar la DB-Library, se debe inicializarla haciendo una llamada a la función **dbinit()**. Al invocarla, se crean las estructuras de datos: LOGINREC y DBPROCESS. La estructura LOGINREC se usa para colocar la clave y password del usuario de la base de datos antes de establecer una conexión con el servidor SQL.

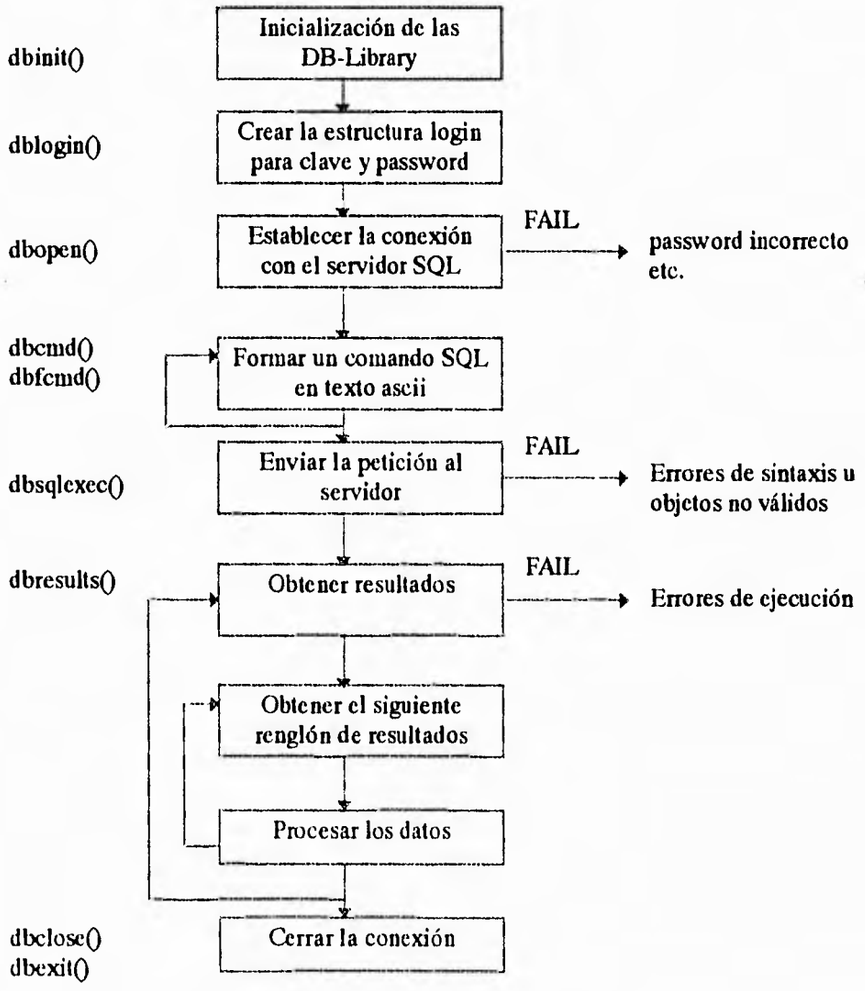


Figura 6.4 Esqueleto básico de un programa cliente

Una vez que se ha inicializado la DB-Library, se especifica una clave con la función **dblogin()**, la cual se almacenará en la estructura LOGINREC. Esta

estructura se envía al servidor SQL usando la función **dbopen()**. El servidor validará la cuenta y password del usuario, si son incorrectos enviará el error correspondiente, de lo contrario se establece la conexión. Si el nombre del servidor no se especifica en esta función, por default será tomado de la variable de ambiente DSQUERY. Para cambiar los valores por default del LOGINREC, se utilizan las siguientes instrucciones:

```
login=dblogin();
DBSETLUSER(login,"usuario")
DBSETLPWD(login,"password")
```

Nota: Para declarar un password nulo, se deberá de especificar el comando DBSETLPWD(login,"") en C.

La estructura de datos DBPROCESS

Su función es la de proveer las estructuras básicas de datos para que se puedan comunicar el programa de aplicación y el servidor SQL especifico o un Open Server.

Típicamente incluye: un buffer para comandos SQL, los datos que fueron devueltos por el servidor y estado de la información. Se incluye en el primer argumento de casi cualquier llamada de la DB-Library. Esta estructura de datos se inicializa con la función **dbproc()** de la siguiente manera:

```
dbproc=dbopen (login,"nombre_del_servidor")
```

dbopen siempre regresará un puntero a la estructura DBPROCESS si los datos de la clave del usuario son válidos o un NULL si un error ocurre.

estructura se envía al servidor SQL usando la función **dbopen()**. El servidor validará la cuenta y password del usuario, si son incorrectos enviará el error correspondiente, de lo contrario se establece la conexión. Si el nombre del servidor no se especifica en esta función, por default será tomado de la variable de ambiente DSQUERY. Para cambiar los valores por default del LOGINREC, se utilizan las siguientes instrucciones:

```
login=dblogin();
DBSETLUSER(login,"usuario")
DBSETLPWD(login,"password")
```

Nota: Para declarar un password nulo, se deberá de especificar el comando DBSETLPWD(login,"") en C.

La estructura de datos DBPROCESS

Su función es la de proveer las estructuras básicas de datos para que se puedan comunicar el programa de aplicación y el servidor SQL especifico o un Open Server.

Típicamente incluye: un buffer para comandos SQL, los datos que fueron devueltos por el servidor y estado de la información. Se incluye en el primer argumento de casi cualquier llamada de la DB-Library. Esta estructura de datos se inicializa con la función **dbproc()** de la siguiente manera:

```
dbproc=dbopen (login,"nombre_del_servidor")
```

dbopen siempre regresará un puntero a la estructura DBPROCESS si los datos de la clave del usuario son válidos o un NULL si un error ocurre.

Los errores pueden ser causados por clave incorrecta del usuario, o que el servidor no este corriendo, un nombre del servidor incorrecto, etc. No proceden las funciones de la DB-Library cuando una llamada a la función **dbproc** devuelve un valor NULL.

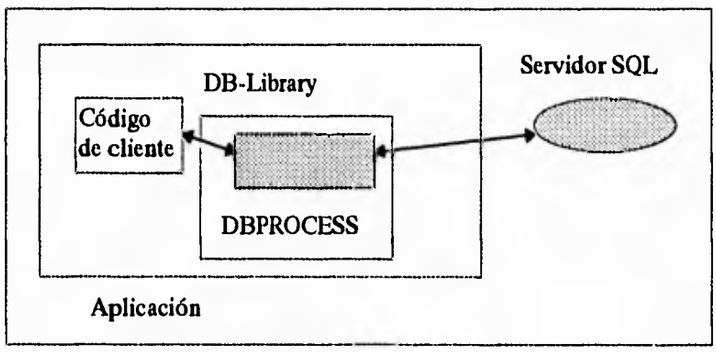


Figura 6.5 La estructura de datos DBPROCESS entre cliente y servidor

El nombre del servidor se encuentra en el archivo *interfaces* y forma parte del software de Sybase, se encuentra bajo el directorio especificado en la variable de ambiente SYBASE. Un ejemplo de su contenido sería el siguiente:

```
#
# Created: 11/25/1993 (sybase)
#
# SYBASE on tzetzal (132.248.27.10) using tcp
#       services: query (1024) master (1024) console (1025)
#
## SYBASE on tzetzal.dcaa.unam.mx
##      Services:
##              query  tcp     (1024)
##              master tcp     (1024)
##              console        tcp     (1025)

SYBASE
        query tli tcp /dev/tcp \x0002040084f81b0a0000000000000000
        master tli tcp /dev/tcp \x0002040084f81b0a0000000000000000
        console tli tcp /dev/tcp \x0002040184f81b0a0000000000000000
```

El archivo llamado *interfaces* contiene los nombres de servidores SQL que se encuentran en la red. Regularmente se localiza bajo el directorio especificado por la variable de ambiente SYBASE. Los programas clientes de Sybase son portables entre máquinas siempre y cuando éstas sean de la misma arquitectura y sistema operativo. Así, yo puedo programar un cliente en una máquina que tenga instalado Sybase, y el código ejecutable junto con el archivo *interfaces* (es necesario transferirlo puesto que tiene información acerca del servidor) pueden ser transferidos a otra máquina conectada a la red que no tenga instalado el producto. En la máquina remota, antes de ejecutar el programa se deben definir adecuadamente las variables de ambiente SYBASE y DSQUERY, más adelante mostramos un ejemplo de cómo hacerlo. Esto representa una gran ventaja, dado que los programas clientes no necesariamente deben residir en la máquina local.

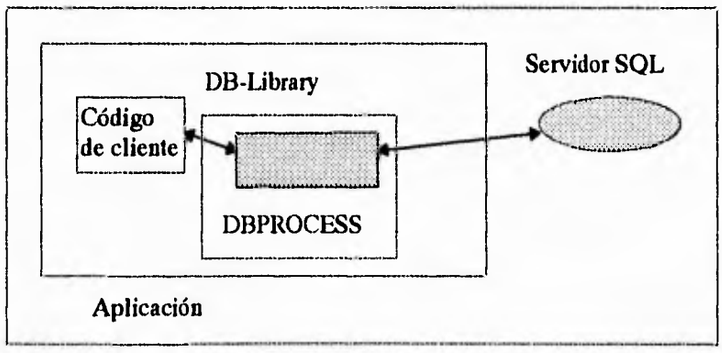


Figura 6.5 La estructura de datos DBPROCESS entre cliente y servidor

Para cada conexión del cliente con el servidor SQL existe una estructura DBPROCESS (aún si hay múltiples DBPROCESS con la misma clave y el mismo password). Una aplicación puede abrir múltiples DBPROCESS a múltiples servidores, y múltiples DBPROCESS a un servidor.

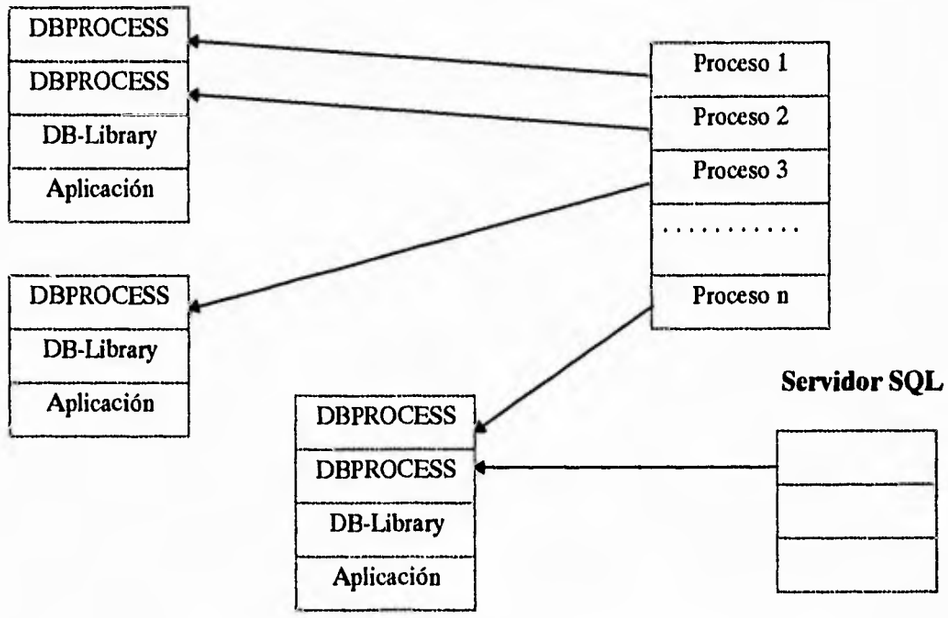


Figura 6.6 Relación de procesos entre programas clientes y servidor SQL

Construyendo comandos SQL

Una vez que se ha creado la conexión entre nuestro programa cliente y el servidor SQL apropiado para las funciones definidas anteriormente, podemos comenzar a crear comandos SQL con otra función especial llamada **dbcmd()**. Esta función únicamente forma el comando SQL almacenándolo en la estructura DBPROCESS. La sintaxis es la siguiente:

```
dbcmd(dbproc,"sentencia SQL como cadena ASCII")
```

Recordemos que la mayoría de las funciones de la DB-Library tienen como argumento la variable **dbproc**, esto es simplemente para identificar la sesión en la

que estamos trabajando. En el programa de aplicación se podría generar otra sesión totalmente diferente a la primera (utilizando otra base de datos y otra clave de acceso por ejemplo) y reconocerla con otro nombre por ejemplo *dbproc2*, por lo tanto, es importante saber a cual conexión nos referimos para evitar confundirnos entre las sesiones que genere nuestro programa cliente. Los procedimientos almacenados (stored procedures) deberán ser precedidos por un "execute" (o exec) si el procedimiento almacenado es especificado después de la primera línea con otro **dbcmd**(). Ejemplos:

1. ```
   dbcmd (dbproc, "select  * from authors");
   dbcmd (dbproc, " where au_id > 10");
   ```

2. ```
   dbcmd (dbproc, "sp_help authors");
   dbcmd (dbproc, " execute  sp_help sales");
   ```

3. ```
   char *sqltext;
   .....
   sqltext = "select * from pubs..sales";
   dbcmd ( dbproc, sqltext );
   ```

Enviando Comandos al servidor SQL

Una vez que se ha especificado el comando (o comandos) SQL en la estructura DBPROCESS con ayuda de la función dbcmd, el siguiente paso es mandarlo(s) a ejecutar en el servidor con la función **dbsqlexec**() de la siguiente forma:
   ```

return_code=dbsqlexec(dbproc); y después la DB-Library espera por los primeros resultados.

Todo el proceso por lotes mandado (batch) deberá de ser sintáctica y semánticamente correcto o la rutina tendrá un error determinado por el servidor SQL.

Valores devueltos por el servidor

El servidor enviará un mensaje SUCCEED si no existieron errores en la ejecución del comando SQL, de otra manera mandará un mensaje de FAIL si es que existieron errores de sintaxis u objetos ilegales, la codificación típica para el FAIL es la siguiente:

```
if (dbsqlexec(dbproc) == FAIL)
        /* salida o error de procesamiento */
```

Se debe llamar a la función **dbresults()** para cada comando en batch, regresa un mensaje de SUCCEED si existen resultados a procesar, o un FAIL con un error y NO_MORE_RESULTS muy útil cuando múltiples comandos han sido enviados en un batch. La codificación típica para esta función es:

```
while ( dbresults(dbproc) != NO_MORE_RESULTS )
        dbprrow(dbproc)       /* y continuo con el programa */
```

Normalmente lo que se tiene que hacer con los resultados es referenciar los datos, o moverlos a variables internas para formatearlas y procesarlas. Para un acceso más rápido se puede utilizar la función **dbprrow()** para desplegar todos los resultados en la pantalla.

Terminando la aplicación cliente

Para terminar una sesión de nuestro programa cliente con el servidor SQL, se utiliza la función **dbclose()** la cual limpia la DBPROCESS especificada y libera el espacio que se estaba ocupando, y la función **dbexit()** termina con todos los DBPROCESS que estén abiertos para esta aplicación, este es el equivalente funcional de utilizar un **dbclose** para cada DBPROCESS aún abierto en la aplicación, aunque **dbexit** no sale del programa.

Ejemplo de un programa cliente en Sybase:

```
#include <sybfront.h>
#include <sybdb.h>
#include <syberror.h>
main ()
{
 DBPROCESS  *dbproc;
 LOGINREC   *login;
 RETCODE    return_code;

 if (dbinit() == FAIL)
     exit(ERREXIT);
 login = dblogin();
 DBSETLUSER(login,"curso01");
 DBSETLPWD(login,"curso");
 dbproc = dbopen(login,NULL);
 if (dbproc == NULL)
     exit(ERREXIT);
 dbcmd (dbproc,"select * from authors");
 if (dbsqlexec(dbproc) == FAIL)
 {
   dbexit();
   exit(ERREXIT);
 }
 while (dbresults(dbproc) != NO_MORE_RESULTS )
    dbprrow(dbproc);
 dbexit();
 exit(STDEXIT);
}
```

Para compilarlo, se recomienda utilizar la herramienta make de Unix, donde el archivo Makefile sería el siguiente:

```
# Archivo makefile para los programas clientes de Sybase
#
# Cambia las siguientes definiciones de acuerdo a tu sistema
SYBASE   = /opt/sybase
INCDIR   = $(SYBASE)/include
LIBDIR   = $(SYBASE)/lib
HEADERS  = $(INCDIR)/sybfront.h  \
           $(INCDIR)/sybdb.h
DBLIBS = $(LIBDIR)/libsybdb.a
INCLUDE  = -I. -I$(INCDIR)

all: prueba

prueba: $(HEADERS) prueba.c
        gcc $(INCLUDE) prueba.c $(DBLIBS) -lm -lnsl -o prueba
```

Pero eso no es todo, se debe también definir la variable de ambiente **SYBASE** y **DSQUERY** (nombre del servidor que se encuentra en el archivo **interfaces**) de la siguiente forma:

```
% setenv SYBASE /opt/sybase
% setenv DSQUERY SYBASE
% make prueba
gcc -I. -I/opt/sybase/include prueba.c /opt/sybase/lib/libsybdb.a -lm -lnsl -o
prueba
%
```

En este caso, el resultado de la última instrucción, genera el archivo ejecutable llamado *prueba*. Por lo tanto, se puede invocar desde la línea de comandos y el resultado en pantalla será el de la instrucción *select * from authors*.

Sentencias SQL con parámetros

Sintaxis:

dbfcmd(dbproc,cadena_del_comando,argumento1,argumento2, . . .)

Para cada ocurrencia % en la cadema_del_comando, convierte y sustituye su valor por el siguiente en la lista de argumentos. El máximo número de parámetros en cada llamada es de 8, también puede combinarse con llamadas a la función **dbcmd()**. Ejemplo:

```
char *table="ventas";
char *storid="6380";
int qty=50;
. . . .
dbfcmd(dbproc,"select * from %s",table);
dbfcmd(dbproc," where qty > %d",qty);
dbfcmd(dbproc," or stor_id=%s",storid);
```

Procesando resultados

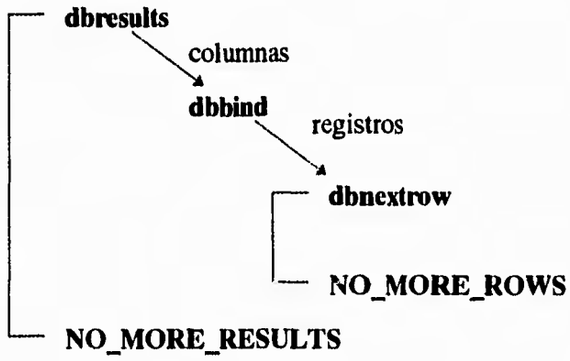
Vimos la función **dbresults()** para obtener los resultados que devuelve el servidor. Sin embargo, antes de enviarlos a pantalla, podemos procesar cada renglón y manipularlo individualmente. Una de las estrategias para este procesamiento es usando la función **dbbind()**. Los objetivos que se persiguen con esto son:

1. Establecer un enlace entre los datos devueltos por el servidor y las variables del programa cliente.
2. Convertir los datos de un tipo en la base de datos, a cualquier otro tipo dentro del programa.
3. Probar la ausencia o presencia de resultados.

Suponga que tratamos de ejecutar la siguiente instrucción SQL en un programa cliente:

```
select columna1,columna2,columna3 from tabla
```

Tal vez nos interesa obtener un registro de la tabla. Para ello, la función **dbbind()** puede mapear cada columna por separado a variables del programa. Se debe hacer una llamada a esta función por cada columna y va acompañada a su vez de la función **dbnextrow()** para obtener registro a registro.



La función **dbbind()** siempre se llama después de **dbresults()** y antes de **dbnextrow()**. Su sintaxis es la siguiente:

```
dbbind(dbproc,# de columna,tipo_variable,longitud_variable,variable)
```

donde:

- # de columna: Es un entero correspondiendo a la columna del comando select. Por ejemplo:

select nombre,tel from empleado	nombre es la columna1
select tel,nombre from empleado	nombre es la columna2

- tipo_variable: Representa el tipo de la variable del programa, no el tipo de la columna en la tabla. Si este tipo difere del de la columna en la bases de datos, y la

conversión es legal, podemos convertir datos de un tipo a otro. Los tipos de variables más comunes en la programación de clientes en C son:

Tipo_variable en dbbind	Tipo de dato en el programa en C	Tipo de dato en el servidor SQL
INTBIND	DBINT	INT
CHARBIND	DBCHAR Rellena de blancos y no termina con nulo	CHAR
STRINGBIND	DBCHAR Rellena de blancos y termina con nulo	CHAR
NTBSTRINGBIND	DBCHAR No rellena de blancos y termina con nulo	CHAR

Tabla x. Tabla de conversión de tipos de un programa cliente

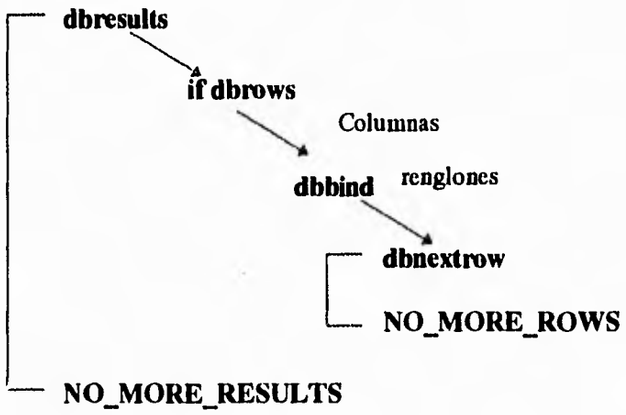
- longitud_variable: Puede ser 0 si la variable es lo suficientemente grande para almacenar la longitud de la columna convertida. Si se especifica n, solo toma los primeros n caracteres. Para los enteros, este valor es siempre 0.

- variable: Es el nombre de la variable definida en el programa en C. Los tipos más comunes son los que se presentaron en la tabla anterior: DBINT y DBCHAR. Para variables tipo entero, es necesario hacer la asignación especificando la dirección de la misma. Por ejemplo: &variable.

La función **dbnextrow()** debe llamarse por cada registro de la columna mapeada, puede devolver cualquiera de los siguientes valores:

NO_MORE_ROWS	No hay más registros
REG_ROW	Todavía hay registros
FAIL	Error

También podemos hacer uso de la función **dbrows()** antes de usar **dbbind()** para verificar que existen resultados a procesar. dbrows puede devolver SUCCEED si hay renglones. De esta forma, tendríamos algo así:



Ejemplo de un programa cliente utilizando las funciones **dbbind()** y **dbnextrow()**:

```
#include <sybfront.h>
#include <sybdb.h>
#include <syberror.h>

main ()
{
 DBPROCESS  *dbproc;
 LOGINREC   *login;
 RETCODE    return_code;
 char titulo[80];
 int precio;

 if (dbinit() == FAIL)
     exit(ERREXIT);
 login = dblogin();
 DBSETLUSER(login,"curso12");
 DBSETLPWD(login,"curso");
 dbproc = dbopen(login,NULL);
```

```
if (dbproc == NULL)
    exit(ERREXIT);
printf("\n Escribe el nombre de un titulo: ");
gets(titulo);
dbcmd (dbproc,"select title,price from titles");
dbfcmd (dbproc," where title=\"%s\"",titulo);
if (dbsqlexec(dbproc) == FAIL)
{
  dbexit();
  exit(ERREXIT);
}
while (dbresults(dbproc) != NO_MORE_RESULTS )
  dbprrow(dbproc);
printf("\n\n Escriba el precio: ");
scanf("%d",&precio);
dbfcmd (dbproc,"update titles set price=%d",precio);
dbfcmd (dbproc," where title=\"%s\"",titulo);
if (dbsqlexec(dbproc) == FAIL)
{
  dbexit();
  exit(ERREXIT);
}
dbexit();
exit(STDEXIT);
}
```

6.3 Sistema de inscripciones a cursos implantado en Kioscos de Información

El sistema de inscripciones a cursos implantado en kioscos de información no es solo tener una computadora con multimedia (imagen, video, texto, etc.) y que el público pueda accederla. El objetivo que se persigue es lograr estructurar en un solo ambiente, varias herramientas que no solo trabajan a nivel de computadoras personales, sino aquellas más poderosas que corren en sistemas como X Window.

Ya vimos que Mosaic puede interactuar con programas externos conocidos como CGI's, los cuales pueden estar escritos en lenguaje de programación C, shell

de Unix y otros. Ahora, vamos a utilizar *formas* en hipertexto (HTML) para diseñar por ejemplo, una solicitud de inscripción a algún curso.

En el kiosco, el usuario podrá llenar una solicitud tecleando sus datos y una vez llena la forma, la información capturada será enviada a un programa cliente en Sybase. Este programa se encargará de insertar los datos proporcionados por el usuario desde la terminal, a una base de datos remota. De esta forma, tenemos que desarrollar una aplicación donde involucremos a Mosaic como ambiente gráfico a manera de *front end* y una base de datos en Sybase. Implícitamente, el usuario estará ejecutando operaciones con comandos SQL tales como insertar, borrar y actualizar datos en una tabla, simplemente utilizando botones desde Mosaic en la pantalla *touchscreen*. El proceso será transparente, el usuario nunca se va a enterar que los datos que proporcionó viajarán en la red y llegarán a una máquina remota donde se encuentra la base de datos.

Actualmente, Mosaic puede correr en Windows de Microsoft. Por otra parte, tenemos la versión comercializada del mismo software llamado NetScape (vea sección 5.5), el cual también corre en Windows pero en una forma más eficiente que Mosaic. Además, ambos programas (Mosaic y NetScape) se están difundiendo en la Red Internet en forma impresionante. La gente que tiene sus computadoras personales y estaciones de trabajo conectadas a la red, pueden disponer de los servicios que proporciona la comunidad de la World Wide Web (WWW). Uno de esos servicios será precisamente el sistema de inscripción a cursos.

Si el lector cuenta con Mosaic y una conexión a Internet, puede inscribirse sin necesidad de acudir al kiosco de información. El problema es que no toda la gente dispone de una computadora conectada a la red y de un equipo de hardware más o menos eficiente para soportar las cantidades de información que Mosaic requiere manejar (sobre todo cuando se tratan de imágenes y video).

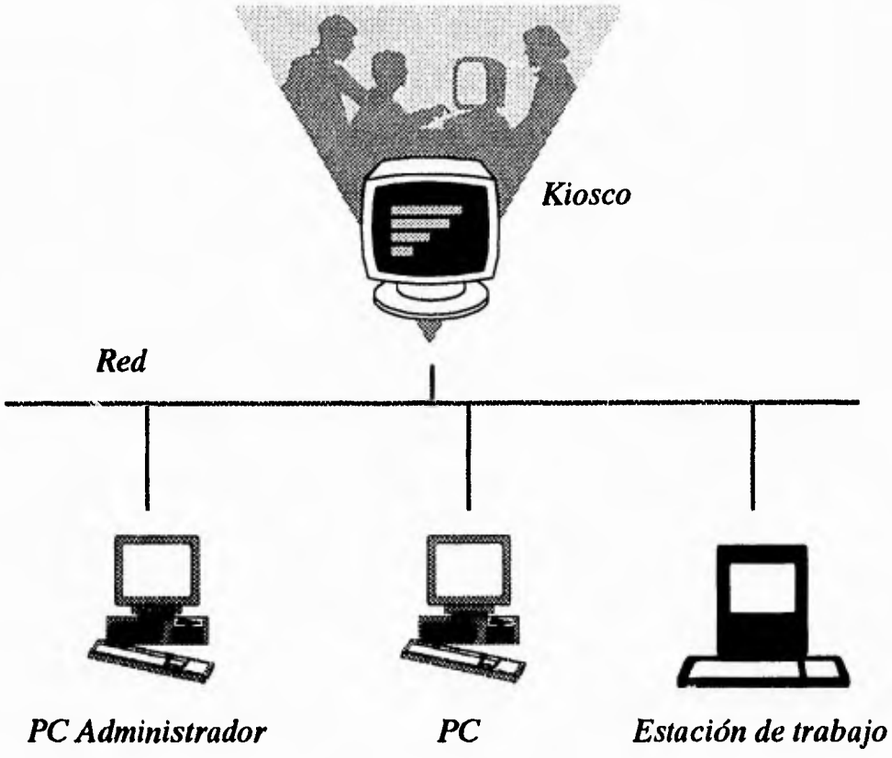


Figura 6.7 Usuarios del Sistema de Inscripciones

Con los kioscos, ponemos a disposición de la comunidad universitaria y público en general, un servicio de la Internet. Asimismo, aprovechamos recursos de otras máquinas que se encuentran dentro de la UNAM. Pero el sistema no termina ahí, esa es solo la primera parte, porque ahora necesitamos realizar todo un diseño para la administración de los cursos. En esta parte, contamos con los administradores, que son las personas encargadas de programar el calendario, inscribir alumnos, generar constancias, reportes, etc.

El problema ahora es, diseñar el proceso de inscripción a cursos y modelarlo para hacer uso de un manejador de Base de datos como Sybase (debido a las ventajas que presenta y que fueron discutidas en la sección 6.2.1), crear tablas es el siguiente paso. Una vez hecho esto, el sistema funcionará y los administradores

deberán estar capacitados para usar SQL y Sybase cada vez que requieran hacer alguna operación en la base de datos. Pero, ¿porqué no diseñar todas esas consultas SQL desde Mosaic?. Al hacerlo, los administradores del sistema desde su PC o estación de trabajo (utilizando Mosaic o NetScape) pueden realizar las tareas de administración sin tener necesariamente conocimientos de Sybase y SQL.

6.3.1 Descripción del proceso

La Dirección de Cómputo para la Administración Académica es una dependencia que cuenta con departamentos que se encargan del desarrollo de sistemas, soporte técnico a equipos y recursos humanos. El proceso de programación de calendarios e inscripción a cursos es actualmente una tarea tardada. Por ejemplo, para programar un curso, primero hay que revisar si existe el personal para impartirlo, luego si existe aula disponible y equipo con los recursos suficientes que soporten el software requerido, buscar un horario adecuado es el siguiente paso y se debe evitar programar dos cursos al mismo tiempo y en la misma aula. Al manejar Bases de Datos, podemos manipular la cantidad de información que se requiere, en base a tablas controlamos horarios, períodos, aulas, profesores, cursos y alumnos.

Actualmente el proceso de inscripciones consta de los siguientes pasos:

- El calendario de cursos semestral, se planea con un tiempo considerable de anticipación.
- Se programa un curso con el horario, aula y profesor adecuados.
- La inscripción de alumnos se realiza acudiendo a las oficinas de la dependencia con la constancia que acredite o que indique tener los conocimientos suficientes

para tomar el curso. Se considera que el alumno cumple con el requisito anterior, si la constancia tiene una calificación mayor o igual a 8.

- El alumno realiza el pago correspondiente al curso.
- Se genera una lista de alumnos por cada curso.
- Al finalizar cada curso, se reciben calificaciones de los alumnos por parte de los profesores.
- En base a la lista, se generan las constancias (tómese en cuenta que se generan tantas listas de calificaciones como cursos se imparten), en el peor de los casos si por alguna razón estas calificaciones se extravían, hay que hacer una nueva petición al profesor para que pueda proporcionar una copia, aunque él no está obligado a conservarlas. Por lo tanto, si el profesor las pierde, la información nunca será obtenida o no será verídica. Para que la DGSCA pueda expedir una constancia, el alumno deberá aprobar el curso con una calificación mayor o igual a 8. Actualmente la generación de constancias es a través de una persona que se encarga de imprimirlas con los datos del alumno.
- Un alumno no se puede inscribir a un curso sin haber acreditado el curso antecedente (seriación).
- Se conservan las listas de alumnos con sus respectivas calificaciones.

Todo este proceso es manual y tardado, nuestro trabajo es automatizarlo llegando inclusive a la generación de constancias vía red, con ayuda de impresoras de alto volumen instaladas también en la DCAA, y de las cuales hablaremos más adelante.

Entre los procedimientos de inscripción a los cursos de la DCAA que debemos automatizar se encuentran: inscripciones, consultas, calificaciones, reportes, control de alumnos, profesores, cursos, aulas y generación de constancias.

El sistema será utilizado por dos tipos de usuarios: el que opera el sistema en modo administrador, que es aquel usuario que puede programar cursos, horarios, profesores, etc., y los usuarios finales, que son los que se inscriben y consultan los cursos del calendario. El proceso de nuestro sistema, se describe a continuación.

Usuarios Finales

El usuario puede inscribirse de la siguiente forma: Primero, deberá usar la terminal del Kiosco. En la pantalla se le pedirá su identificación de usuario (en este caso, utilizamos su RFC como clave de acceso) para saber si existe o no en la Base de Datos, estos datos se encuentran en una tabla. Si el usuario se inscribe por primera vez, en la pantalla aparecerá una solicitud de inscripción (usando *formas* en HTML), donde se le pedirán sus datos personales y una vez capturados, se actualizará la Base de Datos para ser dado de alta en el sistema y tener derecho a inscripción.

Una vez dado de alta el usuario o haber confirmado la existencia de su registro, se procede a mostrar el calendario de cursos al que se puede inscribir. Obviamente, el administrador se encarga de programar el calendario y automáticamente los cursos programados son presentados al usuario a manera de consulta, la información mostrada es el nombre del curso, período y horario. La inscripción del usuario se almacena en una tabla temporal, porque lo que hace, es reservar su lugar en algún curso, y deberá confirmar su inscripción cuando lo pague en las oficinas de la dependencia. Hasta ese entonces, estará formalmente dado de alta en ese grupo como alumno inscrito. Ese mismo registro será actualizado nuevamente cuando el curso termine, con la calificación que el profesor proporcione.

En el proceso de inscripción, también se toman en cuenta los antecedentes del alumno. Si el curso al que se desea inscribir, requiere algún otro antecedente, automáticamente el sistema debe averiguar si lo ha aprobado con una calificación mayor o igual a 8, de no ser así, la inscripción no procede. Sin embargo, debe considerarse otro aspecto importante. Pudiera ser que el alumno tiene constancias expedidas por otra dependencia que no sea DGSCA, en este caso, la inscripción procederá cuando el alumno las presente en la DCAA y sus calificaciones queden registradas en otra tabla únicamente para fines estadísticos y considerar que tiene aprobado el curso antecedente.

Usuarios Administradores

La administración del sistema será en forma sencilla y sin mayores complicaciones. El usuario administrador es el encargado de programar el calendario, asignar aulas y profesores, emisión de reportes y generación de constancias. Todos estos procedimientos se podrán hacer con ayuda de Mosaic, sólo que aquí no es necesaria la estación de trabajo (se puede utilizar una computadora personal).

En este caso, el administrador no podrá programar el calendario semestral si no tiene cursos, aulas, períodos y horarios dados de alta en la Base de Datos. Por lo tanto, las primeras tareas serán: dar de alta cursos, períodos, horarios y aulas. Para hacerlo, el sistema desde Mosaic/NetScape le presentará *formas* (hechas en HTML) a ser llenadas por el usuario en cada caso, los datos capturados serán enviados a la Base de Datos vía red.

Una vez que tiene información en la Base de Datos, puede comenzar con la programación de los cursos. En este proceso, el sistema se comportará en forma dinámica. Esto es, para programar un curso se requiere por ejemplo: nombre del

mismo, período, horario y aula. Un cliente en Sybase se encargará de averiguar si existe la información requerida en la Base de datos, de ser así, la presentará al administrador en forma de botones y menús desplegables (también usando lenguaje HTML) con el fin, de que el usuario pueda elegir desde Mosaic, los campos necesarios para la programación del curso. De lo contrario, lo notificará enviando algún mensaje de error.

Los cursos programados desde Mosaic serán los mismos que los alumnos podrán consultar en el Kiosco (dado que toda la información se encuentra en la Base de Datos y a través de clientes se puede acceder a ella).

Los procesos de bajas y cambios en períodos, horarios, etc., será en forma similar al de altas. Se consulta a la Base de Datos y se le presenta al usuario, los registros que puede borrar o cambiar. En general, el manejo de la información será muy sencilla desde Mosaic, el administrador simplemente usará el *ratón* para seleccionar botones y menús que se generan dinámicamente desde programas clientes.

6.3.2 Diseño del sistema

Tomando en cuenta la descripción del proceso explicado en la sección anterior, el diseño consiste en elaborar los diagramas de flujo de información y las tablas del sistema necesarias, donde se debe considerar la normalización de las mismas.

El diagrama correspondiente a las inscripciones de los alumnos, es el siguiente:

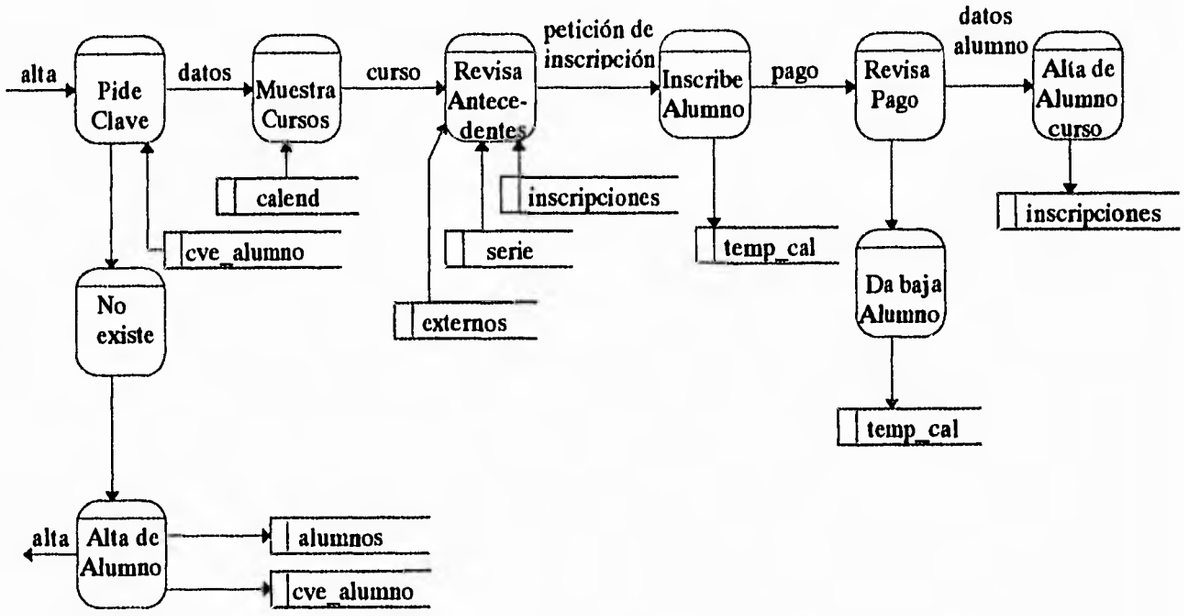


Figura 6.8 Diagrama de flujo para inscripciones

Para la programación de cursos, se sigue el siguiente diagrama:

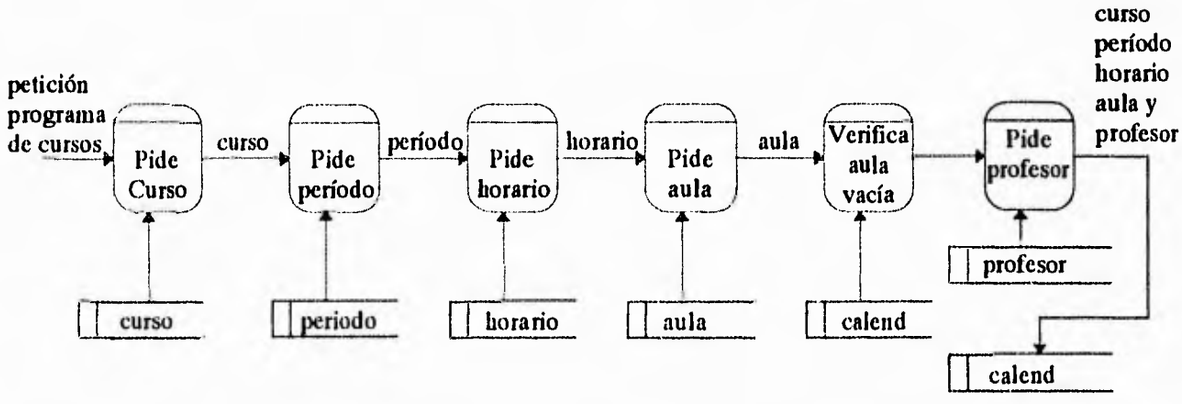


Figura 6.9 Diagrama de flujo de Programación de Cursos

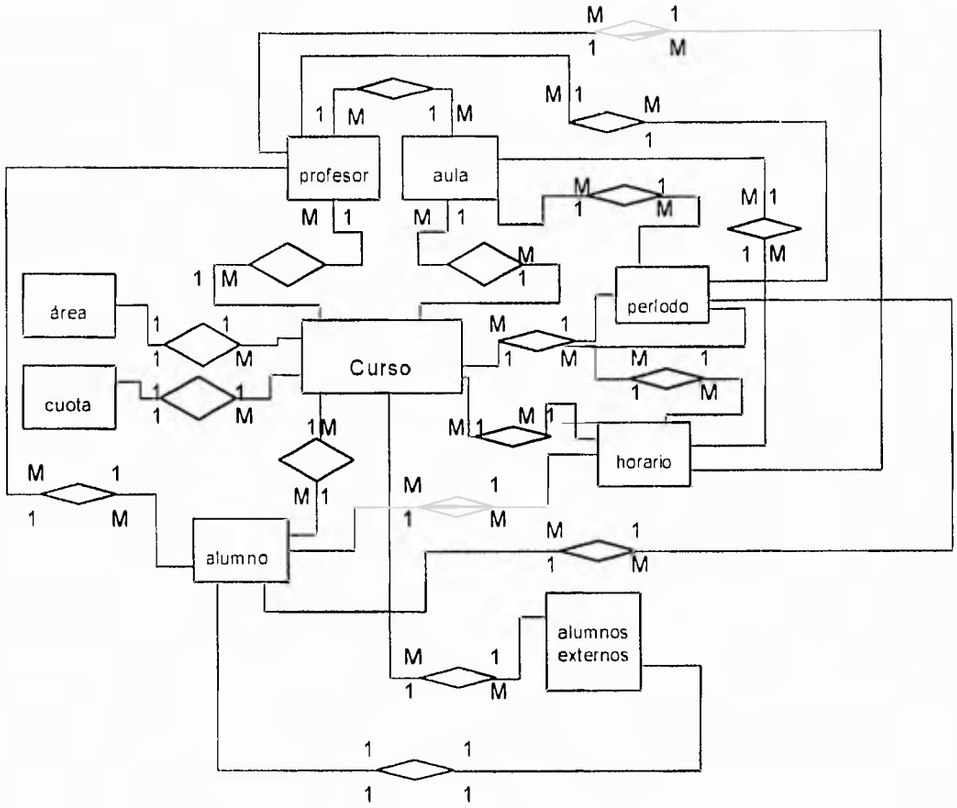


Figura 6.10 Diagrama de Entidades y Relaciones para el sistema de inscripciones

Esqueletos de las tablas del modelo de la Base de datos:

ALUMNOS (clave alumno, día, mes, año, dirección, teléfono, escolaridad, tipo de alumno)

CLAVES_ALUMNOS (clave alumno, nombre, apellido paterno, apellido materno)

AREA (clave área, área)

CUOTAS (clave cuota, cuota)

CURSO (clave curso, curso, clave área, clave cuota)

HORARIO (clave horario, hora inicio, hora fin)

PERIODO (clave período, dia inicio, mes inicio, dia fin, mes fin, semestre)

AULAS (clave aula, aula, cupo, comentarios)

SERIE (clave curso, clave curso 2, clave curso 3, clave curso 4, clave curso 5)
PROFESOR (clave profesor, nombre, dirección, teléfono, comentario)
EXTERNOS (clave alumno, clave curso, calificación, comentario)

Tablas derivadas del Modelo de la Base de Datos:

CALEND (clave curso, clave período, clave horario, clave profesor, clave aula)
INSCRIPCIONES (clave curso, clave período, clave horario, clave alumno, clave profesor, calificación)
TEMP_CAL (clave alumno, clave curso, clave período, clave horario)

Después de diseñar la base de datos, se procede a la creación de la misma. En las siguientes secciones, explicaremos el desarrollo de las interfaces entre Mosaic y la Base de Datos.

6.4 Desarrollo

Una vez que se tiene el modelado del sistema, pasamos ahora al desarrollo e implantación del mismo. Comenzaremos primero, con la creación de las tablas en la Base de Datos en Sybase, con esto, podemos ya realizar operaciones tales como: insertar, modificar y borrar datos en las tablas con las operaciones básicas del lenguaje SQL. Y recuerde, cada sentencia SQL que se utilice en la manipulación de las tablas, puede ser implantada en un programa cliente escrito en lenguaje C, tal como se mencionó en la sección 6.2.2.

El siguiente paso, es diseñar las *formas* en Mosaic para la captura de la información, donde debemos incluir campos para que un programa externo (CGI) pueda insertar los datos en una tabla de nuestra Base de datos.

La parte complicada es la siguiente tarea, aquí debemos diseñar las interfaces entre Mosaic y clientes en Sybase a través de CGI's. Si tengo mis programas clientes en Sybase, puedo insertar ese código en un CGI y de esta forma, interactuar desde Mosaic con programas de aplicación en UNIX e incluso con otros servidores hacia nuestra base de datos.

El objetivo del diseño de sistemas con Mosaic, es el de compartir información con otros usuarios de la Red Internet. Sin embargo, habrá ocasiones donde será necesario restringir el acceso a determinados documentos. En las siguientes secciones, hablaremos a detalle acerca de este aspecto de seguridad, y cómo fue solucionado este problema al estar diseñando el sistema de inscripciones.

Sybase está instalado en otra estación de trabajo (no la que compone al Kiosco), así que haremos uso de una base de datos remota, por lo que el problema se vuelve aún más interesante. Cabe mencionar que el sistema de inscripciones será implantado en la estación de trabajo donde Sybase está instalado, aunque no necesariamente debe ser así, recuerde que los clientes en Sybase son portables entre máquinas (siempre y cuando tengan la misma arquitectura y Sistema Operativo) tal como se discutió en la sección 6.2.2. La ventaja de hacer esto, es que podemos compartir el mismo sistema de inscripción a cursos del Kiosco, con otros usuarios conectados a la Red. Las transacciones del Kiosco en este caso, serán remotas y transparentes para el usuario.

El diseño del sistema lo realizaremos en la estación de trabajo SparcStation 10 de Sun Microsystems y que forma parte del equipo con el que cuenta la DCAA, también tiene instalada la versión 4.9.2 y System X de Sybase. El sistema de administración de los cursos se realizará pensando en una computadora personal

conectada a la Red utilizando NetScape, donde el usuario administrador podrá, desde su oficina, realizar los cambios necesarios en su calendario. Para este caso, no es indispensable la estación de trabajo, la versión 1.1 de NetScape para Windows de Microsoft cumple con las características deseadas tal como lo veremos en las siguientes secciones.

6.4.1 Creación de las tablas en SYBASE

La forma de crear las tablas en SYBASE fue utilizando una herramienta en SYBASE llamada Data Workbench (dwb) la cual presenta una ventana como interfaz con el servidor SQL. Nos permite ejecutar comandos SQL en forma sencilla y rápida, así que podemos generar las tablas de una base de datos desde una ventana. Los comandos que más se utilizaron para la manipulación de estas tablas fueron: *select * from nombre_tabla*, *create table*, *insert nombre_de_la_tabla*, *delete nombre_tabla* y *drop nombre_de_la_tabla*. Por ejemplo, para crear una tabla llamada *alumnos* se tiene que teclear la siguiente instrucción: *create table*, y como argumentos tiene:

1. El nombre de la tabla
2. Los nombres de las columnas
3. El tipo de dato de cada columna
4. nulos permitidos en cada columna

En el proyecto se creó una tabla llamada *alumnos* de la siguiente manera:

```
create table alumnos
(clave char(3) notnull,
 dia   smallint  null,
 mes   char(10)  null,
```

year smallint null,
direccion char(80) null,
telefono char(10) null,
escolaridad char(15), tipo_alumno char(30))

En caso de no especificar si el campo es nulo (null) o notnull, el valor por default será notnull. El número máximo de columnas posibles en una tabla es de 250.

En caso de que la tabla no cumpla con nuestros requerimientos, es posible eliminarla mediante el comando *drop nombre_de_la_tabla*. La forma de eliminar la tabla del ejemplo anterior será mediante la instrucción: *drop alumnos*, aunque es importante señalar que una vez que una tabla ha sido eliminada, se borra con toda la información y no hay forma de recuperarla.

Una vez que la tabla ha sido creada, es posible examinar su estructura mediante el comando *sp_help*, en este caso, para verificar la estructura de la tabla *alumnos* se teclea el comando *sp_help alumnos*, y a continuación se desplegará la fecha y hora de creación de la tabla, el nombre de la columna, el tipo de datos que pertenece a cada nombre de la columna, su longitud y un nombre por defecto en un dado caso. Para poder insertar valores a la nueva tabla se hará mediante el comando *insert*. Para el ejemplo anterior, si se desea insertar valores en la tabla *alumnos*, se hará mediante este comando de la siguiente forma:

insert alumnos values ('123', '12', '12', '95', 'San Diego 32',
'3982007', 'Pasante', 'sobresaliente')

Usando procedimientos similares, se fueron creando las tablas en la base de datos del sistema en SYBASE.

Los tipos de datos más utilizados en la base de datos y en general en todas las tablas fueron: *char* (caracter), *smallint* (entero numérico) y *text* (texto), ya que las restricciones de los campos y la forma en que estos pudieran ser manipulados, se manejó al momento de programarlos como clientes en SYBASE en lenguaje C.

6.4.3 Generación de formas en HTML

Para la realización de nuestro sistema, fue necesario el uso de *formas* en HTML. Estas se utilizaron básicamente para la elaboración de pantallas que nos ayudarán a manipular la información de cualquier usuario que tenga interés en tomar un curso de cómputo en la DCAA. En esta parte del proyecto se tuvieron que aplicar todos los conocimientos relativos a la generación de *formas* en HTML (ver sección 4.6.1), para que en el kiosco el usuario pueda observar pantallas desde las cuales él mismo se pueda inscribir a un curso de manera sencilla, y pueda observar en ese momento el calendario de cursos de la dependencia. El primer documento HTML que se va a presentar cuando un usuario se quiera dar de alta (en su código fuente) estará representado de la siguiente manera:

```
<title>Sistema de Inscripci&oacute;n a Cursos</title>
<pre>
<h1>
  Universidad Nacional Aut&oacute;noma de M&eacute;xico
</h1>
</pre>
<pre>
<a href="/presenta-dgsca/DCAA.html">
<h2>
  Direcci&oacute;n de C&oacute;mputo para la Administraci&oacute;n
Acad&eacute;mica </a>
</h2>
</pre>
<pre>
<h1>
        Sistema de Inscripci&oacute;n a cursos
</h1>
</pre>
```

En esta parte del documento se está especificando el título de la forma y el encabezado de la misma, se puede notar que las etiquetas varían en algunas partes

del encabezado, siendo algunas de la forma <H2> y otras de la forma <h3>; esto sólo se refiere al tamaño de letra que tomarán cada uno de los encabezados, siendo en <h2> la especificación de un tamaño de letra más grande que el marcado con la etiqueta de <h3>. Cabe aclarar que es indistinto que se especifique la letra "h" con mayúscula o minúscula.

```
<h2>
Bienvenido al sistema de inscripci&oacute;n a cursos.
Por favor coloque su dedo sobre el campo de RFC (hasta que el
cursor este intermitente) y despu&eacute;s tecleelo, por ejemplo:
BAGF-681019 </h2>
```

En esta parte se muestran las instrucciones de cómo el usuario tiene que entrar al sistema utilizando la pantalla *touchscreen*, y de como deberá introducir su RFC, ya que será la llave principal en las tablas para el manejo de todos los alumnos.

```
<form method="POST" action="http://tzetzal.dcaa.unam.mx/cgi-bin/usuario1">
```

Aquí se especifica el método de la *forma* tipo POST, y el URL/CGI asociado a la misma será http://tzetzal/cgi-bin/usuario1.

```
<pre>
RFC: <input size=11 maxlength=11 name="RFC">

 <input type="submit" value="Aceptar">

 <input type="reset" value="Borrar RFC"></pre>
<p>
</form>
```

En esta sección se está definiendo el campo de entrada que nos va a permitir capturar el RFC del interesado, se puede observar que cada campo tiene un tamaño

de entrada, una longitud máxima y un nombre con el cual se va a reconocer al momento que lo reciba el CGI para su codificación.

```
Si usted no est&aacute; dado de alta en el sistema <a href="./forma.html">
presione aqu&iacute; </a><p>
<img
src="http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/line.blue.gif">
```

En esta parte se esta haciendo referencia a otro documento html llamado: <a href="./forma.html">, en el caso que el usuario no esté dado de alta en el sistema, y pueda llenar una solicitud de inscripción.

```
<a href="http://xel-ha.dcaa.unam.mx/home/kiosko/html/home_page.html"><h2>
Salir
del sistema</h2></a> <address>
```

Alternativamente si al usuario no le interesa seguir con el proceso de inscripciones, con un simple contacto a esta liga abandonará el sistema. La *forma* desde Mosaic se verá como la imagen de la figura 6.12.

La solicitud de inscripción a cursos, fue realizada en HTML, el archivo (en código fuente) que se utiliza para la **captura automática de los datos del usuario.** Estará representado de la siguiente manera:

```
<title>Solicitud de Inscripcion a Cursos</title>
<H2>  UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO </H2>
<h2> Direcci&oacute;n General De Servicios de C&oacute;mputo
Acad&eacute;mico</h2>
<h3> SOLICITUD DE INSCRIPCION A CURSOS</h3>
<hr>
<form method="POST" action="http://tzetzal.dcaa.unam.mx/cgi-bin/syb2">
```

Únicamente explicaremos los campos correspondientes a la *forma*. Aquí se especifica método tipo POST, y el URL/CGI asociado a la forma será http://tzetzal.dcaa.unam.mx/cgi-bin/syb2.

```
<h3> Datos Personales </h3>
<ul>
<li> Nombre: <input size=20 maxlength=20 name="nombre">
</li></ul>
<p>
<ul><li>
   Apellido Paterno: <input size=15 maxlength=15 name="ap_paterno">
</li></ul>
<ul><li>
   Apellido Materno: <input size=15 maxlength=15 name="ap_materno">
</li></ul>
<ul><li>Domicilio: <input size=70 maxlength=70 name="direccion"></li></ul>
<ul><li>Telefono: <input size=7 maxlength=7 name="tel"></li></ul>
<h4>Fecha de nacimiento mes/dia/a&ntilde;o :(Ejemplo 19/Julio/70)</h4>
<ul><li>
Dia: <input size=2 maxlength=2 name="dia">
</li></ul>
```

En esta sección se define una serie de campos de entrada que nos van a permitir capturar los datos del interesado, se puede observar que cada uno de estos campos tiene un tamaño de entrada, una longitud máxima y un nombre con el cual se va a representar al momento que llegue al CGI para su identificación.

```
<strong>Mes:</strong>
<input type=radio name=order value="Enero" CHECKED >Enero
<input type=radio name=order value="Febrero">Febrero
<input type=radio name=order value="Marzo">Marzo
<input type=radio name=order value="Abril">Abril
<input type=radio name=order value="Mayo">Mayo
<input type=radio name=order value="Junio">Junio
<input type=radio name=order value="Julio">Julio
<input type=radio name=order value="Agosto">Agosto
<input type=radio name=order value="Septiembre">Septiembre
<input type=radio name=order value="Octubre">Octubre
<input type=radio name=order value="Noviembre">Noviembre
<input type=radio name=order value="Diciembre">Diciembre <p>
```

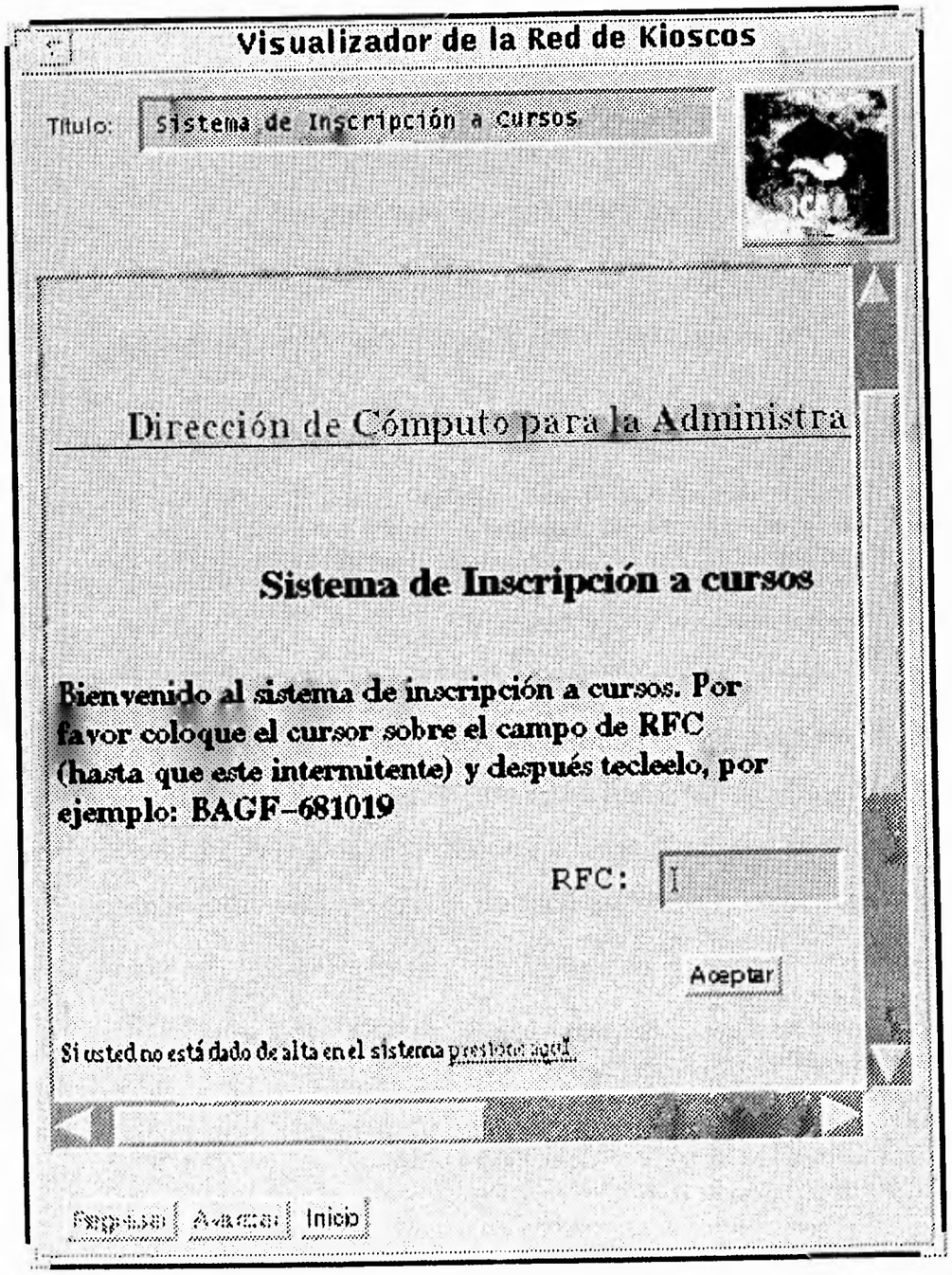


Figura 6.12 Pantalla de inscripción a cursos presentada en Mosaic

En esta parte del documento se utilizaron "botones booleanos" tipo *radio* los cuales nos están dando una entrada por default dependiendo del mes de nacimiento del usuario, ya que forzosamente en alguna de estas opciones quedará su elección sin tener la oportunidad de elegir alguna otra o dos al mismo tiempo. Nótese que la primera opción: <input type=radio name=order value="Enero" CHECKED >Enero está incluido el parámetro CHECKED, el cual está asignando por default el valor de enero a esta sección. Este tipo de botón está representado por un pequeño rombo, y para el caso del parámetro CHECKED hará que este pequeño rombo se ilumine haciendo notar al usuario que puede elegir cualquiera de las opciones del mes que pertenezca a su fecha de nacimiento. Sin embargo, aunque el usuario pueda elegir el botón, el CGI recibirá "Enero", "Febrero", etc., dependiendo del valor (*value*) del campo "order".

```
<ul><li>
A&ntilde;o : <input size=2 maxlength=2 name="year">
</li></ul>
<hr>
```

En esta parte de la forma se está definiendo el campo del año de nacimiento. También se puede observar que se está utilizando el elemento *&ntilde*; el cual nos permite definir la letra "ñ". También debemos recordar que las etiquetas <li> y <ul> nos sirven para especificar viñetas y sangrías respectivamente.

```
<ul><li>
<strong>Nivel Actual de Escolaridad :</strong>
```

La etiqueta hace que el texto se vea en negritas.

```
<input type=radio name=escol value="Preparatoria" CHECKED>Preparatoria
<input type=radio name=escol value="Profesional">Profesional
<input type=radio name=escol value="Pasante">Pasante
<input type=radio name=escol value="Titulado">Titulado
<input type=radio name=escol value="Posgrado">Posgrado
</li></ul>
```

En este caso se vuelve a utilizar el botón de elección tipo *radio*, sólo que en esta ocasión es para el caso de la elección del nivel de escolaridad del interesado. Note que el nombre del campo asociado ahora es "escol", mientras que en el caso anterior fue "order".

```
<ul><li> RFC:<input size=11 maxlength=11 name="rfc">
<hr>
```

En esta parte se define el campo RFC, con un tamaño de entrada de 11 caracteres y una longitud máxima del mismo valor.

```
Cuando termines de teclear tus datos, presiona aqui:
<input type="submit" value="Aceptar"> <p>
Para borrar los datos de la forma, presiona este bot&oacute;n:
<input type="reset" value="Borrar">
</form>
<hr>
```

Por último se definen dos botones. El primero tipo *submit*, hace que Mosaic pida al servidor el CGI identificado por el URL asociado a la *forma*. El segundo (tipo *reset*) para borrar la información capturada y devolver valores por default (si es que los hay) de la *forma* en caso de que el usuario se equivoque o quiera volver a comenzar el llenado de la *forma* nuevamente. La *forma* HTML desde Mosaic se verá como lo muestra la Figura 6.13.

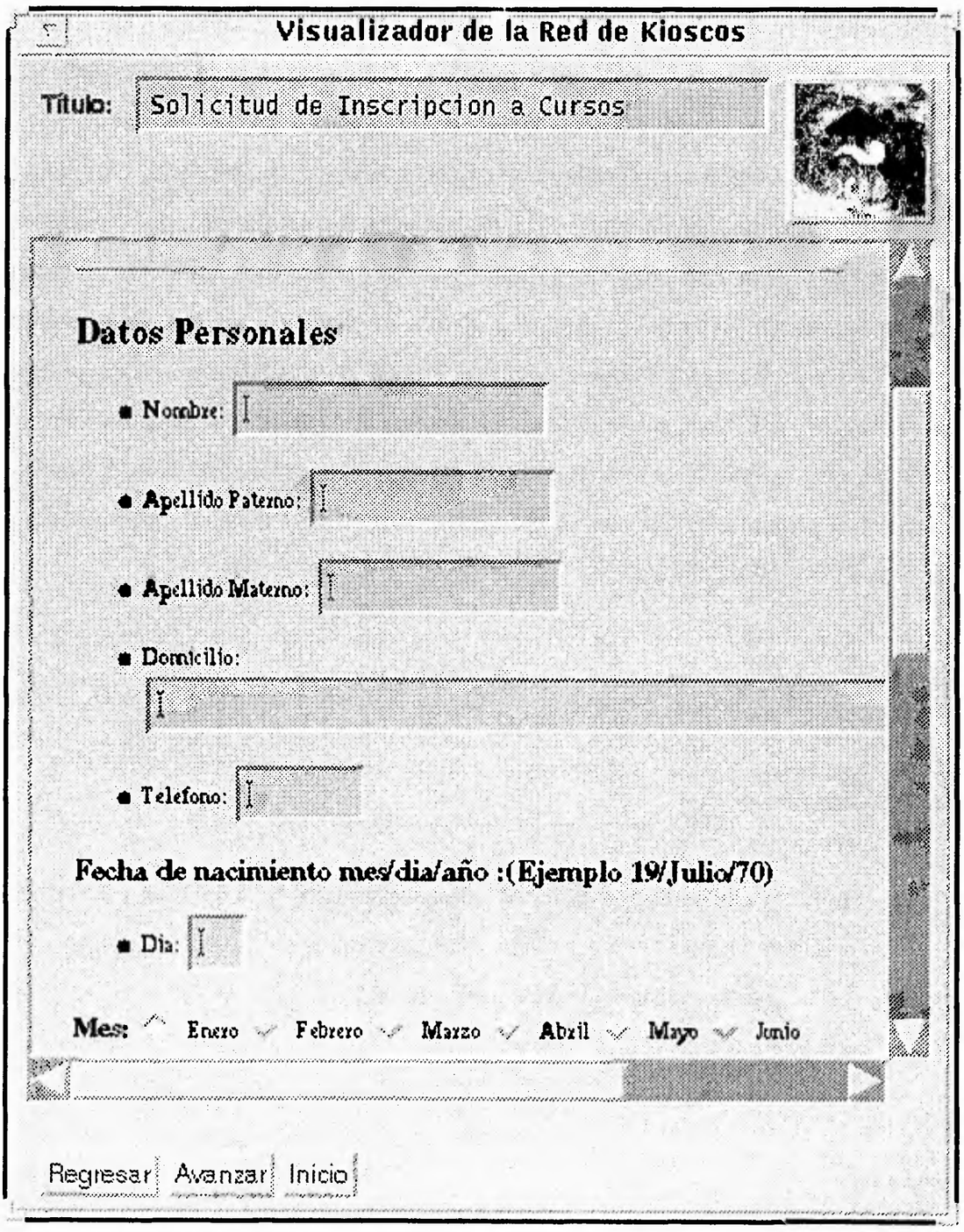


Figura 6.13 Solicitud de inscripción a cursos representada en un documento HTML

Existe otra parte del sistema manejada única y exclusivamente por el **administrador del mismo**, en ésta también se utilizaron *formas*. La principal diferencia entre las que se manejaron para el kiosco y para la parte del administrador del sistema es que en esta última se tiene más versatilidad con respecto al manejo de etiquetas, lo que trae como consecuencia que se tenga un manejo más complejo de los datos. Es el caso por ejemplo de la *forma* para dar de alta un aula, en donde se tuvieron que utilizar otro tipo de etiquetas que nos permitieran manejar la entrada de un texto, no importando su longitud donde se pudieran definir características particulares de la misma. Recuerde que desde la PC, el usuario usará Netscape como visualizador, así que las pantallas del sistema se realizaron pensando en una computadora personal, incluimos además, algunas instrucciones de HTML plus para hacer más presentables los menús. Se describirá a continuación el código fuente de la *forma* para dar de alta un aula:

```
<title>Aulas</title>
<center>
<H3>UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO </H3>
<h3> Direcci&oacute;n General De Servicios de C&oacute;mputo
Acad&eacute;mico</h3>
```

En esta parte se define el título, sólo que en este caso se agregó la etiqueta la que de manera automática ejecuta el centrado del título anterior de la *forma*. La etiqueta forma parte de HTML plus y es reconocida por Netscape más no por Mosaic.

```
<strong> AULAS </strong>
<hr>
</center>
<form method="POST" action="http://tzetzal.dcaa.unam.mx:87/cgi-bin/aula1">
<strong> Aula </strong>
<ul>
```

En esta parte se define nuevamente el método tipo POST, y el URL/CGI asociado a esta forma será http://tzetzal.dcaa.unam.mx:87/cgi-bin/aula1. Nótese que se está llamando a un nuevo servidor representado por tzetzal.dcaa.unam.mx:87 en el puerto 87, del cual hablaremos más adelante.

```
<textarea name="aula" cols=50 rows=10>
</textarea><p>
</ul></li>
```

Aquí se define una nueva modalidad para poder capturar un texto no importando su longitud, con la etiqueta <textarea>. El área de captura del texto estará definida por un rectángulo, y sus dimensiones son definidas por "cols=" para la altura y por "rows="para el ancho del mismo. Esta etiqueta se utilizó a consecuencia del uso de un tipo de dato que maneja SYBASE que es *text* el cual permite almacenar información en una columna de una tabla, no importando su longitud.

```
<hr>
<address>
<li><a href="./admonmenu.html">Regreso al men&uacute; principal </a><p>
<li><a href="http://tzetzal.dcaa.unam.mx:87/admonini.html">Salir del
sistema</a> <p>
</address>
```

Por último se define una opción alternativa para el caso que se quiera salir del sistema.

La forma visualizada en Mosaic estaría representada tal como lo muestra la figura 6.14.

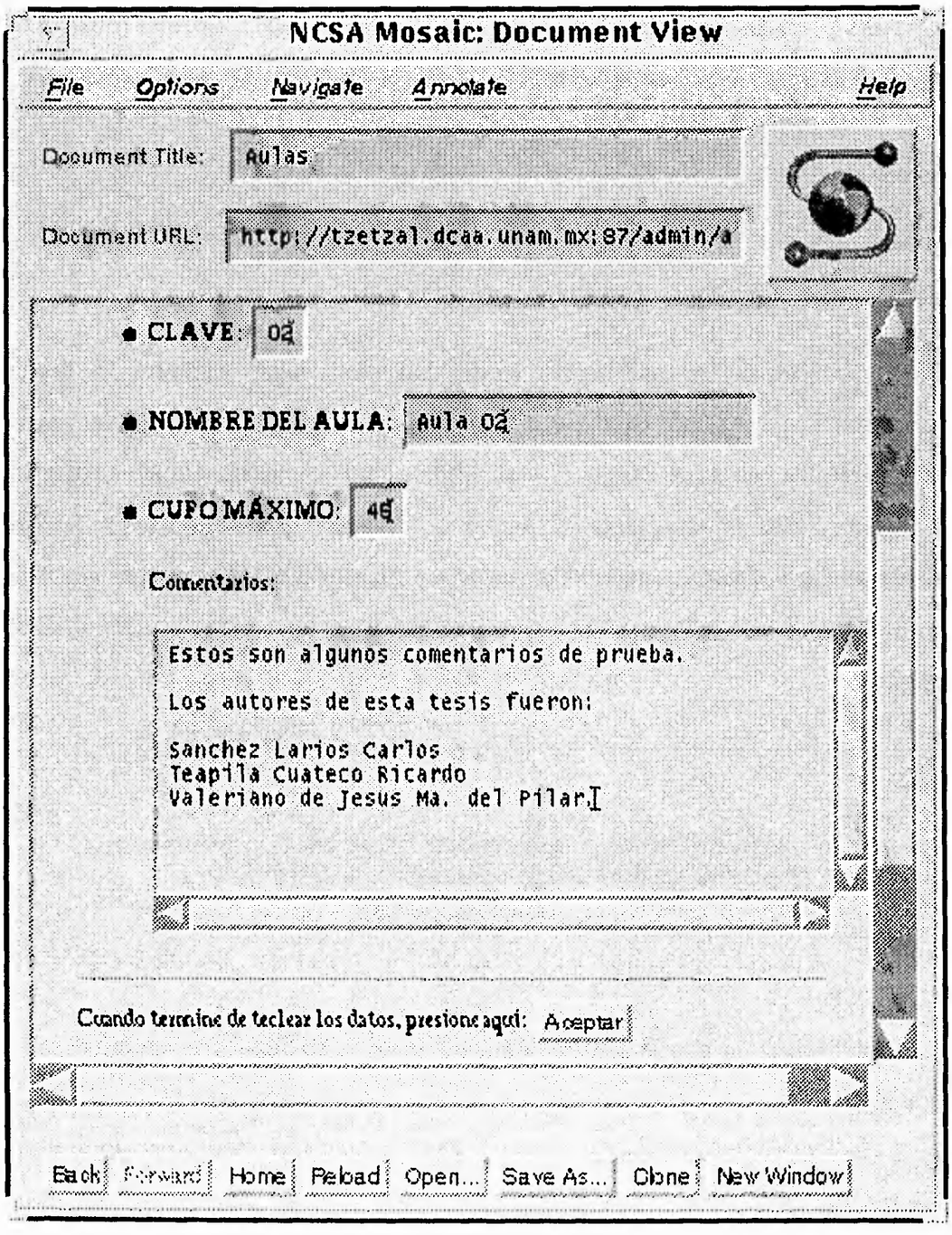


Figura 6.14 Representación de la forma: "alta de aulas" en Mosaic

También se incluyó una serie de *formas* que permitieron dar de alta, baja o cambiar la misma. A continuación se describirá una forma (en su código fuente) para poder dar de alta un área en el sistema:

```
<strong> &Aacute;REAS</strong>
</center>
</strong>Digite el nombre y clave de la nueva &aacute;rea:</strong>
<form method="POST"
action="http://tzetzal.dcaa.unam.mx:87/cgi-bin/area1"><pre>
```

El URL asociado a esta forma es http://tzetzal.dcaa .unam.mx:87/cgi-bin/aula1.

```
CLAVE: <input size=2 maxlength=2 name="clave"><p>
&Aacute;REA: <input size=50 maxlength=50 name="area"> <p>
```

En esta sección se definen dos campos de entrada que nos van a permitir capturar los datos del aula, se puede observar que cada uno de estos campos tiene un tamaño de entrada, una longitud máxima y un nombre con el cual se va a representar a ese campo al momento que llegue al CGI para su identificación.

```
        <input type="submit" value="Aceptar"><p>
        <input type="reset" value="Borrar">
</pre>
<hr>
```

Aquí se definen dos botones los cuales ya fueron explicados anteriormente. La forma visualizada en Mosaic estaría representada como lo muestra la figura 6.15.

Todas las demás *formas* restantes se realizaron de manera similar a las ya explicadas anteriormente salvo algunos detalles.

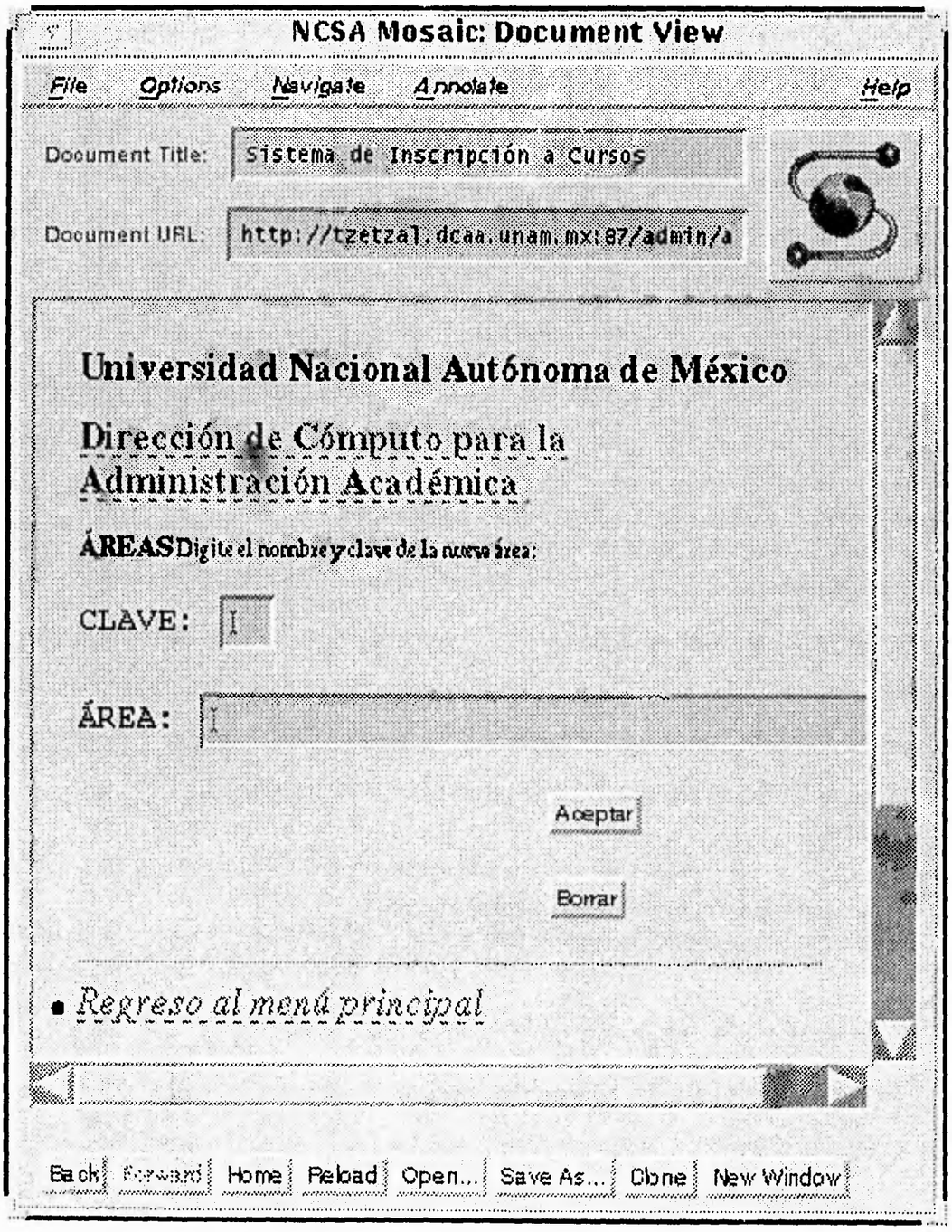


Figura 6.15 Representación de una forma en Mosaic para dar de alta un área

6.4.3 Programación de las interfaces entre Mosaic y clientes en Sybase a través de CGI's

En la sección 4.6.2 definimos el concepto de CGI (Common Gateway Interface) como programas que permiten comunicar a Mosaic con otro tipo de aplicaciones externas, como ejemplos tenemos a los scripts en UNIX o programas en lenguaje C. Regularmente se aplican a la decodificación de datos capturados desde una *forma* en HTML desde Mosaic. Ahora, explicaremos el procedimiento que siguen este tipo de programas y la aplicación que encontramos como interfaz a una Base de Datos en Sybase.

Decodificación de formas con CGI's

Existen dos métodos que pueden ser utilizados para acceder a los datos de una *forma*, éstos son GET y POST. Dependiendo del método que se esté utilizando, se recibirán los resultados codificados de la *forma* de diferente manera.

- método GET

  Si la forma está utilizando el método METHOD="GET" con su respectiva etiqueta FORM, el programa CGI recibirá los datos codificados de la *forma* en la variable de ambiente QUERY_STRING.

- método POST

  Si la forma está utilizando el método METHOD="POST" con su respectiva etiqueta FORM, el programa CGI recibirá los datos codificados de la *forma* vía *stdin (standard input* o entrada estándar). El servidor **no** enviará un EOF (End Of File o fin de archivo) al final de los datos, en sustitución a

eso se deberá utilizar la variable de ambiente CONTENT_LENGTH para determinar que cantidad de datos que se deberán leer desde la entrada estándar.

¿Como decodificar los datos de las formas?

Cuando se escribe una forma, cada una de las entradas tiene una etiqueta asociada llamada NAME. Cuando el usuario coloca datos en esos campos, la información es codificada dentro de la forma. El valor de cada uno de esos elementos de entrada está dado por el usuario. Los datos de las formas es una serie de pares del tipo: *nombre=valor* separados por el caracter &. Cada par *nombre=valor* es un URL codificado, por ejemplo los espacios son cambiados por signos mas (+) y algunos caracteres son codificados en hexadecimal.

En vista que la mayoría de los usuarios se han presentado con el problema de decodificar los datos de una *forma*, ya existen algunos programas en la red que realizan esta decodificación. Los más comunes son aquellos que viene acompañados con el software de instalación del servidor http. El URL con los ejemplos, es el siguiente:

ftp://ftp.ncsa.uiuc.edu/Web/httpd/Unix/ncsa_httpd/cgi/ncsa-default.tar.Z

El archivo *ncsa-default.tar.Z* se puede transferir con ftp o desde Mosaic.

Variables de ambiente de un CGI

Para poder pasar datos al script (CGI), el servidor *http* utiliza variables de ambiente. Estas variables son colocadas cuando lo ejecuta, las siguientes variables de ambiente son definidas para todas las peticiones:

- SERVER_SOFTWARE

  El nombre y versión del software del servidor contestando la petición (y ejecutando el CGI). El formato es: nombre/versión.

- SERVER_NAME

  El nombre del host donde reside el servidor, su dirección IP o como pudiera aparecer en la misma referencia al URL.

- GATEWAY_INTERFACE

  Es la versión de la especificación del CGI que el servidor está ejecutando. Su formato es CGI/revisión.

Las siguientes variables de ambiente son específicas para una petición, siendo asignadas por el mismo CGI:

- SERVER_PROTOCOL

  Es el nombre y versión del protocolo del servidor http que acompaña al CGI. El formato es: protocolo/versión.

- SERVER_PORT

  El número de puerto donde la petición fue enviada.

- REQUEST_METHOD

  El método con el cual la petición fue realizada. Para el caso de HTTP los más comunes son del tipo "GET", "POST".

- PATH_INFO

  El path o trayectoria extra dada por el cliente. En otras palabras, los scripts pueden ser accedidos por su nombre de trayectoria virtual, seguidos por un complemento de esta trayectoria. La información extra es mandada como PATH_INFO, la cual deberá ser decodificada por el servidor si llega de un URL, antes que sea pasado al CGI.

- SCRIPT_NAME

  Es una trayectoria virtual al script que está siendo ejecutado, se utiliza para la misma referencia de los URL´s.

- QUERY_STRING

  Contiene la información que sigue después del signo "?" en el URL asociado al script. No deberá ser decodificada bajo ninguna circunstancia, y deberá de ser incluida cuando en la petición exista consulta de información.

- REMOTE_HOST

  Es el nombre del host remoto que está realizando la petición. Si el servidor no cuenta con esta información, entonces ignorará esta variable de ambiente y tomará en cuenta a la variable REMOTE_ADDR.

- REMOTE_ADDR

  Es la dirección IP del host remoto que está realizando la petición.

- AUTH_TYPE

  Si el servidor puede identificar a un usuario, y el script está protegido para su uso, esta variable de ambiente tiene el método especificado por el protocolo para identificarlo.

- REMOTE_USER

  Si el servidor puede identificar a un usuario, y el script está protegido, el **nombre** de usuario remoto se encuentra en esta variable de ambiente.

- CONTENT_TYPE

  Para aquellas peticiones que utilizan en las *formas* el método HTTP POST y PUT, ésta es la variable de ambiente donde se encuentra el tipo de contenido de los datos.

- CONTENT_LENGTH

  Esta variable de ambiente especifica la longitud del contenido de la variable que fue dada por el cliente.

Ejemplos de estas variables de ambiente se pueden ver en el siguiente URL visualizado desde Mosaic:

http://hoohoo.ncsa.uiuc.edu/cgi/examples.html

Por ejemplo, para una *forma* manejando el método POST, las variables de ambiente que manejaría serían las siguientes:

CGI/1.1 test script report:

argc is 0. argv is .

SERVER_SOFTWARE = NCSA/1.5b2
SERVER_NAME = hoohoo.ncsa.uiuc.edu
GATEWAY_INTERFACE = CGI/1.1
SERVER_PROTOCOL = HTTP/1.0
SERVER_PORT = 80
REQUEST_METHOD = POST
HTTP_ACCEPT = text/plain, application/x-html, application/html,
text/x-html, text/html, audio/basic, audio/x-aiff, image/gif, image/jpeg,
image/tiff, image/x-portable-anymap, image/x-portable-bitmap,
image/x-portable-graymap, image/x-portable-pixmap, image/x-rgb,
image/rgb, image/x-xbitmap, image/x-xpixmap, image/xwd, image/x-xwd,
image/x-xwindowdump, video/mpeg, application/postscript,
application/x-dvi, message/rfc822, application/x-latex,
application/x-tex, application/x-texinfo, application/x-troff,
application/x-troff-man, application/x-troff-me, application/x-troff-ms,
text/richtext, text/tab-separated-values, text/x-setext, */*
PATH_INFO =
SCRIPT_NAME = /cgi-bin/test-cgi
QUERY_STRING =
REMOTE_HOST = tzetzal.dcaa.unam.mx
REMOTE_ADDR = 132.248.27.10
REMOTE_USER =
AUTH_TYPE =
CONTENT_TYPE = application/x-www-form-urlencoded
CONTENT_LENGTH = 0

Note que el nombre del servidor donde reside el CGI es hoohoo.ncsa.uiuc.edu y el cliente donde corre Mosaic es tzetzal.dcaa.unam.mx.

Para aquellas peticiones usando métodos tales como HTTP POST o PUT, la información será enviada al script vía stdin (standard input o entrada estándar).

El servidor enviará, los bytes que tenga la variable de ambiente CONTENT_LENGTH a este descriptor de archivo (stdin). Recuerde que también enviará la variable de ambiente CONTENT_TYPE, por lo que el servidor no está obligado a mandar un EOF (fin de archivo) después de que el script lea la cantidad de bytes especificada en la variable de ambiente CONTENT_LENGTH.

Por ejemplo, si tomamos una *forma* con un método: METHOD="POST", y digamos que los resultados de la *forma* son 7 bytes codificados, entonces estos se verán de la siguiente manera: *a=b&c=d*. Donde, *a* es el nombre del campo asociado a la etiqueta de entrada NAME (en la forma) y *b* el dato capturado.

En este caso, el servidor asignará a la variable de ambiente CONTENT_LENGTH el valor de 7 y a CONTENT_TYPE el valor de *"application/x-www-form-urlencoded"*. El primer byte en la entrada estándar del script será "a", seguido por el resto de la cadena codificada.

El script manda su salida hacia la *stdout* (standard output o salida estándar), la cual puede ser generada por el mismo script o mediante instrucciones al servidor para la recuperación de la salida deseada.

La salida de los scripts comienza con un pequeño encabezado. Éste esta formado por líneas de texto, en el mismo formato que un encabezado HTTP, terminado por una línea en blanco.

Actualmente, la especificación de los CGI's define tres directivas hacia el servidor http:

- Content-type

  Este es el tipo de documento que se está devolviendo.

- Location

  Se utiliza para especificar al servidor que se está regresando una referencia hacia otro documento en lugar del actual. Si el argumento a este documento es un URL, el servidor lo redireccionará al cliente.

  Si el argumento a este documento se encuentra en una trayectoria virtual, el servidor recuperará el documento especificado tal y como si el cliente hubiera realizado esa petición a ese documento originalmente.

- Status

  Este se utiliza para darle al servidor una línea de status del tipo HTTP/1.0 que pueda ser mandada al cliente. El formato es **nnn xxxxx**, donde **nnn** es el código de status de 3 dígitos, y **xxxxx** es la cadena de acompañamiento; por ejemplo: "Forbidden".

Ejemplos:

Supongamos que tengo una *forma* HTML hacia un convertidor. Cuando el convertidor termine su trabajo, generará la siguiente salida en la *stdout* (notará a continuación que las líneas comenzando y terminando con "---" son sólo ilustrativas y no formarán parte de la salida):

```
--- comienzo de la salida ---
Content-type: text/html
--- fin de la salida ---
```

Nótese el espacio en blanco después de Content-type.

Ahora, supongamos que se tiene un script el cual se quiere que devuelva el documento */path/doc.txt* de este servidor justo como si el usuario hubiera solicitado un *http://servidor:puerto/path/doc.txt*. En este caso, el script generaría una salida de este tipo:

--- comienzo de la salida ---
Location: /path/doc.txt
--- fin de la salida ---

El servidor entonces responderá a la petición y la enviará al cliente.

Ahora, imagine que se tiene un script el cual se quiere que haga referencia a un servidor gopher. En este caso, si se quiere que el script haga referencia al usuario como: *gopher://gopher.ncsa.uiuc.edu/*, generaría la siguiente salida:

--- comienzo de la salida ---
Location: gopher://gopher.ncsa.uiuc.edu/
--- fin de la salida ---

Finalmente, si se tiene un script el cual se quiere que se comunique directamente con el cliente (Mosaic). En este caso, si el script es referenciado con la variable SERVER_PROTOCOL de HTTP/1.0, generará la siguiente salida:

--- comienzo de la salida ---
HTTP/1.0 200 OK
Server: NCSA/1.0a6
Content-type: text/plain

Este es un documento de texto plano generado en el momento especialmente para usted.
--- fin de la salida ---

Diseño de la interfaz entre Mosaic y Clientes en Sybase

En el apartado anterior explicamos cómo trabaja un CGI, ahora resta explicar la parte de la programación de los mismos.

El código del servidor *http* viene acompañado de un directorio llamado *cgi-bin*, que es donde deben residir nuestros CGI's como programas ejecutables. A su vez, también viene un directorio llamado *cgi-src*, el cual contiene el código fuente de un programa que decodifica datos de una forma (post-query.c).

Las formas que utilizaremos utilizarán el método POST (ver sección 6.4.2), y de acuerdo a la especificación, debemos hacer uso de 3 variables de ambiente:

REQUEST_METHOD
CONTENT_TYPE
CONTENT_LENGTH

Cada una tiene un significado para nuestro script y su contenido se mencionó anteriormente. Con el método POST, nuestro CGI recibirá los datos de la entrada estándar (stdin) y debemos usar la variable de ambiente CONTENT_LENGTH para saber cuántos datos vamos a leer.

El primer paso de nuestro programa CGI, será la validación de las 3 variables de ambiente involucradas, y después hay que implantar un procedimiento para leer los datos de la entrada estándar y capturarlos en variables locales, esto nos permitirá manipularlos como si fueran leídos directamente desde el teclado. Recuerde que un programa en C, regularmente recibe los datos del usuario desde el teclado actuando en forma interactiva con él. Sin embargo, en el caso de Mosaic, no hay forma de hacer eso directamente, es por esa razón que se hacen uso de variables de ambiente. Si nuestro programa logra capturar los datos de una *forma* en variables locales,

podemos por ejemplo, formar con ellos mensajes en HTML y enviarlos de regreso a Mosaic para visualizarlos.

La mejor forma de entender cómo trabaja un CGI es con un ejemplo. El siguiente programa decodifica los datos de una forma y los envía de vuelta a Mosaic en forma de hipertexto.

```
#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);
   printf("<title>Resultado de la Consulta</title>\n");

   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 script solo puede usarse para decodificar datos de una forma. \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>Resultado de la consulta</H1>");
  printf("Tu enviaste los siguientes datos de pares de datos nombre/valor:<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);
}
```

Las funciones definidas en el programa anterior (por ejemplo *unescape_url*), se encuentran en el archivo **util.c** que se muestra en el Apéndice B. La compilación se lleva a cabo de la siguiente manera (suponga que el CGI se llama post.c):[6]

```
% gcc -c util.c
% gcc post.c util.o -o post
```

Luego, hay que transferir el archivo ejecutable *post* al directorio cgi-bin que se encuentra bajo el directorio donde reside el servidor http.

Ahora, vamos a explicar a grandes rasgos el programa anterior. Primero, como nuestro script va a devolver resultados a Mosaic en forma de hipertexto, debemos hacer uso de la directiva hacia el servidor "Content-Type: text/html" (discutida anteriormente), la cual nos define el tipo de información que devolvemos. En este caso, se trata de un documento html.

```
printf("Content-type: text/html%c%c",10,10);
```

[6] Puede notar que primero se genera el código objeto (util.o) del archivo util.c, posteriormente se liga ese código con el del programa fuente llamado post.c.

Note que el mensaje se envía como si fuera a pantalla, aunque en realidad lo recibirá Mosaic. A partir de esta instrucción, podemos hacer uso de las directivas válidas para el lenguaje HTML, todos los mensajes "a pantalla" irán directamente a Mosaic en forma de hipertexto.

Luego, de acuerdo a la especificación, debemos validar las variables de ambiente REQUEST_METHOD y CONTENT_TYPE, en este caso, su contenido deberá ser "POST" y "application/x-www-form-urlencoded" respectivamente, puesto que nuestra forma está referenciada con el método POST tal como se había mencionado.

```
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 script solo puede usarse para decodificar datos de una forma. \n");
   exit(1);
}
```

Puede observarse que si el contenido de cualquiera de las 2 variables de ambiente es diferente al esperado, enviamos un mensaje de error y terminamos el programa.

Por último, tenemos el procedimiento que captura los datos de la entrada estándar a variables locales. Recuerde que la cantidad de datos a leer, está dado por la variable CONTENT_LENGTH, por lo que antes debemos convertirla a un valor entero para saber cuándo terminará nuestro procedimiento.

```
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,'=');
    }
```

En este caso, la cadena de datos a leer de la entrada estándar es de la forma *a=b&c=d*, donde *a* y *c* son los valores asociados a la etiqueta NAME del campo de entrada en la forma, *c* y *d* son los datos tecleados en la *forma* por el usuario, el signo "&" es el separador de los pares de datos. Así, buscamos el signo & porque sabemos que los caracteres a su izquierda, son los datos que el usuario proporcionó. Por el contrario, si buscamos el caracter "=" sabemos que los caracteres a su izquierda forman el nombre del campo de la *forma* asociado a esa entrada (NAME).

Ejemplo: Suponga que se tiene la siguiente *forma* en hipertexto:

```
<title>Ejemplo de una forma</title>

<h4>Los datos capturados ser&aacute;n decodificados por el CGI post</h4><p> <hr>
<form method="POST" action="http://tzetzal.dcaa.unam.mx/cgi-bin/post">
<input type="hidden" name="var" value="escondido">
Nombre del alumno:<input size=25 maxlength=25 name="nombre"><p>

Comentarios:
<textarea name="comentario" cols=30 rows=3>
</textarea><p>

Borrar: <input type="checkbox" name="borrado" value="verdadero"><p>

Escolaridad:
<SELECT NAME="columna">
<OPTION>Preparatoria
<OPTION>Posgrado
<OPTION>Profesional
<OPTION>Titulado
</SELECT><p>
```

```
<input type="submit" value="Aceptar"> <p>
</form>
<hr>
```

Desde Mosaic, la *forma* con datos capturados por el usuario se vería así:

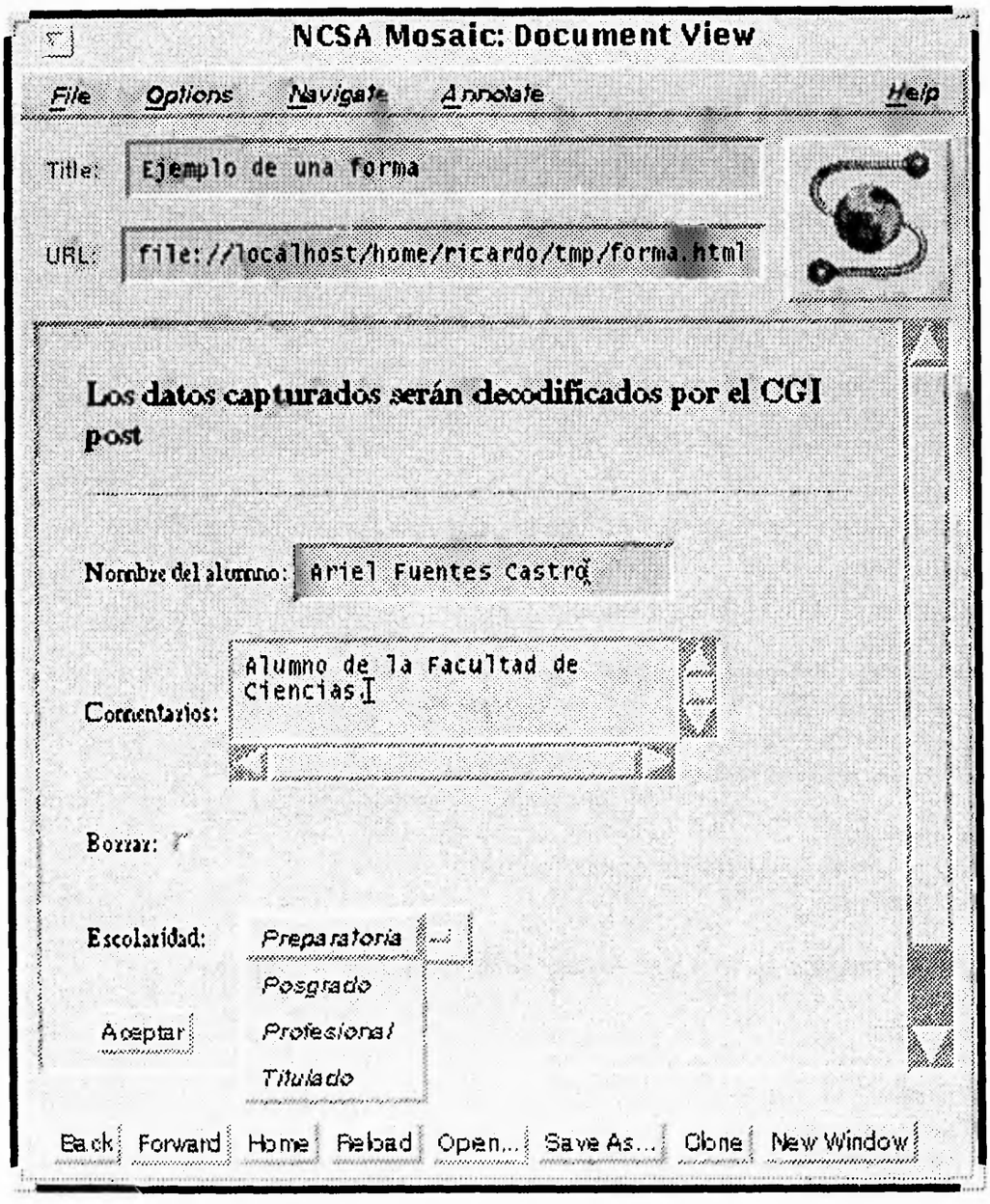


Figura 6.16 Ejemplo de una forma con CGI

Note que el URL que decodicará los datos de la *forma*, se llama:

http://tzetzal.dcaa.unam.mx/cgi-bin/post

Al teclear "Aceptar", el servidor http ejecutará el programa *post*, el cual genera la salida que se muestra en la siguiente figura.

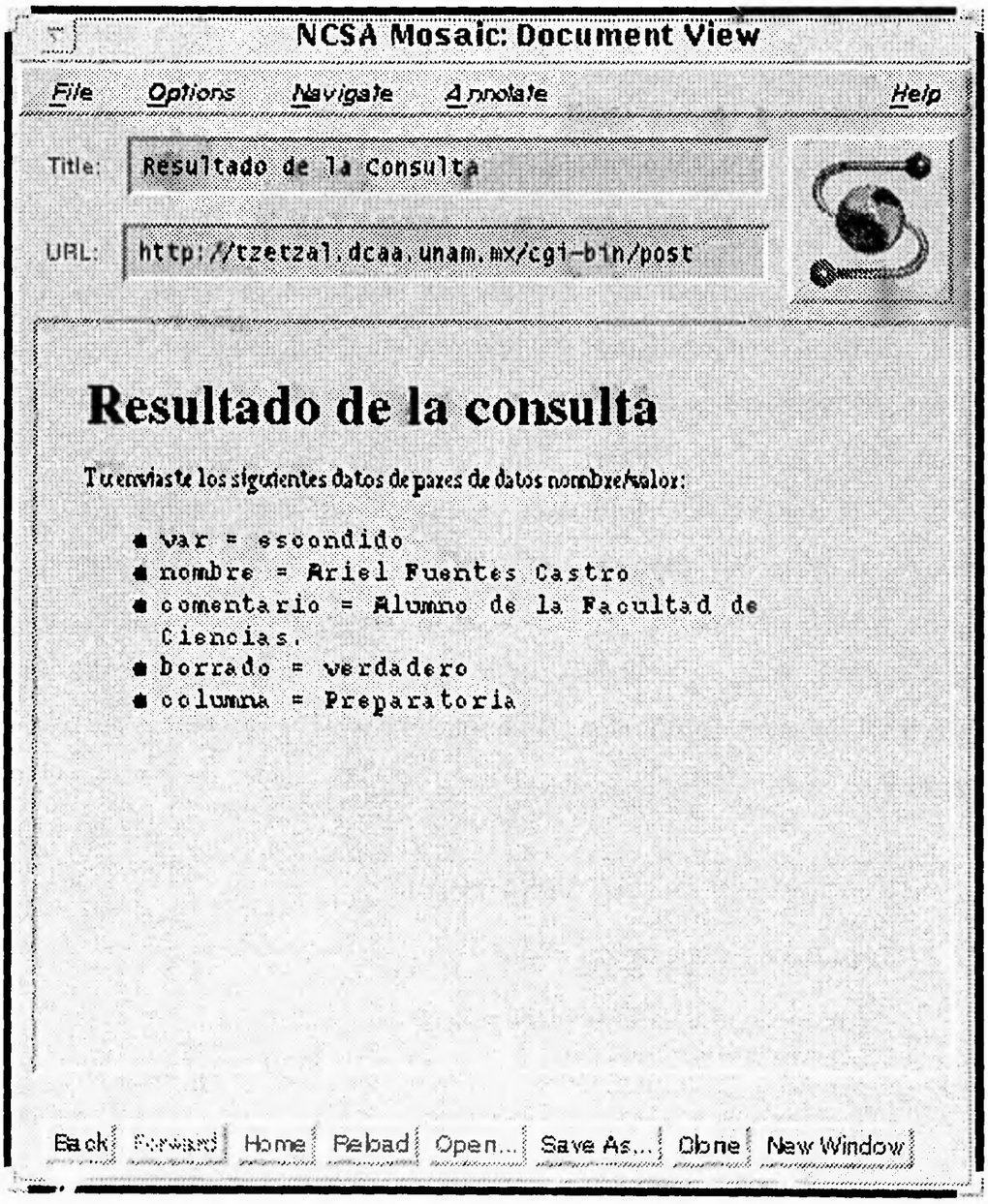


Figura 6.17 Resultado del CGI post

Observe los pares:

var=escondido
nombre=Ariel Fuentes Castro
comentario=Alumno de la Facultad de Ciencias.
borrado=verdadero
columna=preparatoria

La cadena original que nuestro CGI recibirá de la entrada estándar, es la siguiente:

var=escondido&nombre=Ariel+Fuentes+Castro&comentario=Alumno+de+la+Facultad+de%0ACiencias.%0A&borrado=verdadero&columna=Preparatoria

Recuerde que el par *var=escondido* es un campo de entrada tipo "hidden" que el usuario no verá en Mosaic (aunque está presente). Note que los espacios en blanco se reciben como un signo "+", el separador de campo es el caracter "&" y el retorno de carro se recibe como "%0A", la variable de ambiente CONTENT_LENGTH tendrá un valor igual a 131 que es la longitud de la cadena. Entonces, nuestro CGI debe separar cada campo e identificar el nombre de la etiqueta NAME del campo de entrada en la *forma* original, así como su valor asociado el cual fue proporcionado por el usuario desde el teclado en Mosaic (ver Figura x.x). Hacemos énfasis al siguiente procedimiento que forma parte de nuestro programa:

```
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,'=');
  }
```

Observe que hay un ciclo *for* el cual terminará hasta que el número de datos leídos de la entrada estándar sea igual al contenido de la variable *cl* (la cual contiene el valor de CONTENT_LENGTH como un número entero e igual a 131).

El procedimiento en el primer ciclo, busca al caracter "&" (para obtener el primer par de datos "var=escondido") con la siguiente instrucción:

```
entries[1].val = fmakeword(stdin,'&',&cl);
```

Aquí, entries[1].val tiene "var=escondido"

Luego, reemplaza los caracteres "+" por el correspondiente espacio en blanco.

```
unescape_url(entries[1].val);
```

entries[1].val tiene "var=escondido"

El siguiente paso, es eliminar los caracteres de escape.

```
unescape_url(entries[1].val);
```

En este primer par de valores, no hay caracteres de escape, por lo tanto:
el contenido de entries[1].val sigue siendo "var=escondido"

Por último, separamos el nombre de la etiqueta de entrada en la *forma* y el dato tecleado por el usuario utilizando como delimitador el signo "=".

```
entries[1].name = makeword(entries[1].val,'=');
```

Ahora: entries[1].val tiene "escondido"
y entries[1].name tiene "var"

Los siguientes ciclos tendrán un comportamiento similar:

Segundo ciclo:

Contenido de entries[2].val	Contenido de entries[2].name
nombre=Ariel+Fuentes+Castro	--------
nombre=Ariel Fuentes Castro	--------
nombre=Ariel Fuentes Castro	--------
Ariel Fuentes Castro	nombre

Tercer ciclo:

Contenido de entries[3].val	Contenido de entries[3].name
comentario=Alumno+de+la+Facultad+de%0ACiencias.%0A	--------
comentario=Alumno de la Facultad de%0ACiencias.%0A	--------
comentario=Alumno de la Facultad de Ciencias.	--------
Alumno de la Facultad de Ciencias.	comentario

Los últimos dos ciclos, serán parecidos al primero. Al terminar el ciclo *for*, el CGI tendrá los datos codificados en el arreglo *entries*, en el campo *val* se almacena el dato tecleado por el usuario y el campo *name* tiene el nombre del campo de entrada. Por lo tanto, podemos manipular las variables como cualquier otra, puesto que son locales, una de las aplicaciones está en los clientes en Sybase.

Recuerde cuando hablamos acerca de la programación de Clientes en Sybase. Si insertamos el código de alguno de ellos en nuestro CGI, tendremos como resultado, la interfaz entre Mosaic y la Base de datos. En el Apéndice C puede encontrar el programa CGI (syb2.c) que decodifica los datos de la Solicitud de Inscripción a Cursos que el usuario proporcionó desde el Kiosco de Información.

Aunque los CGI's no necesariamente trabajan decodificando *formas*. Por ejemplo, se puede realizar un programa que simplemente genere un documento HTML, en lugar de enviarlo a la pantalla, podemos redireccionar su salida a Mosaic. El siguiente ejemplo hace una consulta a una Base de Datos y los resultados recibidos del servidor SQL, los desplegará en forma de hipertexto.

En el Apéndice D mostramos el código de un programa CGI (curso5.c) que hace una consulta a una Base de Datos en Sybase y genera el documento en HTML que se muestra en el Apéndice E (salida del programa). En este caso, la salida del script se direcciona a Mosaic, y el resultado es el que se muestra en la siguiente figura:

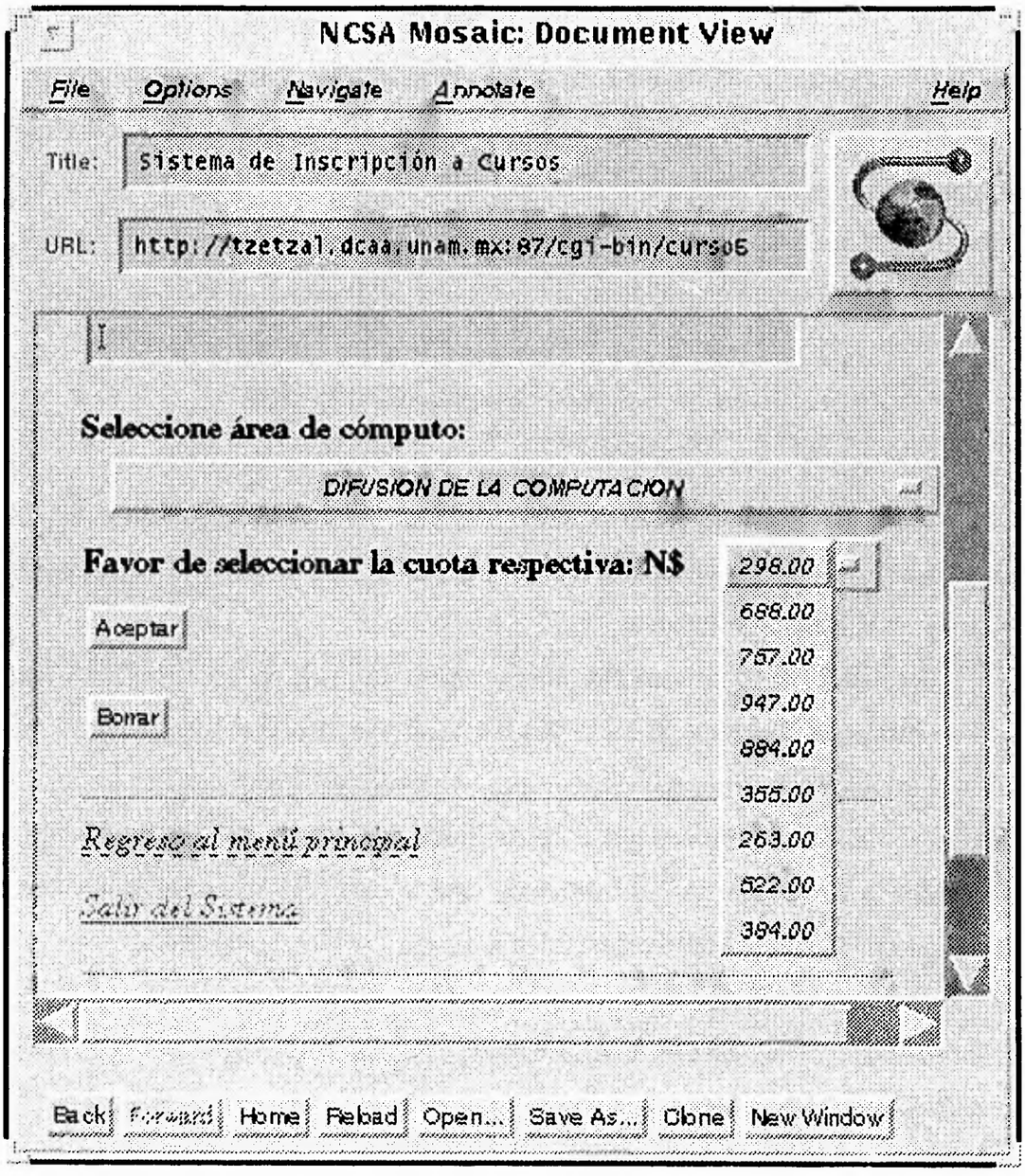


Figura 6.18 Ejemplo de una forma generada por un CGI actuando como cliente en Sybase

Los menús desplegables se forman con información que se encuentra en tablas de nuestra Base de Datos. Los demás campos, también son generados por nuestro CGI.

Regresemos al caso del Kiosco, la información que el usuario proporciona puede almacenarse en la Base de datos (ese es el objetivo). Una vez que el usuario está dado de alta en la misma, puede proceder a consultar la serie de cursos programados en ese semestre y seleccionar alguno para inscribirse. Toda la información recibida, la generarán programas CGI's a partir de las tablas en la Base de Datos, que actuarán como interfaces entre el visualizador y la Base de Datos.

6.4.5 Implantación del Sistema de Inscripciones

En la sección 6.4.3 explicamos las interfaces entre Mosaic y la Base de Datos en Sybase con algunos ejemplos. En base a ello, el siguiente paso es implantar el sistema en el Kiosco de Información.

Recuerde que el sistema está dirigido hacia dos tipos de usuarios, los alumnos que se inscriben directamente en el Kiosco y los usuarios administradores que son los que se responsabilizan de la administración general del sistema. Por lo tanto, los URL's y CGI's que formen parte de la inscripción de los alumnos podrán ser accedidos por cualquier usuario. Mientras que la parte del administrador, debe ser diseñada con mucho cuidado contemplando un cierto nivel de seguridad.

Comencemos primero, con la parte del sistema correspondiente al Kiosco de Información. En la sección 6.3.1 se habló acerca del proceso a seguir en la inscripción de un alumno y lo que presentamos ahora es lo que el usuario físicamente debe ver.

El sistema de inscripciones desde el Kiosco, mostrará al usuario una primer pantalla pidiéndole su RFC (Registro Federal de Contribuyentes y aunque no lo tenga, lo puede formar) para averiguar si existe o no en la Base de Datos. Note que estamos usando *formas* en HTML, cuando el usuario teclee su clave y presione la tecla de "Aceptar", Mosaic hará la petición al servidor http del URL al que se hace

referencia, el cual es un CGI binario que se encargará de decodificar los datos y averiguar si la clave que recibió, existe o no en la Base de Datos.

Si la clave no existe, lo notificará al usuario y le mostrará una liga hacia la Solicitud de Inscripción si es que desea llenarla. De lo contrario, continuará con el proceso; algunos de los programas fuentes correspondientes a los clientes en Sybase que generan los documentos hipertexto en el Kiosco, se encuentran en el Apéndice C. Las figuras 6.19, 6.20, 6.21, 6.22 y 6.23 muestran un proceso de inscripción a través del Kiosco de Información.

El lector se preguntará ahora cómo es que no perdemos los datos del usuario después de haber pasado por 5 CGI's. La respuesta es simple, cuando el usuario teclea su RFC y continúa con el proceso, cada pantalla tiene un campo de entrada tipo "hidden", aunque el usuario no lo vé, su RFC se pasa como parámetro hasta que llega a la última pantalla. Puede consultar el código fuente de los programas en el Apéndice C. Puede notar que cada CGI accede a la Base de Datos del sistema desde un programa cliente, y utilizan sentencias SQL para almacenar los datos o para consultarlos en una tabla. Por ejemplo, para saber si el alumno está dado de alta en la Base de datos, se ejecuta la siguiente sentencia SQL:

```
select * from alumnos where clave=rfc_del_alumno
```

Luego, verificamos en la estructura de datos DBPROCESS con ayuda de la función **dbrows()** si el servidor devolvió resultados. De ser así, el alumno está dado de alta en la Base de Datos y continuaremos con el proceso, de lo contrario, su clave no existe y será necesario que llene la solicitud de inscripción. Note también que el alumno llena solo una vez esta solicitud y la siguiente vez que se quiera inscribir únicamente teclea su RFC para entrar al sistema (puesto que ya está dado de alta).

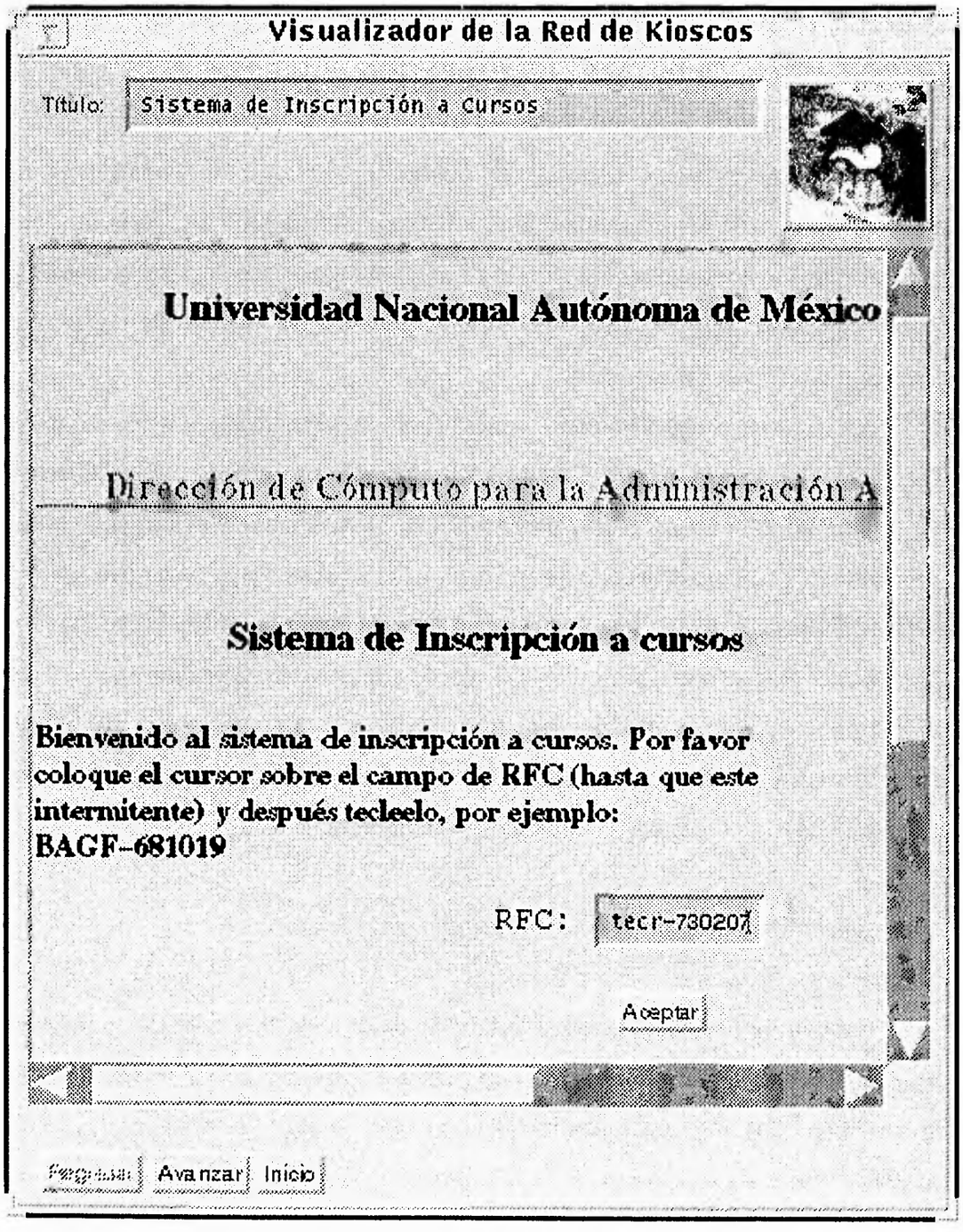


Figura 6.19 Se accede al sistema de inscripciones desde una liga del "home page" del Kiosco, donde al tocarla nos muestra esta pantalla. Después de teclear el RFC, se presiona el botón de "Aceptar".

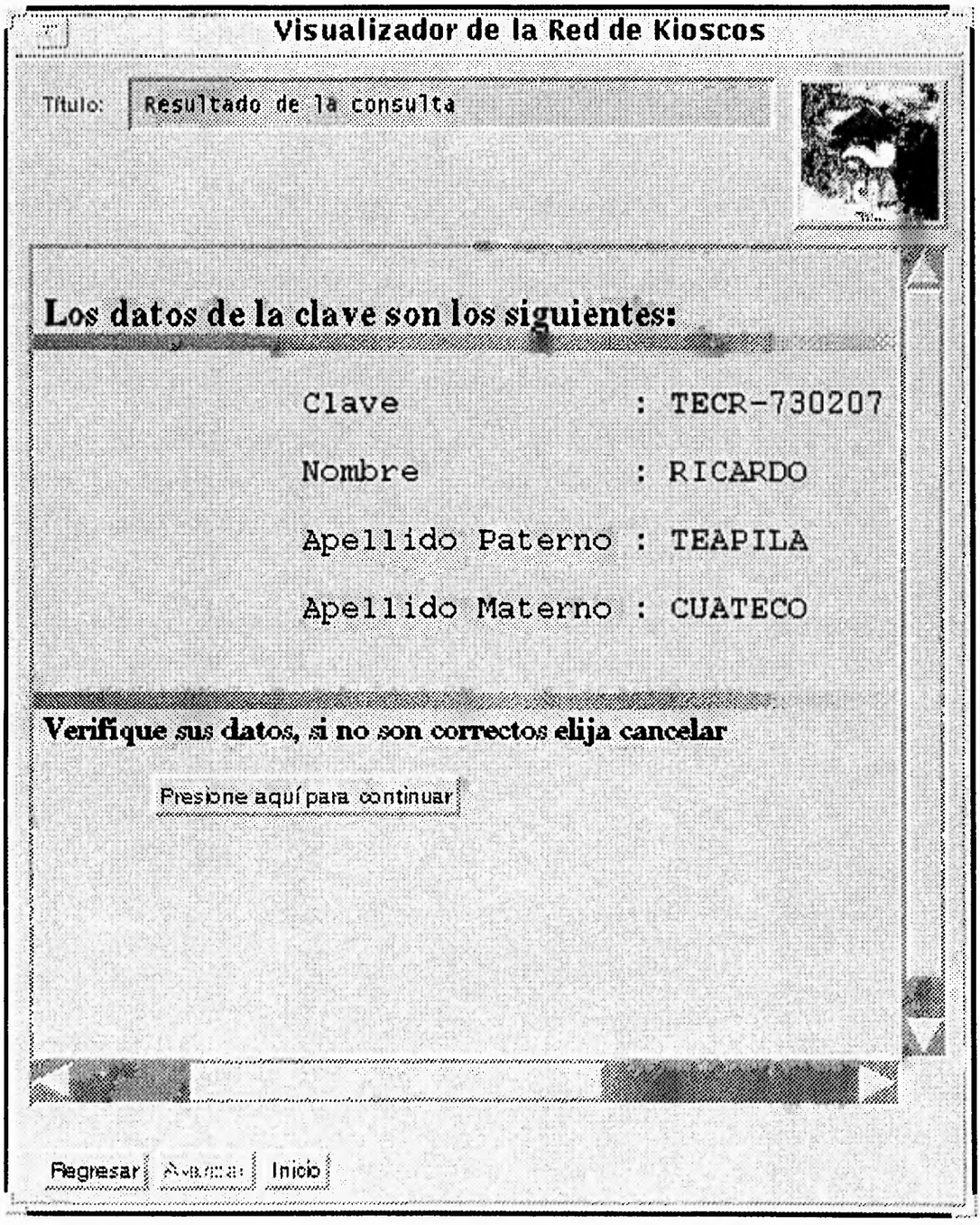


Figura 6.20 El programa syb2.c genera esta pantalla, la información que puede verse (tal como el nombre del alumno), proviene de la Base de datos, en este caso, únicamente se muestra para que el usuario verifique sus datos. Si el usuario continúa con el proceso, la siguiente acción la toma el programa CGI llamado usuario2.

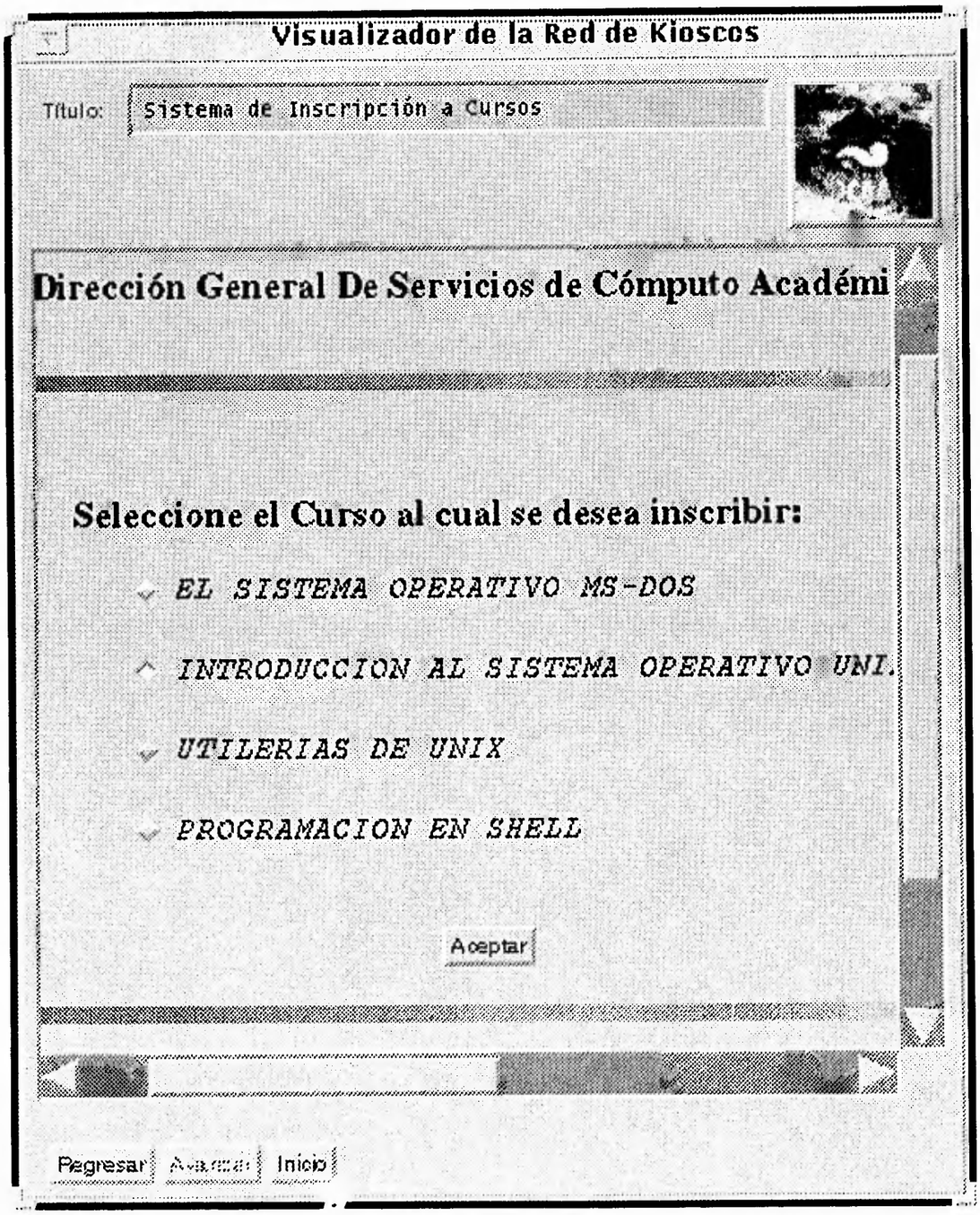


Figura 6.21 Anterior a esta pantalla, existe otra donde se elige el área de cómputo del curso a inscribirse. No la presentamos porque es muy similar a ésta, la diferencia es que el menú corresponde a seleccionar el área de Cómputo. El CGI usuario2 llama a usuario3 y a su vez, usuario3 genera la pantalla que se muestra aquí. Si el usuario presiona "Aceptar", la acción la tomará el CGI usuario4.

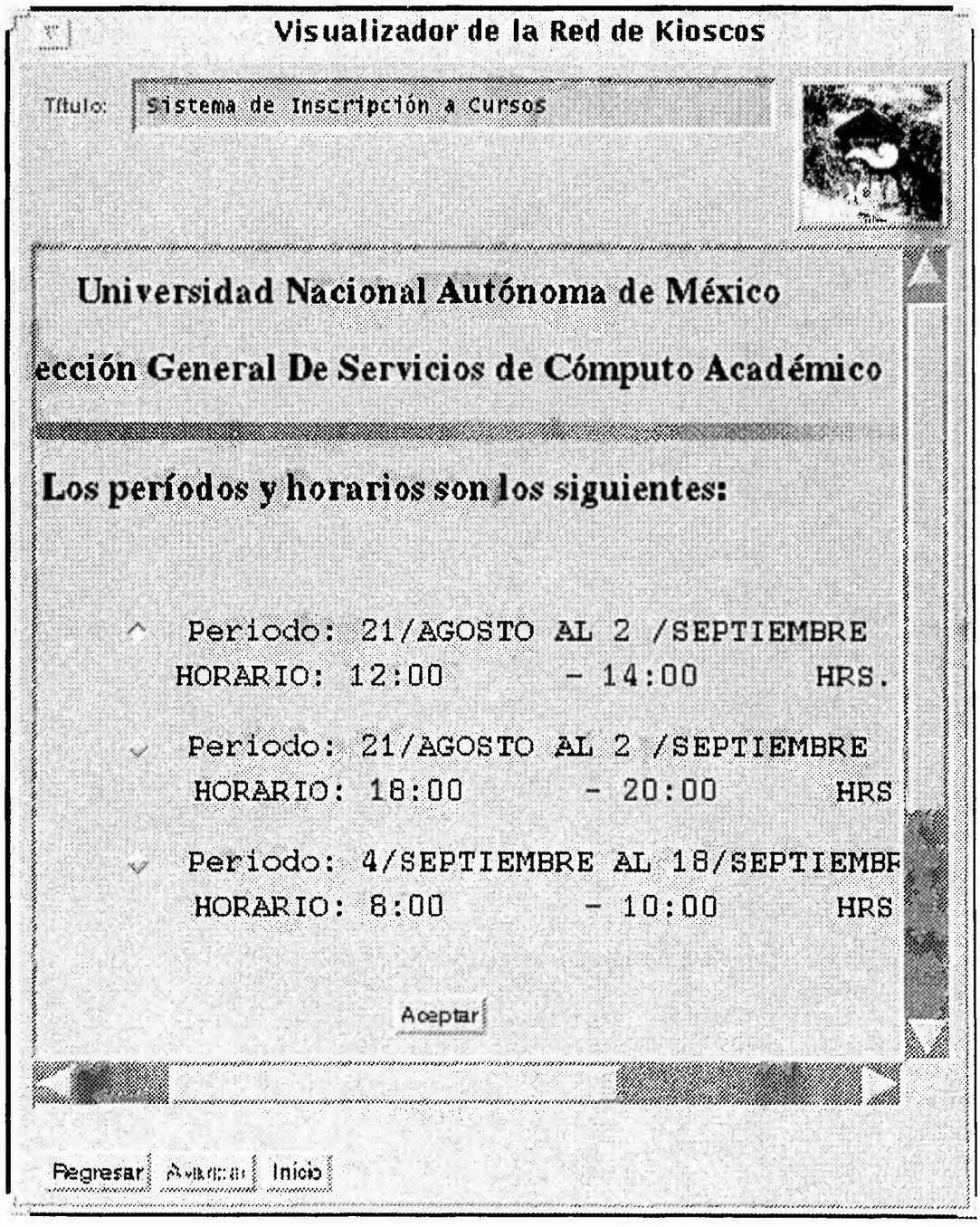


Figura 6.22 El CGI usuario4 genera esta pantalla, muestra al alumno los horarios y períodos programados en el calendario los cuales están en función de la fecha de la consulta. Por ejemplo, si la fecha es 3 de septiembre, el período 21 de Agosto/2 de Septiembre no aparecerá aunque esté programado el curso. Con esto evitamos que el alumno pueda inscribirse a cursos inválidos.

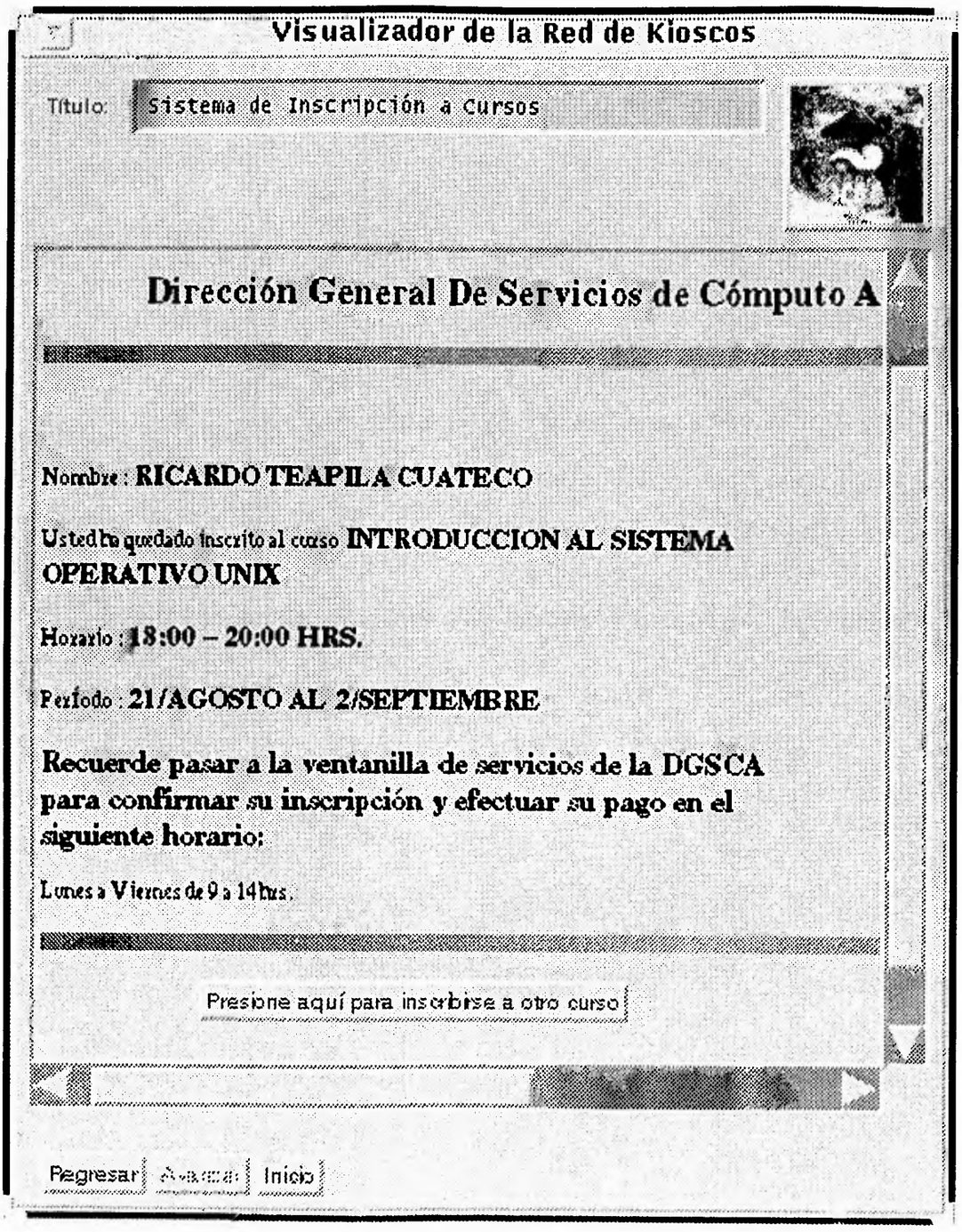


Figura 6.23 La inscripción del alumno se llevó a cabo sin problema alguno. Hacemos hincapié en la frase que le recuerda pasar a pagarlo. El programa que genera esta pantalla, es usuario5.

En la última pantalla, también debe aparecer una liga para que el usuario pueda imprimir (en el Kiosco) su comprobante de inscripción. El programa además, genera un archivo tipo texto con los datos que el usuario observa en la pantalla, y con ayuda de un script hecho en Shell de Unix, mandamos a imprimirlo en el mismo Kiosco. Aquí hacemos una observación, para el sistema de inscripciones el Kiosco requiere de una impresora local, que se encargará de generar el comprobante de inscripción para cada alumno, el cual deberá presentarse al momento de efectuar el pago correspondiente. Puede usarse casi cualquier impresora, los comprobantes se imprimen por separado y localmente se llenan con la información del alumno.

Como segunda parte, tenemos el sistema del administrador. En este caso, las consultas y modificaciones a la Base de Datos se harán vía NetScape en forma similar al proceso anterior, usando las interfaces descritas en la sección 6.4.3. Sin embargo, debemos contemplar el nivel de seguridad de nuestro sistema, porque recuerde que si no tenemos configurado adecuadamente nuestro servidor http, podemos tener graves problemas de seguridad. Y nuestro caso no es la excepción, además de proteger directorios y archivos, sería conveniente restringir o permitir el acceso a una máquina mediante su dirección IP. Además, vamos a ver como podemos asignar una clave y un password a un documento HTML antes de entrar al sistema.

Para el administrador del sistema, consideramos conveniente instalar un segundo servidor *http* por cuestiones de seguridad. De esta forma, podemos proteger directorios y CGI's correspondientes a la administración de esta parte del sistema.

La instalación de este segundo servidor, se hace exactamente igual al primero (descrito en la sección 5.3.2). Recuerde que se instaló en una estación de trabajo SparcStation 10 bajo el directorio /bases/http, para el segundo se eligió el directorio /bases/http_admin también corriendo como standalone, sólo que en el puerto 87.

Mosaic y el servidor http permiten restringir el acceso a determinados clientes, bajo los siguientes criterios:

- A través de una clave y password.
- Autorización de una conexión en base a la dirección IP del cliente.
- Una combinación de los dos métodos anteriores.

Antes de comenzar con la autorización del acceso a los clientes, es necesario tener instalado NCSA httpd 1.0 o una versión mayor, y Mosaic 2.0 o mayor.

El control de acceso a un directorio, está controlado por un archivo llamado .htaccess que reside en ese directorio. El servidor lee el archivo en cada acceso a un documento (o documentos) en el directorio.

Bajo el directorio /home/http/admin residirán todos los documentos html que forman parte del administrador del sistema. Por ejemplo, menús y *formas*.

Supongamos que se desea permitir el acceso a los archivos que se encuentran bajo el directorio /home/http/admin a la clave[7] supervisor con password becario. Primero, se crea el archivo .htaccess en el directorio /home/http/admin con un contenido como el siguiente:

```
AuthUserFile        /bases/http_admin/.htpasswd
AuthGroupFile       /dev/null
AuthName            ByPassword
AuthType            Basic

<Limit GET>
requireuser    supervisor
</Limit>
```

[7] No se trata de una correspondencia entre claves y passwords con el Sistema Operativo Unix (por ejemplo, con el archivo /etc/passwd), las claves y passwords que aquí se definen, residen en otro archivo.

Note que el archivo de passwords se encuentra en otro directorio, en este caso, bajo /bases/http_admin. Si no hay un archivo de grupo para los usuarios, el contenido de AuthGroupFile es /dev/null. AuthName puede ser cualquier cosa y AuthType deberá ser siempre Basic.

La forma de asignar el password a la clave supervisor, es usando el programa htpasswd (se tiene que compilar) que acompaña al software de NCSA httpd y reside bajo el directorio support.

```
% htpasswd -c /bases/http_admin/.htpasswd supervisor
```

Inmediatamente pedirá el password, el cual se almacenará en el archivo .htpasswd de la siguiente forma:

```
supervisor:ylia3tjWkhCK2
```

Si se desea agregar más usuarios con acceso al mismo directorio, se hace utilizando el programa htpasswd sin la opción -c.

Note que el password está encriptado. Ahora, cada vez que se quiera acceder desde Mosaic a un documento bajo el directorio /home/http/admin en el servidor tzetzal.dcaa.unam.mx utilizando el puerto 87, ya sea desde una liga o directamente utilizando el menú de Mosaic, por ejemplo:

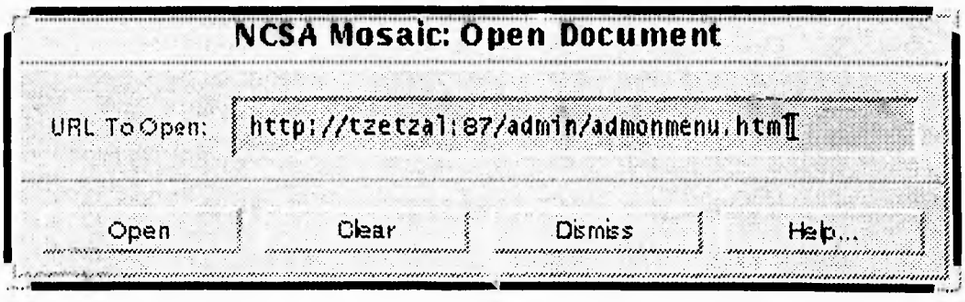


Figura 6.23 Abriendo el archivo admonmenu.html que se encuentra en la máquina tzetzal

aparecerá la siguiente ventana:

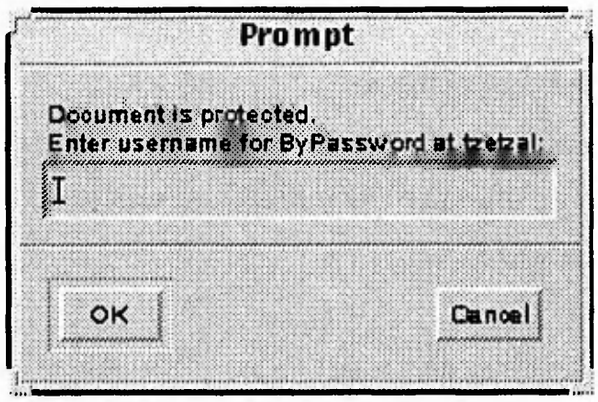


Figura 6.24 Ventana que Mosaic presenta cuando un directorio está protegido

Luego, se teclea la clave supervisor e inmediatamente pedirá el password, mostrando una ventana como la siguiente:

Figura 6.25 Ahora Mosaic pregunta por el password del usuario

Al teclear becario (obviamente no aparece en la ventana) y presionar la tecla "OK", Mosaic nos desplegará el documento html que se está protegiendo.

Por último, podemos permitir el acceso únicamente a una máquina en base a su dirección IP. Por ejemplo, además de tener clave y password de protección en un

cierto directorio, también podemos autorizar a una máquina para que pueda accederlo. En ese caso, el archivo .htaccess sería el siguiente:

```
AuthUserFile        /bases/http_admin/.htpasswd
AuthGroupFile       /dev/null
AuthName            ByPassword
AuthType            Basic

<Limit GET>
requireuser   supervisor
order         deny,allow
deny          from all
allow         from 132.248.63.6
</Limit>
```

Note que se agregaron 3 líneas más, donde autorizamos a la máquina con dirección IP 132.248.63.6 el acceso al directorio. Esta máquina es la del uso del administrador, de esta forma, estamos restringiendo aún más a nuestro sistema garantizando con ello un alto grado de seguridad.

¿Porqué utilizar Netscape?

Netscape en su versión 1.1 ofrece grandes facilidades para navegar a través de la Internet, permite realizar conexiones rápidas y seguras, además con su gran gama de instrucciones de HTML, hacen que el usuario tenga más posibilidades de crear documentos más atractivos.

Respecto a la instalación y configuración es fácil y rápida, y una vez realizada nos podemos conectar por ejemplo, a Estados Unidos y Canadá en pocos segundos. Siempre que necesitemos acceder a Internet, Netscape incluye todo lo que necesitamos para hacer la conexión. A pesar de interactuar con imágenes de alta resolución y gran cantidad de texto, es posible manejar descompresión de imágenes JPEG.

En realidad las grandes facilidades y la amplia variedad de herramientas que proporciona Netscape son útiles para la creación de documentos o simples consultas. Es por eso que en este proyecto se incluye la versión de Nestcape. Ahora, pensando en el caso de aplicación: Sistema de Inscripción a Cursos de la DCAA, se tuvieron que tomar en cuenta varios factores, como el siguiente:

¿Quienes serán los usuarios finales y desde dónde pueden hacer uso del sistema?. Ya se había comentado que se tienen dos tipos de usuarios finales: los administradores del sistema y los alumnos que se inscriben. Primero se pensó en el uso del kiosco donde cualquier persona pueda inscribirse, y segundo pensando que el usuario puede acceder al sistema desde su lugar de trabajo o su propia casa, y obviamente desde cualquier lugar de la Internet. Es entonces necesario, pensar en una versión especial del Kiosco para PC[8] (utilizando Netscape y lenguaje HTML), debido a que no se puede tener un equipo como las estaciones de trabajo instaladas por todas partes, pero sí la mayoría de las personas tienen acceso a una PC (aunque no todas tengan conexión a Internet). En nuestro caso, los usuarios administradores podrán hacer uso del sistema utilizando una PC sin Modo Kiosco y desde su oficina.

Primero analicemos los requerimientos de Netscape para una PC, se ve que son diferentes para otras arquitecturas:

- se requiere de una máquina con procesador 386SX o superiores
- disponer de 1 MB en disco duro para el archivo compactado de Netscape
- 4 MB mínimo u 8 MB de RAM recomendables
- un medio físico recomendable, como lo puede ser cualquier arquitectura de red o un módem de 14.4 Kbps
- 6 MB de disco duro para la descompresión del archivo de Netscape

[8] Netscape es el visualizador más popular que se ha difundido enormemente en las Computadoras Personales. Es eficiente, pero no tiene modo Kiosco. La gente lo utiliza en lugar de Mosaic.

- Windows 3.1, Windows para Trabajo en Grupo (Windows for Workgroups) o Windows NT.

Para instalar Netscape en nuestra PC es necesario obtener primeramente el archivo compactado, ya sea de 16 o 32 bits; si desea Nestcape de 32 bits deberá tener la extensión[9] para Windows (archivo win32.zip), de lo contrario con la versión de 16 será suficiente.

Mediante una conexión hacia el servidor FTP anónimo *ftp.netscape.com* en el directorio */pub/netscape/Windows* se puede transferir el archivo llamado *n16e11n.exe* o *n32e11n.exe* (dependiendo de la versión que se desee). En importante señalar que las versiones de Nestcape también cambian y la ubicación de las mismas pueden variar a las que aquí presentamos, pero usted, teniendo una versión del software puede acceder a las demás navegando por la Internet (para encontrar Netscape en Internet, se puede utilizar un servidor *Archie*).

Una vez que ya se tiene el archivo, el paso siguiente es descompactarlo invocándolo por su nombre, el proceso genera el archivo *setup.exe*. Este último archivo es el que se ejecuta para instalarlo en nuestra PC, *setup.exe* automáticamente carga el ambiente gráfico Windows y ejecuta la instalación creando un grupo de trabajo y en elemento de grupo (ícono) para Netscape. Una vez creado el elemento de grupo de Netscape se procede a acceder a él, y es en este momento cuando entramos al mundo de Internet. Todas las versiones de Mosaic y Netscape cargan un home page por default. En este caso está abierto el URL hacia *http://home.netscape.com/home/welcome.html*, pero al igual que las otras versiones es posible cambiar la configuración por default. Tal vez no sea posible cambiar toda

[9] La extensión de 32 bits para windows se puede obtener del mismo nodo donde se obtiene Netscape. Generalmente viene acompañado con la extensión. Si se tiene algún problema al momento de instalar la versión de 16 bits que no necesita extensión, puede hacerlo entonces con la extensión de 32 bits.

la presentación de Netscape porque no se tiene acceso al código fuente, pero se proporciona un menú para realizar solo algunos cambios.

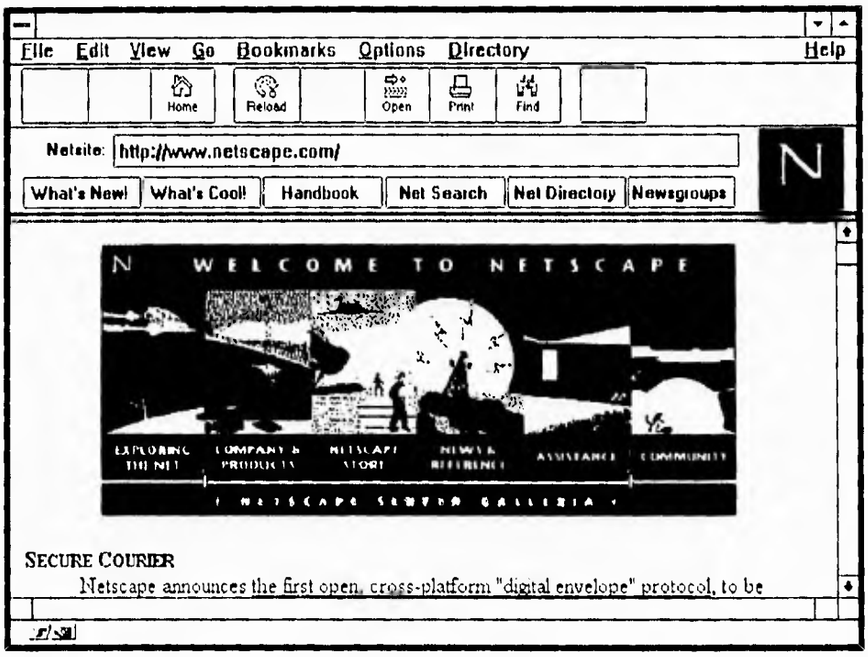


Figura 6.26 Home Page de Netscape

Para cambiar el cambio del "home page" de default se procede a hacer lo siguiente:

- ir al menú de **Options**
- seleccionar **Preferences...**
- en la caja de *Set preferences on* buscar la palabra *Styles*
- se selecciona *Home Page Location* y se introduce la dirección del home page que se desea que se cargue por default (ver figura 6.27).

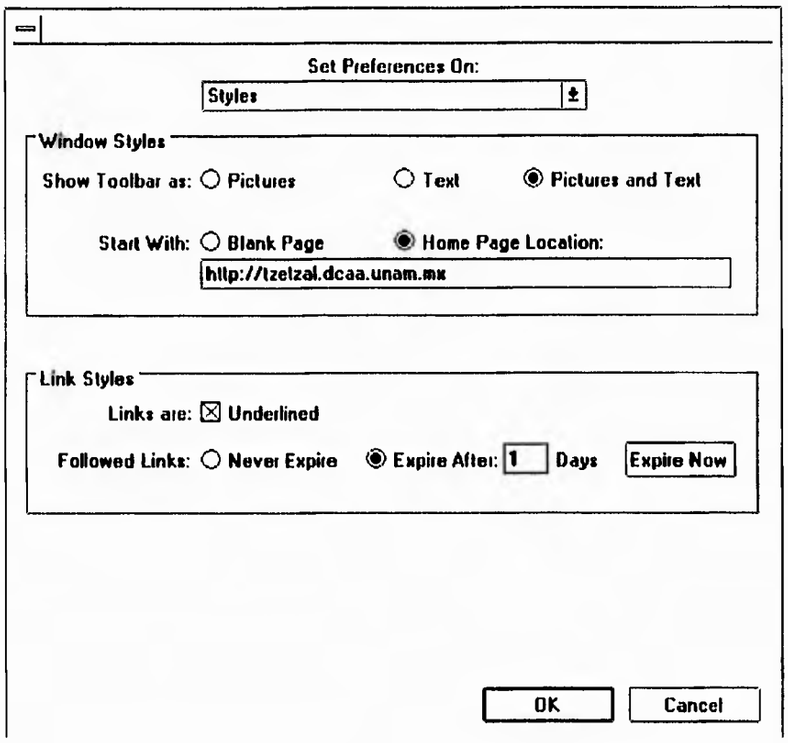


figura 6.27 Cambiando el Home page de defaul para Netscape en PC

Al indicar el home page por default se puede incluir la ruta de donde lo tomará, en nuestro caso el URL para el sistema de inscripción es el siguiente:

http://tzetzal.dcaa.unam.mx:87/admonini.html

el número 87 al que se hace referencia en el URL, es el puerto por el que se comunicará con servidor (recuerde que se instaló un segundo servidor) y es necesario, ya que con esto se tiene acceso a la parte del sistema correspondiente al administrador. La pantalla de presentación o el "home page" que tiene el sistema es como el que se muestra a continuación:

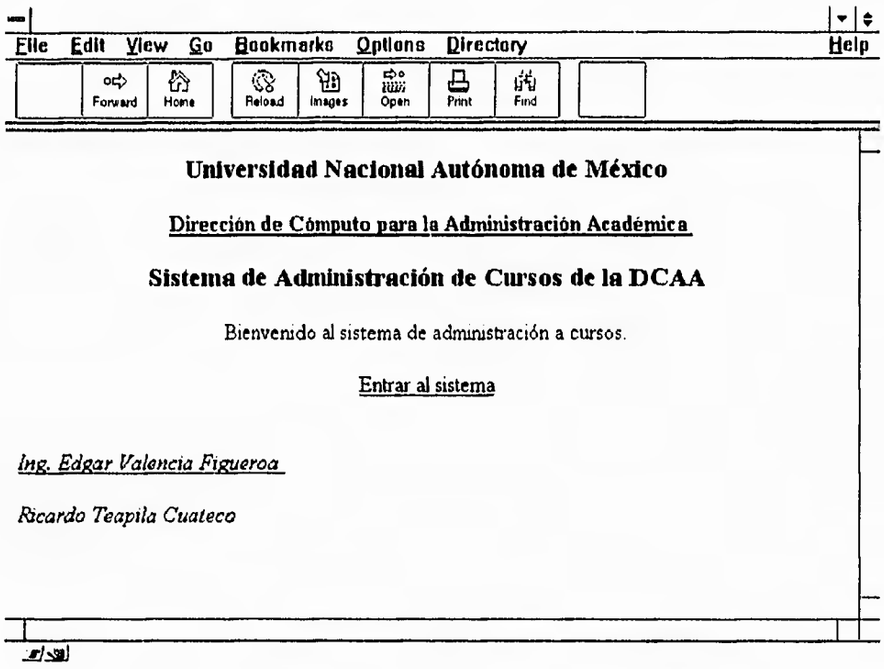


figura 6.28 Home page del sistema administrador de inscripciones a cursos de l a DCAA

Cuando entramos al sistema se presenta una ventana pidiendo la clave y password de acceso (se comentó esto anteriormente con la clave supervisor). Observe que es parecida a la que se vió en el sistema X-Window.

En el momento de introducir la clave (username) y el password correcto, se tiene acceso inmediatamente al sistema. Una vez dentro de él, se presenta en un menú las opciones en forma de ligas a otros menús.

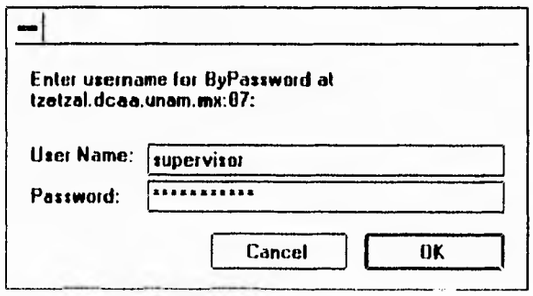


figura 6.29 Pantalla de validación de acceso al sistema

Haciendo un paréntesis, desde el momento en que el usuario entra a Netscape puede personalizarlo. Una de las partes que puede eliminar para apreciar mejor los documentos, son las barras de herramientas, el letrero de los URL's y demás, así como en pasos anteriores se describió la forma de cambiar el "home page" de default, ahora explicaremos cómo eliminar esas partes.

El primer paso es, una vez que se ha cargado Netscape, hay que acceder al menú **Options**. En él se encuentran las siguientes opciones que pueden ser habilitadas o deshabilitadas:

Preferences...

Show Toolbar 1

Show Location 2

Show Directory Buttons 3

Auto Load Images 4

Show FTP File Information 5

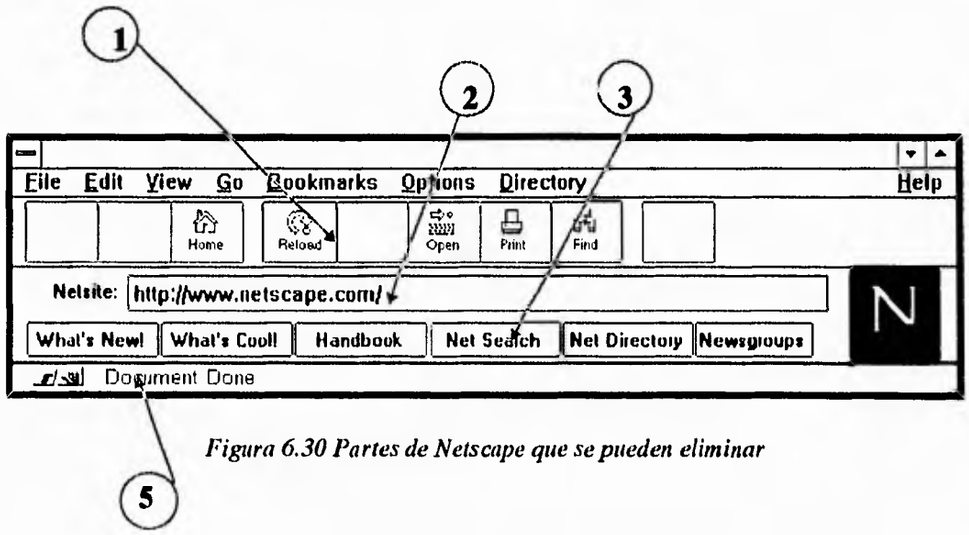


Figura 6.30 Partes de Netscape que se pueden eliminar

Si damos un *click* sobre la opción que deseamos cambiar aparecerá marcada para saber que se ha habilitado la opción. Si usted lo prueba verá que los cambios aparecerán inmediatamente[10].

Continuando con la descripción de entrada al sistema en Netscape, el usuario apreciará una ventana como lo muestra la figura 6.30, donde se encuentran varias ligas hacia otros documentos (frases subrayadas y en color azul).

Por ejemplo, si deseamos dar de alta un curso, damos un *click* sobre la palabra CURSOS, posteriormente se seleccionará al área de cómputo a la que pertenecerá, observe que la *forma* está en base a menús desplegables y botones los cuales se forman con información directamente de la Base de Datos utilizando CGI's.

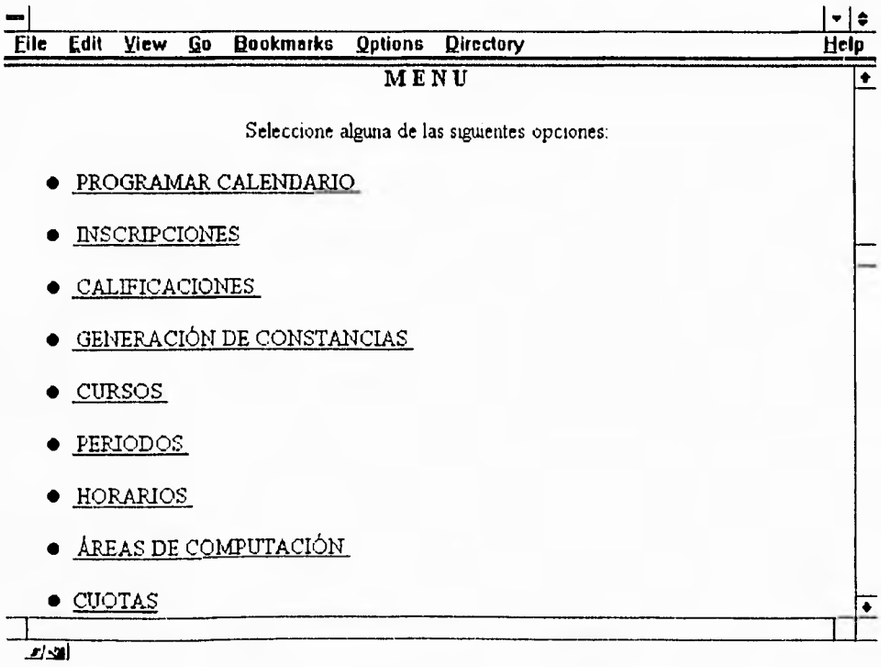


Figura 6.31 Menú del sistema administrador de inscripciones

[10] En algunas de las imágenes que le son presentadas al lector se incluyen pantallas de Netscape con las opciones deshabilitadas para poder apreciar mejor el ejemplo. No indicando con esto que es la forma en que le aparecerán finalmente al usuario del sistema.

En el ejemplo que se muestra en la figura 6.32 se presenta una *forma* con menús desplegables para dar de alta un curso. El usuario solo tiene que llenar los campos que se piden y escoger la opción adecuada cuando exista algún menú. Recuerde que este ejemplo fue explicado anteriormente.

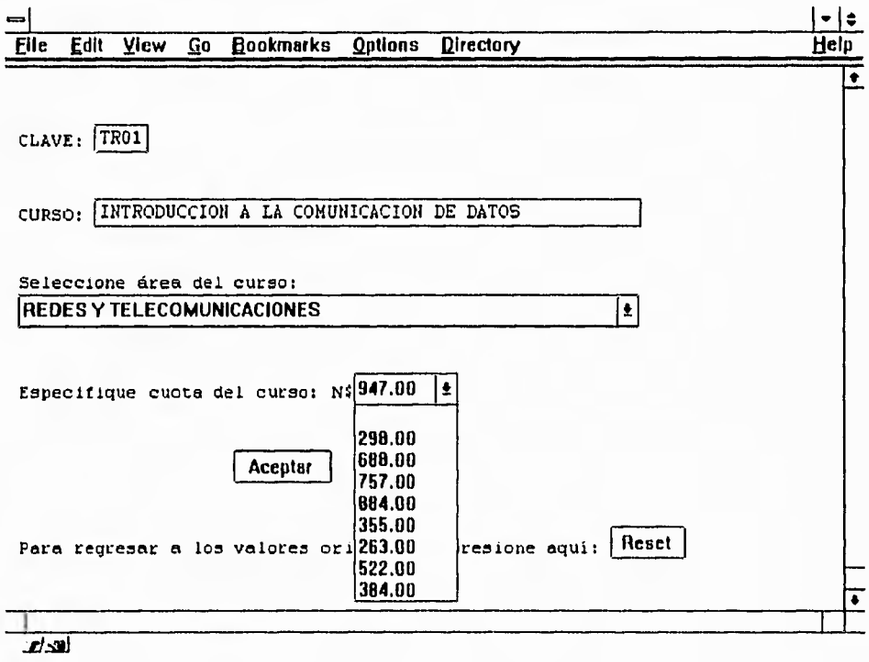


Figura 6.32 Ejemplo de una forma para dar de alta un curso

Las figuras 6.33a y 6.33b muestran los campos para programar un curso en el calendario actual.

Algo muy importante que se debe hacer notar, es que las *formas* no son hechas por el operador del sistema, sino que son generadas dinámicamente por CGI's. Es decir, el programa toma datos de las tablas en Sybase para realizar la *forma*. Por ejemplo, para programar un curso, es necesario saber el nombre del mismo, horario, período y aula. Si la información que se requiere (horarios, períodos, etc.) no existe en la Base de Datos, el sistema no permitirá el uso de ésta opción indicando que falta información para programar un curso.

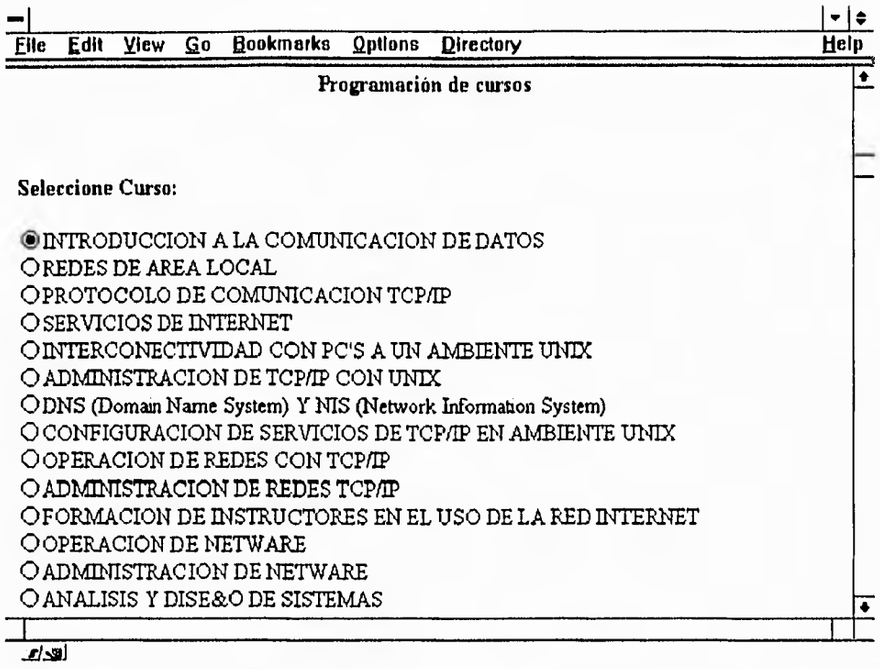


Figura 6.33a Programación de cursos

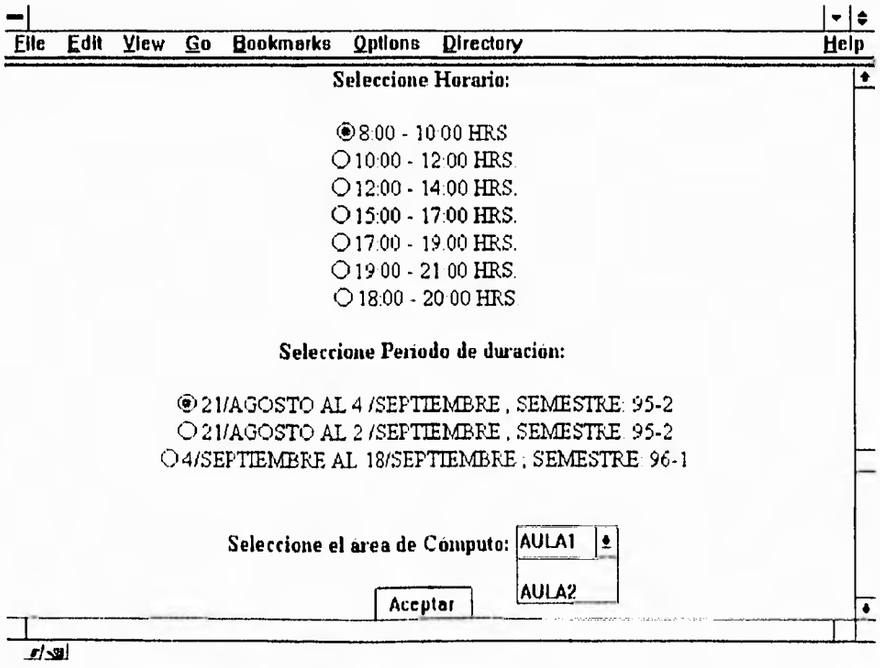


Figura 6.33b Programación de cursos

Otro aspecto muy importante es que sólo se muestra al usuario la información válida, no tiene caso mostrarle períodos atrasados o de semestres anteriores. Con ayuda del sistema operativo, podemos comparar la fecha de la consulta con la fecha del período. Si todavía está dentro del rango, se muestra ese período al usuario, de lo contrario, no aparecerá.

Más detalles del sistema se muestran en el Manual del Usuario.

6.4.5 Pruebas Finales

La etapa final del sistema de Inscripciones corresponde a la realización de pruebas al mismo, para verificar su funcionamiento.

Las pruebas consistieron básicamente primero, de realizar inscripciones desde otro lugar de la red Internet, utilizando Netscape para PC y Netscape/Mosaic para sistema X-Window, los errores más comunes que encontramos fueron las rutas de los URL's que generan las ligas hacia otros documentos. Por ejemplo, si se define una forma de la siguiente manera:

```
<form method="HTTP" action="http://tzetzal:87/cgi-bin/prueba" >
```

Cuando Mosaic trate de recuperar al CGI, lo buscará en la máquina llamada tzetzal. Sin embargo, si no se especifica el dominio del servidor y se está accediendo al documento desde otro dominio, lógicamente no se encontrará al servidor. Por otra parte, recuerde que en el sistema tenemos CGI's que generan formas en HTML para ser visualizadas desde Mosaic (como si las etiquetas fueran enviadas a pantalla). Por lo tanto, debemos tener cuidado con las rutas de los URL's que se están usando así como el número de puerto del servidor.

El sistema fue realizado en la Dirección de Cómputo para la Administración Académica, en este lugar, se encuentra el servidor http del sistema de inscripciones

(y la Base de Datos). Se probó desde el Centro de Cálculo de la Facultad de Ingeniería e inclusive desde la Universidad de Manchester en Inglaterra. Los errores fueron anotados y corregidos.

En la parte del administrador del sistema, después de haber instalado Netscape en la Computadora Personal (que utilizará el administrador) procedimos a probarlo, también al tratar de accederlo desde otra máquina con diferente dirección IP, el examinador Mosaic no pudo recuperar documento alguno.

Volvamos al ejemplo mostrado en las figuras 6.19, 6.20, 6.21, 6.22 y 6.23, recuerde que corresponde a un proceso de inscripción. El siguiente paso, es que el alumno acuda a las oficinas de la DCAA a pagar el curso al que se inscribió. El administrador del sistema solicitándole su clave (RFC), podrá consultar los cursos a los cuales está inscrito en ese momento. Continuando con el ejemplo, las figuras 6.34, 6.35 y 6.36 muestran la forma de confirmar la inscripción de un alumno que previamente se inscribió desde el Kiosco de Información.

Observe que en la Figura 6.35 aparece el curso al que el usuario se inscribió en el ejemplo de la Figura 6.23. La ventaja es que el administrador en este caso, únicamente confirma inscripciones y con ello, agiliza este trámite escolar.

Al terminar el proceso de inscripción, los datos del alumno serán borrados de la tabla temporal para pasarlos a la tabla de inscripciones para que en ese mismo registro, quede asentada su calificación posteriormente.

Más detalles del sistema se muestran en el Manual de Usuario (Apéndice F).

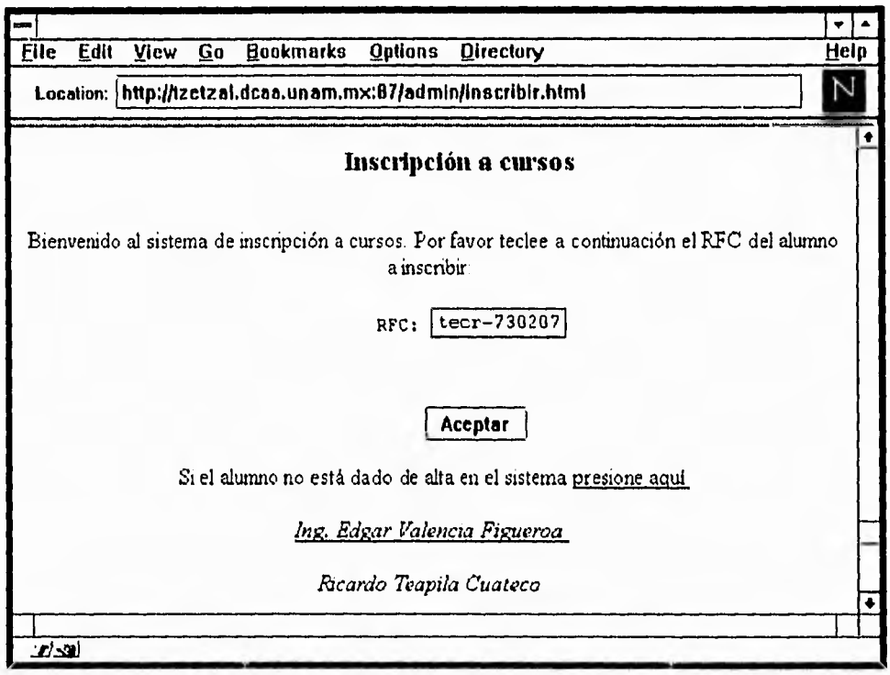


Figura 6.34 Se presenta una pantalla pidiendo la clave del alumno (RFC)

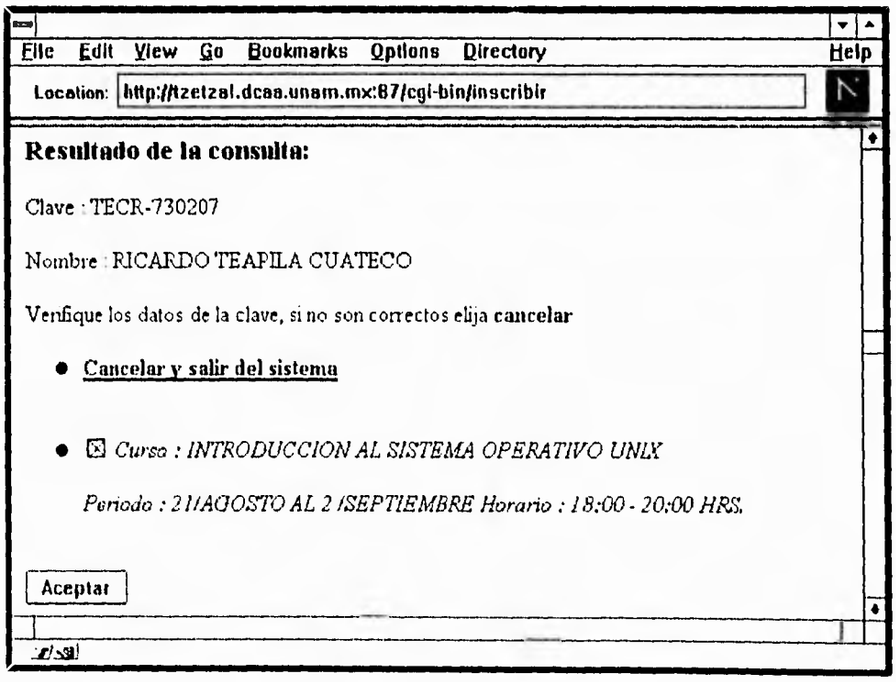


Figura 6.35 Se consulta a la Base de Datos para averiguar los cursos a los cuales se inscribió el alumno

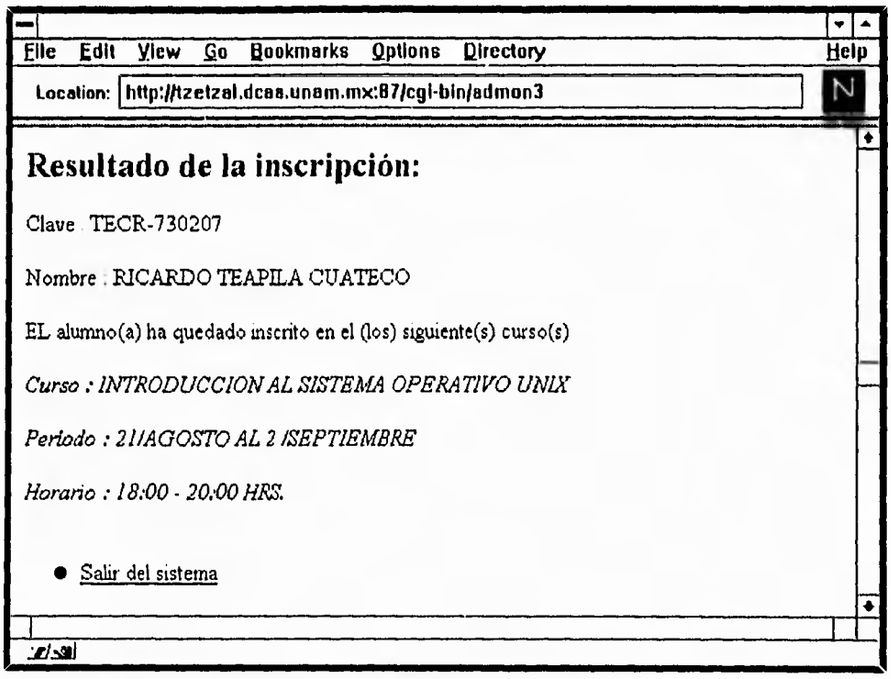
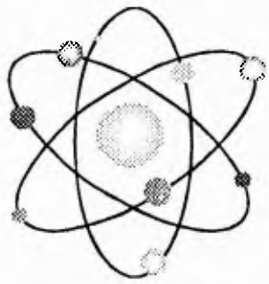


Figura 6.36 Una vez efectuado el pago del curso correspondiente, el alumno queda formalmente inscrito al grupo que había seleccionado en el Kiosco

Capítulo 7



Perspectivas de Desarrollo

Las perspectivas de desarrollo del proyecto *Kioscos de Información* son muy diversas y prometedoras ya que es aplicable para solucionar problemas reales. En el capítulo 2, mostramos algunos ejemplos de kioscos que se han desarrollado en los Estados Unidos.

Hasta el momento en que se escribe esta tesis, se han tenido avances en la tecnología de hardware y software de computadoras.

1. Tenemos ahora la versión comercializada de Netscape, que como visualizador está abarcando el mercado de las computadoras personales. Cuando comenzamos a desarrollar el proyecto (que fue a finales de 1994), no contábamos con Netscape y obviamente su primera versión fue para el sistema X-Window. Posteriormente, se libera la versión para Windows de Microsoft y para mediados de 1995, contamos con la versión 1.1 de este software. Esta versión es mucho más eficiente que las anteriores, soportando el lenguaje HTML plus e imágenes en formato JPEG. Si pensamos en el sistema de inscripciones que desarrollamos en este trabajo, podemos darnos cuenta que ya no sería necesario tener una estación de trabajo, la cual sería reemplazada por una computadora personal, considerando que se ofrecería el servicio de inscripciones únicamente y no el de información

general que incluya videos, imágenes y sonidos puesto que la cantidad de procesamiento que se requiere para este tipo de archivos es aún muy grande. Sin embargo, los kioscos trabajarían en forma similar a los cajeros automáticos de los bancos, dando un servicio específico. Con esto, podemos reducir el costo del kiosco en forma significativa.

2. Dados los avances tecnológicos en las computadoras personales, podemos hasta cierto punto y dependiendo de la aplicación, reemplazar a la estación de trabajo por una PC más o menos con una arquitectura robusta y recursos suficientes que se requieren. Por ejemplo, tipo de procesador, cantidad de memoria, espacio en disco, etc.

3. Dentro de la Red Internet ya existen listas de servidores Web en todo el mundo. Por ejemplo, el "home page" de la DCAA se incluye en algunas de ellas, de esta forma, los demás usuarios de la red saben de nuestra existencia.

4. Actualmente, Sun Microsystems está desarrollando un nuevo visualizador, se llama *Hotjava*, el cual incluye ya un lenguaje programable que nos permite realizar aplicaciones interactivas con el usuario, no únicamente con *formas* en hipertexto sino con imágenes fijas y en movimiento. Esto demuestra que organizaciones comerciales e importantes como Sun Microsystems se preocupan por comercializar productos como los visualizadores, los cuales están pasando más de los laboratorios de las Universidades al mercado comercial.

Como se pudo observar en el capítulo 6, un caso de aplicación fue el sistema de inscripciones a cursos de la DCAA, donde involucramos a Sybase como manejador de base de datos. De esta forma, podemos iniciar una etapa en la cual se

tomen en cuenta otros manejadores y comunicarlos entre ellos vía red a través de Mosaic y CGI's para el desarrollo de nuevas aplicaciones. Por ejemplo, podríamos pensar en una base de datos en Oracle, otra en Informix y otra en Sybase, donde los datos se encuentren en servidores y tablas diferentes. Utilizando programación de clientes, las operaciones sobre los datos se generan en una forma sencilla, donde el usuario no se entera que al insertar, consultar, modificar y borrar información, la transacción es local o remota, y que los datos pueden estar almacenados en diferentes partes de la red. Si incluimos a Mosaic como interfaz gráfica, la aplicación se vuelve aún más poderosa, por las características que este ambiente gráfico presenta y las cuales fueron mencionadas en el capítulo 4.

Dentro de la Universidad, y específicamente hablando del Centro de Enseñanza de Lenguas Extranjeras (CELE), se está proponiendo que las inscripciones a los cursos de esa dependencia, se realice por medio de kioscos. Actualmente, estamos estudiando esa posibilidad y esperamos que dentro de poco tiempo se comience con el proyecto.

Sin embargo, el *Kiosco de Información*, al utilizar un ambiente gráfico, programación de clientes y bases de datos, también se puede contemplar como un producto comercial. Sólo por mencionar algunos ejemplos, se tienen los siguientes:

1. En el sector bancario tendría la perspectiva de desarrollo en que las *formas* para los posibles tarjeta-habientes, se podrían llenar directamente en el kiosco para la inmediata elaboración de su tarjeta de crédito. De esta manera, existiría una red de terminales que atenderían estas solicitudes.

2. Por otra parte, el diseño de formas y CGI's están siendo cada vez más utilizados en la Red Internet por varias organizaciones comerciales. Tenemos el caso de las agencias de viajes, las cuales a través de páginas de presentación en hipertexto,

podrían promover lugares turísticos. Los clientes pueden hacer reservaciones y solicitar información detallada sobre algún paquete de viaje en especial.

3. Los hoteles serían otro caso en el cual podemos instalar kioscos. Cuando un huésped llega, puede consultar información acerca del hotel, como sería: precios de cuartos, servicios que ofrece, lugares de interés, etc. Ahora, que si lo que se desea es solo viajar, podemos tener Kioscos en las terminales de autobuses o aeropuertos, donde el usuario puede consultar las diferentes líneas, destinos, horarios e itinerarios de los viajes y si es posible comprar su boleto o reservar su lugar de viaje desde ahí.

4. Los Kioscos pueden aparecer en lugares atractivos o concurridos como son los centros comerciales, donde los clientes pueden consultar ofertas y localizar por ejemplo, a aquellas tiendas que venden productos especiales así como su ubicación. La gran variedad de información hace que estos sistemas funcionen y que sean de interés para cualquier tipo de persona. Es posible que desde cualquier lugar se pueda tener acceso a la información y realizar compras inclusive desde los propios hogares a través de una computadora.

5. ¿Qué pasa si usted al llegar a una tienda de discos desea comprar el éxito del momento pero no recuerda el nombre de la canción o el nombre del cantante?, tendría que estar buscando por artista o por compañía discográfica. Con ayuda de un Kiosco, se podría consultar los artistas por orden alfabético, por compañía a la que pertenece, etc., y una vez identificado se puede ver una breve reseña de él, sus éxitos, y su canción más reciente e inclusive puede escuchar un fragmento de la misma y saber así, si es la que busca o no.

6. Otro caso lo tenemos en las conferencias donde podemos contemplar una posible aplicación de video conferencia en kioscos. Un expositor podría estar por ejemplo en Monterrey, y la misma conferencia podría estar recibiéndose en el Distrito Federal en una terminal, el expositor podría interactuar directamente con el auditorio no importando su localización geográfica.

En general, se puede ver que este proyecto tiene un sinfín de perspectivas, ya que el principio es que el manejo de la información sea interactiva con los usuarios finales, auxiliándonos de gráficos, sonidos y videos. En su conjunto, podemos diseñar aplicaciones para todas las áreas del conocimiento, industria, comercio, educación, etc.

Capítulo 8

Conclusiones

Antes de concluir necesitamos retomar el objetivo inicial del proyecto, que consiste en la realización de Kioscos de Información, los cuales ofrecerán a la comunidad universitaria y público en general, un servicio de carácter informativo.

Con la realización del proyecto logramos diseñar el prototipo del kiosco. Si analizamos su desarrollo, podemos notar que los kioscos no sólo son terminales para difundir información, sino que pueden interactuar con el usuario aplicándolo a la solución de un problemas diversos.

Visualizadores gráficos como Mosaic y Netscape, se están difundiendo enormemente dentro de la Red Internet, tendiendo a convertirse en estándares para el manejo y recuperación de la información. En los últimos dos años, el número de servidores Web creció de tan sólo unas decenas a algunos miles repartidos por todo el mundo; incluyendo a instituciones educativas, gubernamentales y comerciales.

Como lo mencionamos en las Perspectivas de Desarrollo, este tipo de visualizadores están pasando más de los laboratorios de las Universidades al mercado como productos comerciales, convirtiéndose en un arma competitiva para las empresas. Cada una de ellas, puede promover con ayuda de hipertexto e hipermedia, sus productos comerciales e incluso aceptar pedidos y sugerencias por parte de los clientes a través de la red.

A medida que desarrollamos este proyecto, nos dimos cuenta de la importancia que tiene el estar conectado con las diferentes fuentes de información dentro de la Red Internet. Gracias a ellas, nos enteramos de la existencia de variadas herramientas las cuales podemos explotar permitiéndonos con ello, descubrir nuevos elementos y métodos para realizar mejoras, correcciones y adaptaciones a los sistemas de información.

La computación evoluciona día con día, cuando comenzamos a desarrollar el proyecto de los kioscos, usamos herramientas de software que parecían no tener aplicaciones, pero actualmente están fijando estándares en el manejo de la información.

Con el presente trabajo, adquirimos grandes experiencias al conjuntar 3 tipos de programas diferentes y poderosos: un sistema operativo (Unix), un manejador de Base de Datos (Sybase) y una GUI (Mosaic), todos ellos en un ambiente de red. El primero, corriendo casi en cualquier plataforma de hardware siguiendo la filosofía de los sistemas abiertos, el segundo, un manejador que tiene la capacidad para interactuar incluso con otros manejadores y por lo tanto, con otras Bases de Datos. Por último, una GUI que como interfaz de usuario gráfica hace más amigable a la aplicación. Al operar conjuntamente estos programas y en red, tenemos un claro ejemplo del concepto de Interoperabilidad.

Es evidente que logramos integrar en una aplicación varios conceptos tales como: multimedia, arquitectura cliente/servidor, redes, ambientes gráficos, Bases de Datos, etc. Sin embargo, esto no se hubiera logrado sin las bases y conocimientos adquiridos en la Facultad de Ingeniería los cuales fueron la base fundamental a partir de la cual, logramos resolver un problema con nuevos esquemas de modernización y automatización de procedimientos. Además, nos permitieron comprender conceptos nuevos para nosotros, así como la importancia que tiene el hecho de optimizar recursos en materia de cómputo.

A medida que desarrollamos este proyecto, nos dimos cuenta de la importancia que tiene el estar conectado con las diferentes fuentes de información dentro de la Red Internet. Gracias a ellas, nos enteramos de la existencia de variadas herramientas las cuales podemos explotar permitiéndonos con ello, descubrir nuevos elementos y métodos para realizar mejoras, correcciones y adaptaciones a los sistemas de información.

La computación evoluciona día con día, cuando comenzamos a desarrollar el proyecto de los kioscos, usamos herramientas de software que parecían no tener aplicaciones, pero actualmente están fijando estándares en el manejo de la información.

Con el presente trabajo, adquirimos grandes experiencias al conjuntar 3 tipos de programas diferentes y poderosos: un sistema operativo (Unix), un manejador de Base de Datos (Sybase) y una GUI (Mosaic), todos ellos en un ambiente de red. El primero, corriendo casi en cualquier plataforma de hardware siguiendo la filosofía de los sistemas abiertos, el segundo, un manejador que tiene la capacidad para interactuar incluso con otros manejadores y por lo tanto, con otras Bases de Datos. Por último, una GUI que como interfaz de usuario gráfica hace más amigable a la aplicación. Al operar conjuntamente estos programas y en red, tenemos un claro ejemplo del concepto de Interoperabilidad.

Es evidente que logramos integrar en una aplicación varios conceptos tales como: multimedia, arquitectura cliente/servidor, redes, ambientes gráficos, Bases de Datos, etc. Sin embargo, esto no se hubiera logrado sin las bases y conocimientos adquiridos en la Facultad de Ingeniería los cuales fueron la base fundamental a partir de la cual, logramos resolver un problema con nuevos esquemas de modernización y automatización de procedimientos. Además, nos permitieron comprender conceptos nuevos para nosotros, así como la importancia que tiene el hecho de optimizar recursos en materia de cómputo.

Con los kioscos de información, marcamos una nueva etapa en el desarrollo de sistemas de información. Ahora les damos un nuevo aspecto, interactuamos con aplicaciones poderosas como son los manejadores de Bases de Datos, podemos hacerlos más amigables y en general hacerlos interactuar con otras aplicaciones.

Actualmente en la Dirección de Cómputo para la Administración Académica (DCAA), se están promoviendo por medio de la Subdirección de Recursos Humanos, cursos de preparación dirigidos a becarios y usuarios externos, donde se incluyan tópicos de computación relacionados con Mosaic, programación de clientes en Sybase e Interoperabilidad Mosaic/Sybase. Esto indica que en poco tiempo, los sistemas desarrollados en algunas dependencias de la UNAM tendrán un enfoque como el de los *Kioscos de Información*. Una vez más, al igual que otros compañeros egresados de la Facultad de Ingeniería, aportamos nuevos conocimientos en computación a ésta, nuestra máxima casa de estudios: La **Universidad Nacional Autónoma de México**.

Apéndice A

Contenido del archivo httpd.conf del kiosco

Archivo de configuracion general

ServerType. Directiva que puede ser inetd o standalone.

ServerType standalone

Port: Numero de puerto para el servidor.

Port 80

Si deseas que httpd corra como un usuario y grupo diferente, deberas modificar
esta directiva.

User/Group: El nombre (o #numero) del usuario/grupo.

User root
Group root

ServerAdmin: Direccion de correo electronico donde se pueden reportar problemas.

ServerAdmin edgar@xel-ha.dcaa.unam.mx

ServerRoot: Directorio raiz del servidor
ServerRoot /home/http/

ErrorLog: Nombre del archivo donde se almacenan errores del servidor
Comienza con /, y se encuentra bajo ServerRoot

ErrorLog logs/http_error.log

TransferLog: Archivo que registra los accesos a nuestro servidor
comienza con /, y se encuentra bajo ServerRoot

TransferLog logs/accesos.log

PidFile: Archivo que contiene el pid asociado a nuestro servidor
PidFile logs/httpd.pid

```
# Archivo de configuracion de acceso global. Relativo a ServerRoot
AccessConfig conf/access.conf

# Archivo de configuracion de recursos:
ResourceConfig conf/srm.conf

# Archivo de configuracion de tipos, es decir, como maneja las extensiones de archivos.
TypesConfig conf/mime.types

# ServerName:  Nombre de tu maquina, el cual sera valido para clientes de la Web

ServerName xel-ha.dcaa.unam.mx
```

Contenido del archivo srm.conf

```
# Con este archivo, puedes definir un directorio de documentos HTML para tu servidor

# DocumentRoot: Directorio a partir del cual, tu servidor recuperara documentos.
# Por default, todas las peticiones son tomadas de este directorio

DocumentRoot /

# UserDir: Nombre del directorio donde residen los documentos HTML de los usuarios.
# Se encuentra bajo /www

UserDir html_docs

# DirectoryIndex: Nombre del documento que sera recuperado por default

# DirectoryIndex presenta-dgsca/DCAA.html

# Se definen iconos por default

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: Define icono para archivos que no se puedan desplegar como imagen.

DefaultIcon /icons/unknown.xbm

AccessFileName: Nombre del archivo donde residen nombres y claves de usuarios
para documentos protegidos.

AccessFileName .htaccess

DefaultType: es el tipo por default para documentos que el servidor va a manejar
DefaultType text/plain

Alias: Simplemente se defienen alias para directorios

Alias /icons/ /home/http/icons/

ScriptAlias: Nombre del directorio donde residen los CGI's de los usuarios.

ScriptAlias /cgi-bin/ /home/http/cgi-bin/

Contenido del archivo access.conf

access.conf: Configuracion de acceso global al sistema.
Directorio que permite ser accedido por todos los clientes de la Web
<Directory /home/http/htdocs>

La siguiente directiva puede ser "None", "All" o "FollowSymLinks" dependiendo de
las caracteristicas de tu directorio. En nuestro caso, el servidor seguira ligas simbolicas
bajo este directorio.

Options Indexes FollowSymLinks

La siguiente directiva controla permisos de acceso sobre algun directorio.
Puede ser "None" (nadie entra) o "All" (todos pueden entrar) .

AllowOverride All

Controles que pueden usarse para dar permiso de acceso a ciertas maquinas

<Limit GET>
order allow,deny
allow from all
</Limit>
</Directory>

```
# Modificacion para seguridad: 17/Mar/95

<Directory /usr>
AllowOverride None
Options None
</Directory>

<Directory /opt>
AllowOverride None
Options None
</Directory>

<Directory /home>
AllowOverride All

<Limit GET>
order allow,deny
allow from all

order deny,allow
deny from all
# Permitimos el acceso unicamente a las siguientes maquinas.
allow from xel-ha.dcaa.unam.mx
allow from 132.248.103.200
allow from 132.248.27
allow from 132.248.63
allow from 132.248.170
</Limit>

Options None
</Directory>
```

Apéndice B

Contenido del archivo 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';
         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 C

Contenido del archivo syb2.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sybfront.h>
#include <sybdb.h>
#include <syberror.h>
#include <ctype.h>

#define MAX_ENTRIES 10000

typedef struct {
   char *name;
   char *val;
} entry;

/* Las siguientes funciones se definen 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[]) {
   entry entries[MAX_ENTRIES];
   register int x,m=0;
   int cl,i;

/* Las siguientes variables son definidas para el cliente en sybase */

   DBPROCESS   *dbproc;
   LOGINREC    *login;
   RETCODE     return_code;
   DBCHAR nombre[30],clave[13],ap_paterno[15],ap_materno[15];
   char *autor[10];
   char *nom;

/* Hacemos uso de la directiva "Content-type" para generar los mensajes
```

```
en hipertexto */

   printf("Content-type: text/html%c%c",10,10);

   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 script solo puede usarse para decodificar datos de una forma \n");
      exit(1);
   }
   cl = atoi(getenv("CONTENT_LENGTH"));
   if (cl == 106)
     {
     printf("<pre>\n");
     printf("<h3>                Usted no ha llenado correctamente la forma</h3>\n\n");
     printf("<h3>  Favor de seleccionar alguna de las siguientes opciones:
</h3>\n\n\n");
     printf("<img
src=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/line.blue.gif\">\n\n\n");
     printf("<a
href=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/pc/forma.html\"><h3>Re
greso a la forma anterior</h3></a>\n\n");
     printf("\n\n\n");
     printf("<a
href=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/pc/inscripciones.html\"><
h3>Cancelar</h3></a>\n");
     printf("</pre>");
     exit(1);
     }

/* Procedimiento que decodifca los datos de la entrada estandar */

   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,'=');
   }

/* Convertimos a mayusculas todos los datos */
```

```
    for(x=0; x <= m; x++){
      nom=entries[x].val;
      while (*nom!='\0')
        {
        *nom=toupper(*nom);
        nom++;
        }
      while (*entries[x].val == ' '){
        entries[x].val=entries[x].val++;
        entries[x].val++;
      }
    }

/* Los siguientes condicionales verifican que no se omita algun dato en
la forma que llena el usuario */

    nom=entries[0].val;
    if (*nom == NULL){
      printf("<pre>\n");
      printf("<h3>          Usted no ha especificado correctamente su nombre</h3>\n\n");
      printf("<h3>       Favor de seleccionar alguna de las siguientes opciones</h3>\n\n");
      printf("<img
src=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/line.blue.gif\">\n\n\n");
      printf("<a
href=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/pc/forma.html\"><h3>Re
greso a la forma anterior</h3></a>\n\n");
      printf("<a
href=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/pc/inscripciones.html\"><
h3>Cancelar </h3></a>\n");
      printf("</pre>");
      exit(1);
    }

    if (*entries[1].val == NULL)
    {
      printf("<pre>\n");
      printf("<h3>          Usted no ha especificado su Apellido Paterno</h3>\n\n");
      printf("<h3>       Favor de seleccionar alguna de las siguientes opciones</h3>\n\n");
      printf("<img
src=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/line.blue.gif\">\n\n\n");
      printf("<a
href=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/pc/forma.html\"><h2>Re
greso a la forma anterior</h2></a>\n\n");
      printf("<a
```

```
href=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/pc/inscripciones.html\"><
h3>Cancelar</h3></a>\n");
    printf("</pre>");
    exit(1);
  }
  if (*entries[2].val == NULL)
  {
    printf("<pre>\n");
    printf("<h3>          Usted no ha especificado su Apellido Materno</h3>\n\n");
    printf("<h3>       Favor de seleccionar alguna de las siguientes opciones</h3>\n\n");
    printf("<img
src=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/line.blue.gif\">\n\n\n");
    printf("<a
href=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/pc/forma.html\"><h3>Re
greso a la forma anterior</h3></a>\n\n");
    printf("\n\n\n");
    printf("<a
href=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/pc/inscripciones.html\"><
h3>Cancelar</h3></a>\n");
    printf("</pre>");
    exit(1);
  }
  if (*entries[7].val == NULL)
  {
    printf("<pre>\n");
    printf("<h3>       Usted no ha especificado su a&ntilde;o de Nacimiento</h3>\n\n");
    printf("<h3>       Favor de seleccionar alguna de las siguientes opciones
</h3>\n\n");
    printf("<img
src=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/line.blue.gif\">\n\n\n");
    printf("<a
href=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/pc/forma.html\"><h3>Re
greso a la forma anterior</h3></a>\n\n");
    printf("<a
href=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/pc/inscripciones.html\"><
h3>Cancelar</h3></a>\n");
    printf("</pre>");
    exit(1);
  }

  if (*entries[10].val == NULL)
  {
    printf("<pre>\n");
    printf("<h3>                Usted no ha especificado su RFC</h3>\n");
    printf("<h3>       Favor de seleccionar alguna de las siguientes opciones
</h3>\n\n");
```

```
    printf("<img
src=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/line.blue.gif\">\n\n\n");
    printf("<a
href=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/pc/forma.html\"><h3>Re
greso a la forma anterior</h3></a>\n\n");
    printf("<a
href=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/pc/inscripciones.html\"><
h3>Cancelar</h3></a>\n");
    printf("</pre>");
    exit(1);
  }

/* Comienza la parte del cliente en Sybase */

  if (dbinit() == FAIL)
    {
     printf("<h1>El servidor esta abajo</h1>");
     exit(ERREXIT);
    }
  login = dblogin();
  DBSETLUSER(login,"ricardo");
  DBSETLPWD(login,"ricardo");
  dbproc = dbopen(login,NULL);
  if (dbproc == NULL)
    {
     printf("<h1>El servidor no se encuentra arrriba</h1>");
     exit(ERREXIT);
    }
dbfcmd (dbproc,"select * from claves_alumnos where clave=\"%s\"",entries[10].val);
  if (dbsqlexec(dbproc) == FAIL)
  {
    printf("<h2>Error durante el proceso de acceso al servidor</h2>");
    dbexit();
    exit(ERREXIT);
  }

  dbresults(dbproc);
  if (dbrows(dbproc) == SUCCEED )
    {

      /* Hay resultados a procesar */

      printf("<pre>\n");
    printf("<p>\n");
    printf("<h3>       La clave especificada ya existe</h3> ");
```

```
        printf("<h3>       Favor de seleccionar alguna de las siguientes opciones
</h3>\n\n");
        printf("<img
src=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/line.blue.gif\">\n\n\n");
        printf("<a
href=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/pc/forma.html\"><h3>Re
greso a la forma anterior</h3></a>\n\n");
        printf("<a
href=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/pc/inscripciones.html\"><
h3>Cancelar</h3></a>\n");
        printf("</pre>\n");
        exit(1);
    }
  dbfcmd(dbproc,"insert claves_alumnos values ");

dbfcmd(dbproc,"(\"%s\",\"%s\",\"%s\",\"%s\")",entries[10].val,entries[0].val,entries[1].v
al,entries[2].val);

  if (dbsqlexec(dbproc) == FAIL)
  {
     printf("<h2>Error durante el proceso de acceso al servidor</h2>");
     dbexit();
     exit(ERREXIT);
  }
  dbfcmd(dbproc,"insert alumnos values ");
dbfcmd(dbproc,"(\"%s\",%s,\"%s\",%s,\"%s\",\"%s\",\"%s\",\"%s\")",entries[10].val,e
ntries[5].val,entries[6].val,entries[7].val,entries[3].val,entries[4].val,entries[8].val,entries[9].
val);

  if (dbsqlexec(dbproc) == FAIL)
  {
     printf("<h2>Error durante el proceso de acceso al servidor</h2>");
     dbexit();
     exit(ERREXIT);
  }
  printf("<pre>\n");
  printf("<h3>    La clave ha sido dada de alta con los siguientes datos</h3>");
  printf("<ul>%c",10);
  printf("<li>NOMBRE            : %s\n",entries[0].val);
  printf("<li>APELLIDO PATERNO   : %s\n",entries[1].val);
  printf("<li>APELLIDO MATERNO   : %s\n",entries[2].val);
  printf("<li>DIRECCION          : %s\n",entries[3].val);
  printf("<li>TELEFONO           : %s\n",entries[4].val);
  printf("<li>FECHA DE NACIMIENTO :
%s/%s/%s\n",entries[5].val,entries[6].val,entries[7].val);
  printf("<li>ESCOLARIDAD        : %s\n",entries[8].val);
```

```
   printf("<li>SE INSCRIBE A LOS CURSOS : %s\n",entries[9].val);
   printf("<li>RFC              : %s\n",entries[10].val);
   printf("</ul>%c",10);
   printf("</pre>\n");
   dbnextrow(dbproc);
   dbcanquery(dbproc);

/* Ahora definimos una peque&a forma donde pasamos como parametro
escondido, a la clave del alumno, con el fin de no perderla durante el
proceso de inscripcion */

   printf("<form method=\"POST\"
action=\"http://tzetzal.dcaa.unam.mx/cgi-bin/kiosko_pc/usuario2\">\n");
   printf("<input type=\"hidden\" name=\"order\"
value=\"%s\">\n",entries[10].val);
   printf("<ul>\n");

/* Note que el boton de "Aceptar" tiene un contenido diferente, lo que
ocasiona que cuando el usuario presione este boton, su clave sera pasada
como parametro al CGI usuario2 */

   printf("<li><input type=\"submit\" value=\"Presione aqu&iacute; para
inscribirse a cursos\" >\n");
   printf("</form>\n");
   printf("<p>\n");
   printf("<p>\n");
   printf("</ul>\n");
   dbexit();
   exit(STDEXIT);
}
```

Contenido del archivo usuario1.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sybfront.h>
#include <sybdb.h>
#include <syberror.h>
#define MAX_ENTRIES 10000

typedef struct {
   char *name;
   char *val;
} entry;
```

```
/* Las siguientes funciones se definen 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[]) {
  entry entries[MAX_ENTRIES];
  register int x,m=0;
  int cl;

/* Las siguientes variables son validas para el cliente en Sybase */

  DBPROCESS  *dbproc;
  LOGINREC   *login;
  RETCODE    return_code;
  DBCHAR nombre[30],clave[13],ap_paterno[15],ap_materno[15];
  char *autor[10];
  char *nom;

/* Note que hacemos uso de la directiva "Content-type" para que todos los
mensajes, sean en forma de hipertexto */

  printf("Content-type: text/html%c%c",10,10);
  printf("<title>Resultado de la consulta</title>\n");

  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 script solo puede usarse para decodificar datos de
una forma. \n");
    exit(1);
  }

/* Se obtiene de la variable de ambiente CONTENT_LENGTH, el numero de
caracteres a leer de la entrada estandar */

  cl = atoi(getenv("CONTENT_LENGTH"));
```

```
  if (cl == 4)
    {

/* No hay datos a procesar */

    printf("<pre>\n");
    printf("<h3>                Favor de teclear su RFC como clave de acceso al </h3>\n");
    printf("<h3>                        sistema de inscripciones</h3>\n");
    printf("<p>\n");
    printf("<img
src=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/line.blue.gif\">\n\n\n");
    printf("<a
href=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/pc/inscripciones.html\"><
h3>Regreso a la forma anterior</h3></a>\n");
    printf("<p>\n\n\n");
    printf("<p>\n");
    exit(1);
    printf("</pre>\n");
    }

/* Procedimiento que decodifica los datos de la entrada estandar*/

  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,'=');
  }

/* Convierte a mayusculas los datos capturados */

   for(x=0; x <= m; x++){
     nom=entries[x].val;
     while (*nom!='\0')
       {
       *nom=toupper(*nom);
       nom++;
       }
     while (*entries[x].val == ' '){
       entries[x].val=entries[x].val++;
       entries[x].val++;
     }
   }
```

```
/* Comienza el codigo del cliente en Sybase */

  if (dbinit() == FAIL)
     {
      printf("<h1>El servidor esta abajo</h1>\n");
      exit(ERREXIT);
     }
  login = dblogin();
  DBSETLUSER(login,"ricardo");
  DBSETLPWD(login,"ricardo");
  dbproc = dbopen(login,NULL);
  if (dbproc == NULL)
     {
      printf("<h1>El servidor no se encuentra arrriba</h1>\n");
      exit(ERREXIT);
     }
  dbfcmd (dbproc,"select * from claves_alumnos where clave=\"%s\"",entries[0].val);
  if (dbsqlexec(dbproc) == FAIL)
  {
     printf("<h2>Error durante el proceso de acceso al servidor</h2>\n");
     dbexit();
     exit(ERREXIT);
  }
  dbresults(dbproc);

  dbbind(dbproc,1,STRINGBIND,0,clave);
  dbbind(dbproc,2,STRINGBIND,0,nombre);
  dbbind(dbproc,3,STRINGBIND,0,ap_paterno);
  dbbind(dbproc,4,STRINGBIND,0,ap_materno);
  if (dbrows(dbproc) != SUCCEED )
    {

       /* La clave del alumno no existe en la Base de Datos */

     printf("<p>\n");
     printf("<p>\n");
        printf("<pre>\n");
     printf("<h3>                    La clave especificada no existe</h3> \n");
     printf("<p>\n");
     printf("<h3>              Seleccione alguna de las siguientes opciones:</h3>\n\n");
        printf("<img
src=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/line.blue.gif\">\n\n\n");
     printf("<A
HREF=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/pc/forma.html\"><h3>
Dar de alta</h3></A>\n");
     printf("<p>\n");
```

```
      printf("<A
HREF=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/pc/inscripciones.html\"
><h3>Cancelar</h3></A>\n");
      printf("<p>\n");
      printf("<p>\n\n\n");
         printf("</pre>");
      exit(1);
     }

   dbnextrow(dbproc);
   printf("<pre>\n");
   printf("<H3>                    Los datos de la clave son los siguientes:</H3>\n");
   printf("<img
src=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/line.blue.gif\">\n\n");
   printf("                    Clave       : %s \n\n",clave);
   printf("                    Nombre      : %s \n\n",nombre);
   printf("                    Apellido Paterno : %s \n\n",ap_paterno);
   printf("                    Apellido Materno : %s \n\n",ap_materno);
   printf("<p>\n");
   printf("<img
src=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/line.blue.gif\">\n");
   printf("<h4>                    Verifique sus datos, si no son correctos elija
cancelar</h4>\n");
   printf("</pre>\n");

/* Definimos una peque&a forma donde pasamos al CGI usuario2, la clave
del alumno como parametros. El usuario lo vera el campo porque esta
"escondido" */

   printf("<form method=\"POST\"
action=\"http://tzetzal.dcaa.unam.mx/cgi-bin/kiosko_pc/usuario2\">\n");
   printf("<input type=\"hidden\" name=\"order\" value=\"%s\">\n",clave);
   printf("<pre>\n");
   printf("      <input type=\"submit\" value=\"Presione aqu&iacute; para
continuar\">\n");
   printf("</form>\n");
   printf("<p>\n");
   printf("<A
HREF=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/pc/inscripciones.html\"
><h4>Cancelar</h4></A>\n");
   printf("<p>\n\n");
   printf("<p>\n");
   printf("</pre>\n");
   dbexit();
   exit(STDEXIT);
}
```

Contenido del archivo usuario4.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sybfront.h>
#include <sybdb.h>
#include <syberror.h>
#include <time.h>

#define MAX_ENTRIES 10000

typedef struct {
   char *name;
   char *val;
} entry;

/* Las siguientes funciones se definen 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[]) {
   entry entries[MAX_ENTRIES];
   register int x,m=0;
   int cl,cont,n_mes,band;
   struct tm *punt;
   time_t lt;

/* Variables definidas para el cliente en Sybase */

   DBPROCESS   *dbproc;
   LOGINREC    *login;
   RETCODE     return_code;
   DBCHAR
curso[80],mes_ini[15],dia_fin[2],mes_fin[15],hor_ini[10],hor_fin[10],cve_per[3],cve_hor[3];
   DBINT       dia_ini;
   char *mes,*dia;

/* Hacemos uso nuevamente de la directiva "Content-type" */

   printf("Content-type: text/html%c%c",10,10);
```

```
  printf("<title>Sistema de Inscripci&oacute;n a Cursos</title>\n");

  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 script solo puede usarse para decodificar datos de una forma. \n");
    exit(1);
  }

/* Obtenemos la longitud de la cadena a leer de la entrada estandar */

  cl = atoi(getenv("CONTENT_LENGTH"));

/* Procedimiento que decodifica los datos de la entrada estandar */

  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,'=');
  }
  if (m == 0)
  {
  printf("<h2>Favor de seleccionar los cursos:</h2>\n");
  printf("<p><ul>\n");
  printf("<li><a
href=\"http://tzetzal.dcaa.unam.mx/cgi-bin/kiosko_pc/usuario2\">Volver a
intentar</a><p>\n");
  printf("<li><a
href=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/pc/inscripciones.html\"><
h3>Cancelar</h2></a><p>\n");
  exit(1);
  }
  printf("<pre>\n");
  printf("<H3>                    Universidad Nacional Aut&oacute;noma de
M&eacute;xico </H3>\n");
  printf("\n");
  printf("<h3>       Direcci&oacute;n General De Servicios de C&oacute;mputo
Acad&eacute;mico</h3> \n");
  printf("\n");
```

```
  printf("<img
src=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/line.blue.gif\">\n");
  printf("</pre>\n");

/* Comienza el codigo del cliente en Sybase */

  if (dbinit() == FAIL)
    {
     printf("<h1>El servidor esta abajo</h1>\n");
     exit(ERREXIT);
    }
  login = dblogin();
  DBSETLUSER(login,"ricardo");
  DBSETLPWD(login,"ricardo");
  dbproc = dbopen(login,NULL);
  if (dbproc == NULL)
     {
      printf("<h1>El servidor no se encuentra arriba</h1>\n");
      exit(ERREXIT);
     }
  lt = time(NULL);
  punt = localtime(&lt);
  printf("<p>\n");
  printf("<p>\n");
  switch (punt->tm_mon+1) {
        case 1: mes="ENERO";
              break;
        case 2: mes="FEBRERO";
                break;
        case 3: mes="MARZO";
                break;
        case 4: mes="ABRIL";
                break;
        case 5: mes="MAYO";
                break;
        case 6: mes="JUNIO";
                break;
        case 7: mes="JULIO";
                break;
        case 8: mes="AGOSTO";
                break;
        case 9: mes="SEPTIEMBRE";
                break;
        case 10: mes="OCTUBRE";
                break;
        case 11: mes="NOVIEMBRE";
```

```
                break;
        case 12: mes="DICIEMBRE";
                break;
  }

  dbfcmd(dbproc,"select
dia_ini,mes_ini,dia_fin,mes_fin,hor_ini,hor_fin,calend.cve_per,calend.cve_hor
from calend,horario,periodo where calend.cve_cur=\"%s\" and
calend.cve_per=periodo.cve_per and
calend.cve_hor=horario.cve_hor",entries[1].val);
/* and convert(int,periodo.dia_ini) >= %d",entries[1].val,punt->tm_mday);*/

  if (dbsqlexec(dbproc) == FAIL)
  {
       printf("<h2>Error durante el proceso de acceso al servidor</h2>\n");
       dbexit();
       exit(ERREXIT);
  }
  while (dbresults(dbproc) != NO_MORE_RESULTS) {
     dbbind(dbproc,1,INTBIND,0,&dia_ini);
     dbbind(dbproc,2,NTBSTRINGBIND,0,mes_ini);
     dbbind(dbproc,3,STRINGBIND,0,dia_fin);
     dbbind(dbproc,4,STRINGBIND,0,mes_fin);
     dbbind(dbproc,5,STRINGBIND,0,hor_ini);
     dbbind(dbproc,6,STRINGBIND,0,hor_fin);
     dbbind(dbproc,7,STRINGBIND,0,cve_per);
     dbbind(dbproc,8,STRINGBIND,0,cve_hor);
     if (dbrows(dbproc) != SUCCEED )
     {

        /* No existen horarios programados para el curso */

                printf("<p>\n");
        printf("<p>\n");
                printf("<pre>\n");
        printf("<h3>      No existen horarios programados para este curso</h3>\n");
                printf("<p>\n");
        printf("<p>\n");
        printf("<h3>            Seleccione alguna de las siguientes opciones: </h3>\n");
                printf("<p>\n");
                printf("\n\n");
                printf("<form method=\"POST\"
action=\"http://tzetzal.dcaa.unam.mx/cgi-bin/kiosko_pc/usuario2\">\n");

/* Pasamos nuevamente la clave del usuario en forma "escondida" al CGI
usuario2 */
```

```
                printf("<input type=\"hidden\" name=\"order\"
value=\"%s\">\n",entries[0].val);
                printf("                <input type=\"submit\" value=\"Presione
aqu&iacute; para elegir otro curso\" >\n");
                printf("</form>\n");
        printf("<p>\n");
        printf("<p>\n");
                printf("<img
src=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/line.blue.gif\">\n\n");
        printf("<A
HREF=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/pc/inscripciones.html\"
><h2>Cancelar</h2></A>\n\n\n");
                printf("<p>\n");
                printf("</pre>\n");
        exit(1);
        }
        cont=1;
        band=0;

/* Como hay horarios programados para el curso, debemos validar la fecha
del periodo con la fecha de la consulta en base a la fecha del sistema.
Esto hace dinamico a nuestro programa, puesto que solo
desplegara fechas de periodos validas para inscripciones */

    while (dbnextrow(dbproc) != NO_MORE_ROWS) {
        if(!strcmp(mes_ini,"ENERO"))
          n_mes=1;
        if(!strcmp(mes_ini,"FEBRERO"))
          n_mes=2;
        if(!strcmp(mes_ini,"MARZO"))
          n_mes=3;
        if(!strcmp(mes_ini,"ABRIL"))
          n_mes=4;
        if(!strcmp(mes_ini,"MAYO"))
          n_mes=5;
        if(!strcmp(mes_ini,"JUNIO"))
          n_mes=6;
        if(!strcmp(mes_ini,"JULIO"))
          n_mes=7;
        if(!strcmp(mes_ini,"AGOSTO"))
          n_mes=8;
        if(!strcmp(mes_ini,"SEPTIEMBRE"))
          n_mes=9;
        if(!strcmp(mes_ini,"OCTUBRE"))
          n_mes=10;
```

```
        if(!strcmp(mes_ini,"NOVIEMBRE"))
          n_mes=11;
        if(!strcmp(mes_ini,"DICIEMBRE"))
          n_mes=12;
        if
(((n_mes==punt->tm_mon+1)&&(dia_ini>=punt->tm_mday))||(n_mes>punt-
>tm_mon+1)){
        if (cont == 1){
         band=1;
         printf("<pre>\n");
       printf("<h3>          Los per&iacute;odos y horarios son los
siguientes:</h3>\n");
         printf("<form method=\"POST\" action=\"http://tzetzal.dcaa.unam.mx/cgi-
bin/kiosko_pc/usuario5\">");
         printf("<ul>");
         printf("<input type=\"hidden\" name=\"order\"
value=\"%s\">\n",entries[0].val);
         printf("<input type=\"hidden\" name=\"order\"
value=\"%s\">\n",entries[1].val);

         printf("<input type=\"hidden\" name=\"order\" value=\"%s\">\n",cve_hor);
printf("       <input type=\"radio\" name=\"periodo\" value=\"%s\" checked>
Periodo: %d/%s AL %s/%s \n",cve_per,dia_ini,mes_ini,dia_fin,mes_fin);

printf("       HORARIO: %s - %s HRS.\n",hor_ini,hor_fin);
        }
        else {
         printf("<input type=\"hidden\" name=\"order\" value=\"%s\">\n",cve_hor);
printf("       <input type=\"radio\" name=\"periodo\" value=\"%s\"> Periodo:
%d/%s AL %s/%s\n",cve_per,dia_ini,mes_ini,dia_fin,mes_fin);

printf("       HORARIO: %s - %s HRS.\n",hor_ini,hor_fin);
        }
        cont++;
        }
       }
  }
  if (band==0 )
  {
         printf("<p>\n");
         printf("<p>\n");
         printf("<pre>\n");
         printf("<h3>          No existen horarios programados para este curso</h3>\n");
         printf("<p>\n");
         printf("<form method=\"POST\" action=\"http://tzetzal.dcaa.unam.mx/cgi-
bin/kiosko_pc/usuario2\">\n");
```

```
            printf("<input type=\"hidden\" name=\"order\"
value=\"%s\">\n",entries[0].val);
                    printf("          <input type=\"submit\" value=\"Presione aqu&iacute; para
elegir otro curso\" >\n");
            printf("</form>\n");
            printf("<p>\n");
            printf("<img
src=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/line.blue.gif\">\n\n");
            printf("<A
HREF=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/pc/inscripciones.html\"
><h3>Cancelar</h3></a>\n\n\n");
            printf("<p>\n");
            printf("</pre>\n");
            exit(1);
    }

    printf("\n\n");
    printf("                <input type=\"submit\" value=\"Aceptar\">\n");
    printf("</form><p>\n");
    printf("</ul><img
src=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/line.blue.gif\">\n");
    printf("<p>\n\n");
    printf("<a
href=\"http://tzetzal.dcaa.unam.mx/sistemas/inscripciones/pc/inscripciones.html\"><
h3>Cancelar</h3></a>\n\n");
    printf("<p>\n");
    printf("<p>\n");
    printf("</pre>\n");
    dbexit();
    exit(STDEXIT);

}
```

Apéndice D

Contenido del archivo curso5.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sybfront.h>
#include <sybdb.h>
#include <syberror.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;
   DBPROCESS  *dbproc;
   LOGINREC   *login;
   RETCODE    return_code;
   DBCHAR area[80],clave_cur[4],cuota[10];
   char *autor[10];
   char *nom;

   printf("Content-type: text/html%c%c",10,10);
   printf("<title>Sistema de Inscripci&oacute;n a Cursos</title>\n");

printf("<center>\n");
   printf("<H3>UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
</H3>\n\n\n");
   printf("<h3>Direcci&oacute;n General De Servicios de
C&oacute;mputo Acad&eacute;mico</h3>\n\n");
```

```
  printf("\n\n");
  printf("<h3>Altas cursos</h3>\n");
  printf("<hr>\n\n");
  printf("</center>\n");
  if (dbinit() == FAIL)
     {
      printf("<h1>El servidor esta abajo</h1>\n");
      exit(ERREXIT);
     }
  login = dblogin();
  DBSETLUSER(login,"ricardo");
  DBSETLPWD(login,"ricardo");
  dbproc = dbopen(login,NULL);
  if (dbproc == NULL)
      {
       printf("<h1>El servidor no se encuentra arriba</h1>\n");
       exit(ERREXIT);
      }
  dbfcmd(dbproc,"select area from area");
  if (dbsqlexec(dbproc) == FAIL)
        {
        printf("<h2>Error durante el proceso de acceso al servidor</h2>\n");
        dbexit();
        exit(ERREXIT);
        }
  while(dbresults(dbproc) != NO_MORE_RESULTS){
      dbbind(dbproc,1,STRINGBIND,0,area);
      if (dbrows(dbproc) != SUCCEED )
      {
          printf("<p>\n");
          printf("<p>\n");
          printf("<h3>No existen &aacute;reas de c&oacute;mputo
dadas de alta.</h3><p>\n");
          printf("<i>Seleccione alguna de las siguientes
opciones:</i>\n");
          printf("<ul>\n");
          printf("<p>\n");
          printf("\n\n");
          printf("<li><a
href=\"http://tzetzal.dcaa.unam.mx:87/admin/areaalta.html\"> ALTAS </a>
<p>\n\n");
                printf("<li><a
href=\"http://tzetzal.dcaa.unam.mx:87/admin/admonmenu.html\">Regreso al
men&uacute; principal</a><p>\n");
          printf("<li><a
href=\"http://tzetzal.dcaa.unam.mx:87/admonini.html\">Salir del
```

```
sistema</a><p>\n");
            exit(1);
        }

            printf("<strong>Favor de teclear el nombre del nuevo
curso</strong>\n\n\n");
            printf("<form method=\"POST\"
action=\"http://tzetzal.dcaa.unam.mx:87/cgi-bin/curso1\">\n");
            printf("CLAVE: <input size=4 maxlength=4 name=\"clave\"><p>\n\n");
            printf("CURSO: <input size=50 maxlength=80 name=\"curso\"> <p>\n\n");
        printf("<strong>Seleccione &aacute;rea de c&oacute;mputo:</strong>\n");
        printf("<SELECT name=\"area\">");
            while(dbnextrow(dbproc) != NO_MORE_ROWS){
           printf("<option>%s\n",area);
            }
    }
    printf("</SELECT>\n\n\n");
    dbcanquery(dbproc);
    dbfcmd(dbproc,"select cuota from cuotas");
    if (dbsqlexec(dbproc) == FAIL)
      {
        printf("<h2>Error durante el proceso de acceso al
servidor</h2>\n");
        dbexit();
        exit(ERREXIT);
      }
    while(dbresults(dbproc) != NO_MORE_RESULTS){
        dbbind(dbproc,1,STRINGBIND,0,cuota);

        if (dbrows(dbproc) != SUCCEED )
        {
            printf("<p>\n");
            printf("<p>\n");
            printf("<h3>No existen cuotas de cursos
dadas de alta.</h3><p>\n");
            printf("<i>Seleccione alguna de las siguientes
opciones:</i>\n");
            printf("<ul>\n");
            printf("<p>\n");
            printf("\n\n");
            printf("<li><a
href=\"http://tzetzal.dcaa.unam.mx:87/admin/areaalta.html\"> ALTAS </a>
<p>\n\n");
                printf("<li><a
href=\"http://tzetzal.dcaa.unam.mx:87/admin/admonmenu.html\">Regreso al
men&uacute; principal</a><p>\n");
```

```
            printf("<li><a
href=\"http://tzetzal.dcaa.unam.mx:87/admonini.html\">Salir del
sistema</a><p>\n");
            exit(1);
        }
        printf("<strong>Favor de seleccionar la cuota respectiva: N$</strong>\n");
        printf("<SELECT name=\"cuota\">");
        while(dbnextrow(dbproc) != NO_MORE_ROWS){
          printf("<option>%s\n",cuota);
        }
    }
    printf("</SELECT>\n\n\n");
    printf("<center>\n");
    printf("<input type=\"submit\" value=\"Aceptar\"><p>\n\n");
    printf("<input type=\"reset\"
value=\"Borrar\">\n\n");
    printf("</center><p>\n");
    printf("</form>\n");
    printf("<hr>\n");
    printf("<address>\n");
    printf("<center>\n");
    printf("<a
href=\"http://tzetzal.dcaa.unam.mx:87/admin/admonmenu.html\">Regreso al
men&uacute; principal</a><p>\n");
    printf("<a href=\"http://tzetzal.dcaa.unam.mx:87/admonini.html\">Salir
del Sistema</a><p>\n");
    printf("<hr>\n");
    printf("<a href=\"http://tzetzal.dcaa.unam.mx/~edgar/edpres.html\">
Ing. Edgar Valencia Figueroa </a>\n");
    printf("<p>\n");
    printf("Ricardo Teapila Cuateco<p> \n");
    printf("</address>\n");
    printf("\n");
    printf("<hr>\n");
    printf("\n\n\n");
    dbexit();
    exit(STDEXIT);

}
```

Apéndice E

Salida que genera el CGI curso5

```
Content-type: text/html

<title>Sistema de Inscripci&oacute;n a Cursos</title>
<center>
<H3>UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO </H3>


<h3>Direcci&oacute;n General De Servicios de C&oacute;mputo cad&eacute;mico</h3>



<h3>Altas cursos</h3>
<hr>

</center>
<strong>Favor de teclear el nombre del nuevo
curso</strong>


<form method="POST"
action="http://tzetzal.dcaa.unam.mx:87/cgi-bin/curso1">
CLAVE: <input size=4 maxlength=4 name="clave"><p>

CURSO: <input size=50 maxlength=80 name="curso"> <p>

<strong>Seleccione &aacute;rea de c&oacute;mputo:</strong>
<SELECT name="area"><option>DIFUSION DE LA COMPUTACION
<option>COMPUTO INFANTIL
<option>PROCESADORES DE PALABRA
<option>EDICION POR COMPUTADORA
<option>HOJAS ELECTRONICAS DE CALCULO
<option>MANEJADORES DE BASES DE DATOS
<option>GRAFICACION
<option>INTELIGENCIA ARTIFICIAL
<option>DISE&O ASISTIDO POR COMPUTADORA
<option>ELABORACION DE PROGRAMAS EDUCATIVOS COMPUTARIZADOS
<option>REDES Y TELECOMUNICACIONES
```

```
<option>DESARROLLO DE SISTEMAS DE INFORMACION
<option>LENGUAJES DE PROGRAMACION
<option>MATEMATICAS Y ESTADISTICA ASISTIDA POR COMPUTADORA
<option>ARQUITECTURA DE MAQUINAS Y SISTEMAS OPERATIVOS
<option>SUPERCOMPUTO
</SELECT>


<strong>Favor de seleccionar la cuota respectiva: N$</strong>
<SELECT name="cuota"><option>298.00
<option>688.00
<option>757.00
<option>947.00
<option>884.00
<option>355.00
<option>263.00
<option>522.00
<option>384.00
</SELECT>


<center>
<input type="submit" value="Aceptar"><p>

<input type="reset"
value="Borrar">

</center><p>
</form>
<hr>
<address>
<center>
<a
href="http://tzetzal.dcaa.unam.mx:87/admin/admonmenu.html">Regreso al
men&uacute; principal</a><p>
<a href="http://tzetzal.dcaa.unam.mx:87/admonini.html">Salir del Sistema</a><p>
<hr>
<a href="http://tzetzal.dcaa.unam.mx/~edgar/edpres.html">
Ing. Edgar Valencia Figueroa </a>
<p>
Ricardo Teapila Cuateco<p>
</address>

<hr>
```

Apéndice F
Manual de usuario

El siguiente manual da un bosquejo rápido del manejo del sistema. Se excluyen algunas pantallas por ser de la misma forma para distintos procesos, solo se explican las principales y se evitará redundar.

El primer paso para acceder al sistema de inscripciones a cursos de la DCAA es abrir desde Netscape el siguiente URL:

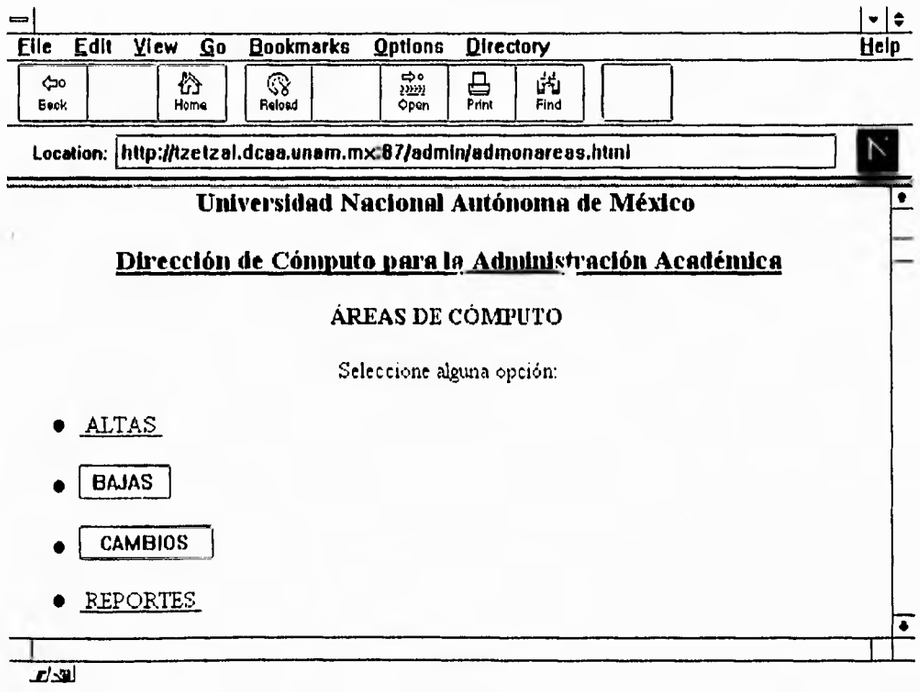
http://tztezal.dcaa.unam.mx:87/admon.ini

en este momento aparecerá el "home page" del sistema que fue mostrado y analizado en el capítulo 6. Al presionar sobre la liga *Entrar al sistema* y validar su entrada como supervisor, se presenta el menú principal. Es importante señalar que si se desea programar un curso y no se han dado de alta los datos para hacerlo, como son las aulas, horarios, cursos, etc., no será posible utilizar esta opción por falta de información, lo mismo sucede si se desea generar constancias o con otro proceso que requiera datos almacenados en las tablas. Con esto nos percatamos que es importante llevar un orden. El primer paso entonces, es dar de alta las áreas de cursos, en el menú principal se selecciona ÁREAS DE CÓMPUTO para entrar a la siguiente pantalla donde se pueden realizar altas, bajas cambios o reportes. En realidad el usuario se dará cuenta que utilizar el sistema no le será nada complicado ya que con los mensajes, botones y cajas de diálogo que se le presentan, es fácil realizar las operaciones que requiere. Por otra parte, existen mensajes preventivos o

de confirmación que le ayudarán a corregir o retroceder la tarea que está haciendo. El sistema se diseñó pensando en usuarios sin conocimientos en el uso de manejadores de bases de datos como Sybase o sistema operativo UNIX.

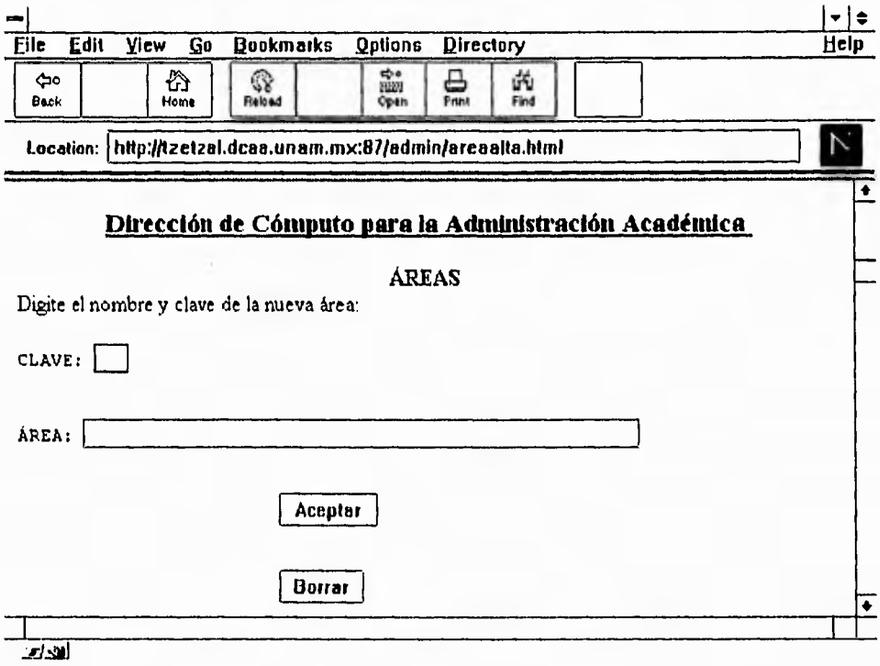
Altas

Para utilizar el sistema, no es necesario que el usuario conozca los comandos para ejecutar inscribir alumnos o programar algún curso, con solo saber entrar al ambiente gráfico Windows y abrir las aplicaciones, será suficiente. En la siguiente figura se muestra la pantalla principal para las áreas de cómputo:



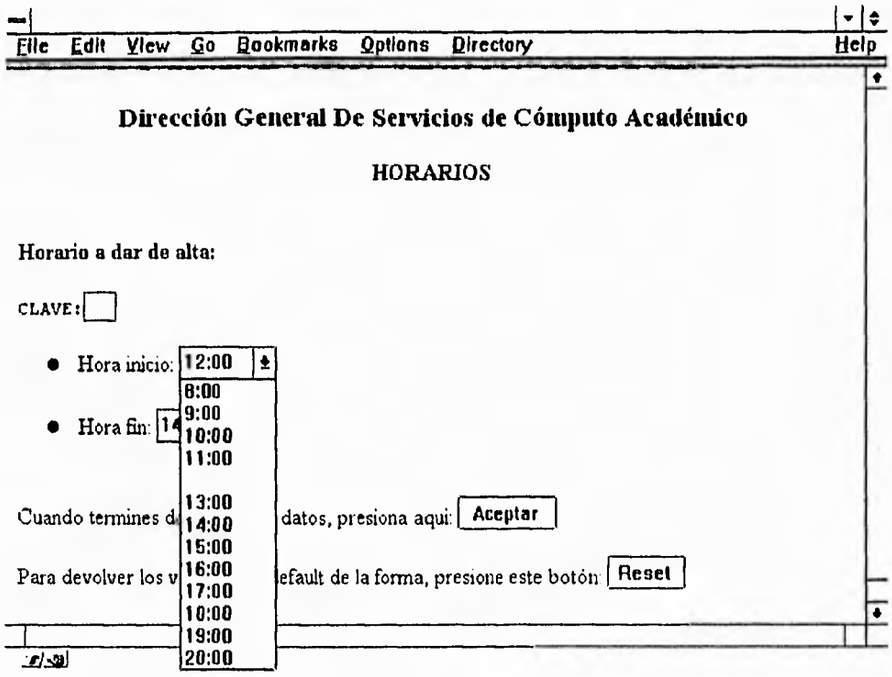
Si se desea dar de alta un área, solo tiene que posicionarse sobre la palabra ALTAS y dar un click con el botón del *ratón*. En la siguiente figura se muestra la forma de captura para dar de alta. En procesos de este tipo, será necesario introducir

una clave, como lo es este caso, pero si ésta ya existiera en la tabla, el sistema lo notificará al usuario. Por recomendación, debe existir una relación entre los nombres y las claves.



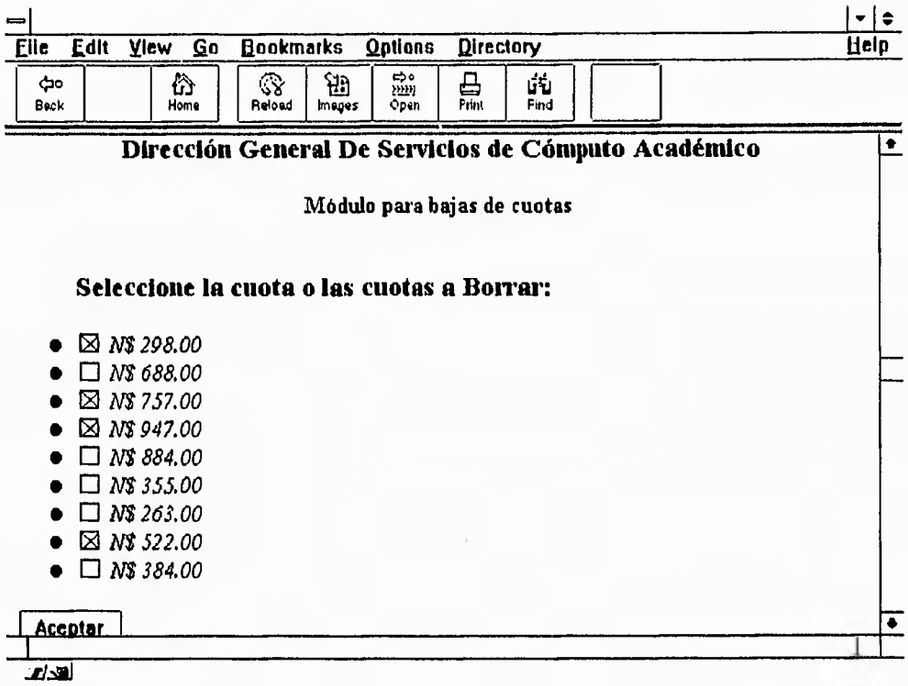
La forma de las pantallas para generar altas, bajas, cambios o reportes es similar para períodos, cursos, aulas, profesores, horarios y áreas, por lo que no es necesario presentar aquí todas esas pantallas ya que tienen la misma estructura.

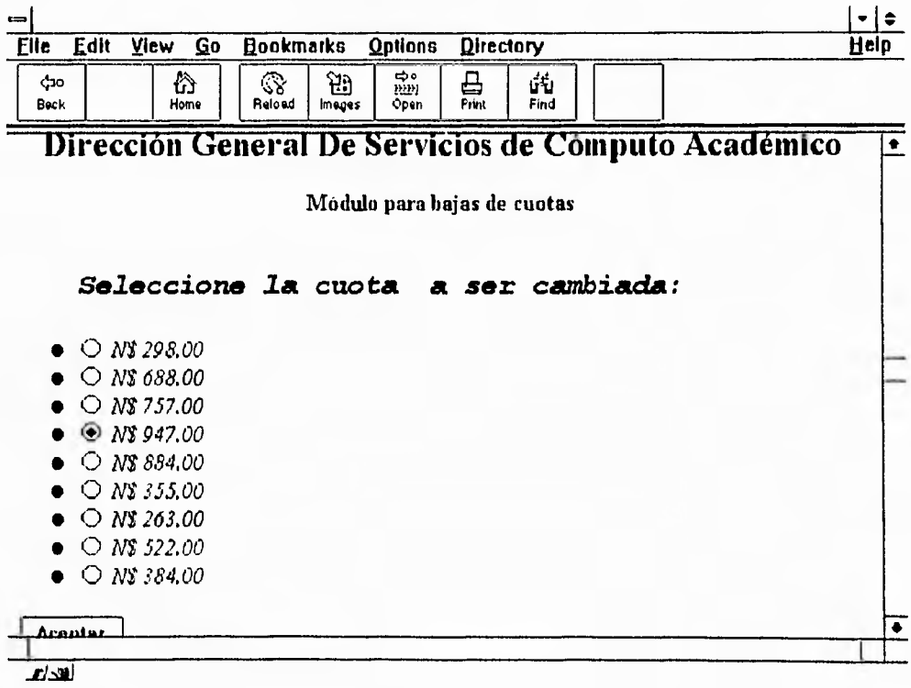
Para dar de alta horarios es mucho mas fácil, puesto que el sistema ya cuenta con datos preestablecidos para la *forma*. En este caso, el usuario solo tendrá que escoger de la lista, el hora de inicio y la hora en que termina. La pantalla se muestra a continuación:



Bajas y cambios

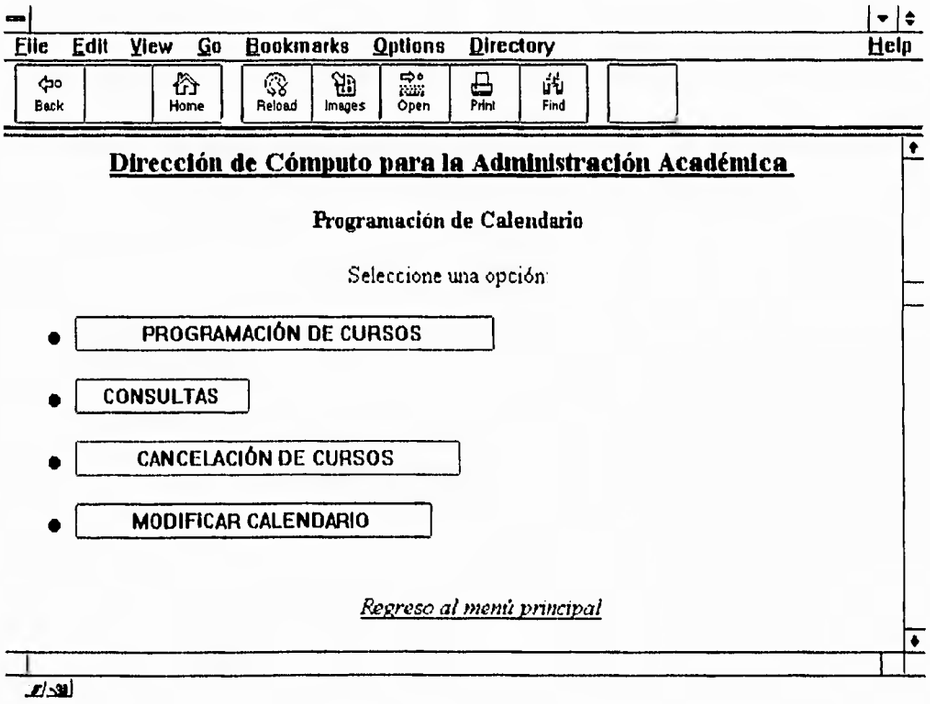
Cuando el usuario desee realizar un cambio o una baja de algún registro en las tablas, verá pantallas que pueden ser superficialmente las mismas, pero si las analizamos nos percataremos de que no es así. Cuando se quiera realizar una baja, el usuario podrá escoger una o más opciones (dependiendo de la cantidad de registros que tenga nuestra tabla), pero, si se encuentra en el módulo de cambios solo se le permitirá seleccionar una de todas las que se muestran. En las siguientes figuras se muestran esas pantallas. En la primera, correspondiente al módulo de bajas, el usuario puede escoger varias opciones, y en la segunda (del módulo de cambios) solo una:


File Edit View Go Bookmarks Options Directory Help
Back Home Reload Images Open Print Find
Dirección General De Servicios de Cómputo Académico
Módulo para bajas de cuotas
Seleccione la cuota o las cuotas a Borrar:
☒ N$ 298.00
☐ N$ 688.00
☒ N$ 757.00
☒ N$ 947.00
☐ N$ 884.00
☐ N$ 355.00
☐ N$ 263.00
☒ N$ 522.00
☐ N$ 384.00
Aceptar


File Edit View Go Bookmarks Options Directory Help
Back Home Reload Images Open Print Find
Dirección General De Servicios de Cómputo Académico
Módulo para bajas de cuotas
Seleccione la cuota a ser cambiada:
○ N$ 298.00
○ N$ 688.00
○ N$ 757.00
◉ N$ 947.00
○ N$ 884.00
○ N$ 355.00
○ N$ 263.00
○ N$ 522.00
○ N$ 384.00
Aceptar

Este tipo de pantallas son las que se muestran durante los procesos de bajas o cambios en general. Como hemos observado hasta este momento, el uso del sistema es realmente sencillo, donde solo debemos cuidar el orden para manipular la información.

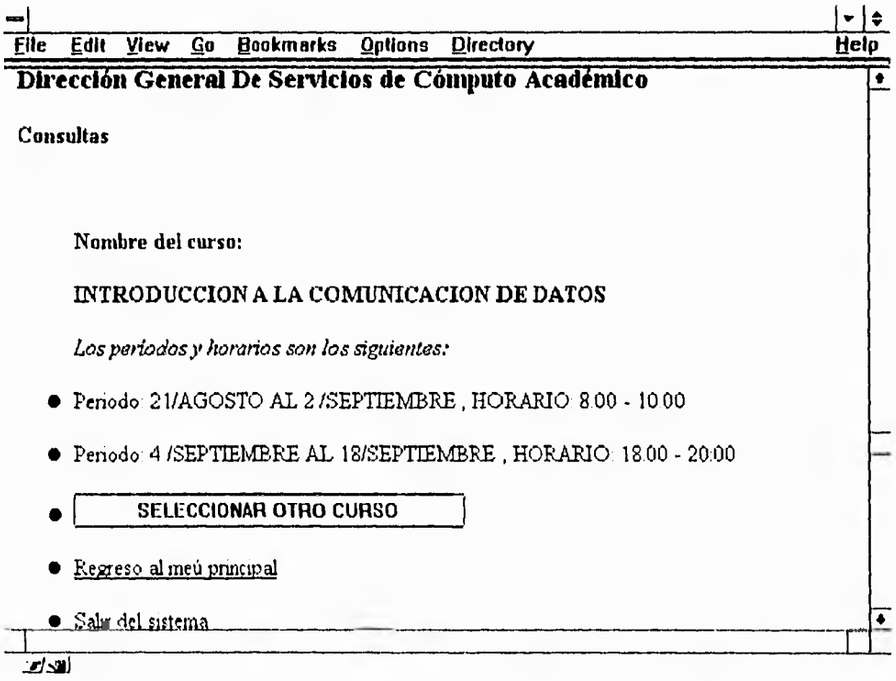
Una vez que los datos se encuentran en tablas, el siguiente paso es programar cursos, para ello es necesario entrar a la opción PROGRAMACIÓN DE CALENDARIO. Las opciones disponibles para el usuario son: programación de cursos, consultas, cancelación y modificación. El menú para la programación de calendario se muestra a continuación.



Al momento de programar los cursos del calendario, la información que aparece en la pantalla es la que se dio de alta en los procesos anteriores, como lo son los horarios, cursos, áreas, aulas, etc. Este proceso en forma completa fue explicado en el capítulo 6 (ver sección 6.4.4).

En la parte de consultas, el usuario puede apreciar la información como si la estuviera viendo en un folleto, donde aparecen los cursos disponibles así como los períodos y horarios sobre los que ya se puede realizar una inscripción. El primer paso para realizar una consulta es seleccionar el área, posteriormente aparecerá una lista de cursos, de éstos se puede escoger uno para ver los diferentes períodos en los que se impartirá.

La siguiente figura nos muestra la pantalla de consultas:



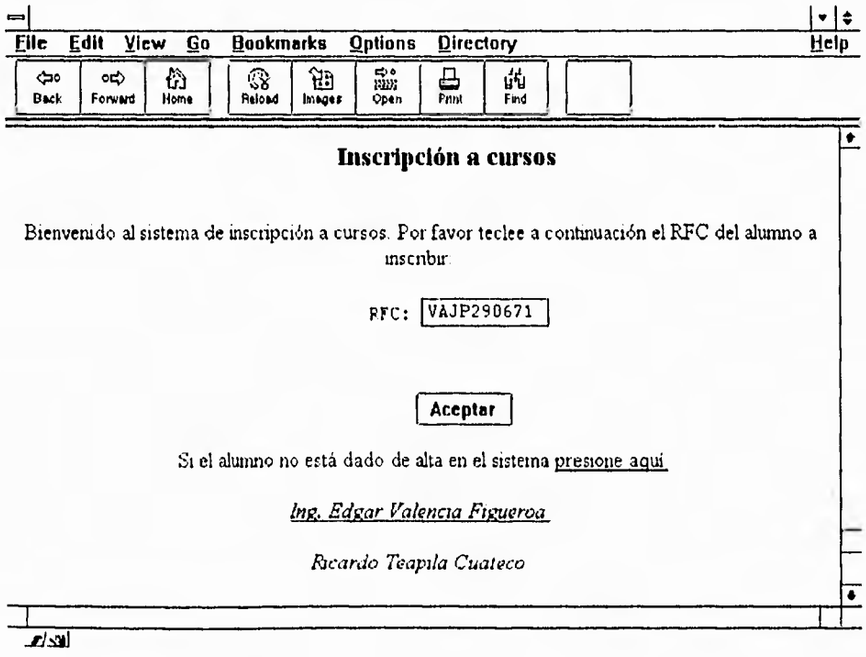
Este tipo de pantallas son las resultantes de unir toda la información proporcionada por el usuario administrador en procesos anteriores.

Para el módulo de cancelación de cursos es necesario al igual que en la programación, seleccionar un área y posteriormente el curso sobre el cual se quiere realizar la operación. Este proceso es muy semejante para la modificación el

calendario, y una vez que se ha llegado a la pantalla donde se muestra la información del período del curso, es posible realizar los cambios necesarios, como pueden ser: cambiar el área a la que pertenece, el horario, etc.

Inscripciones

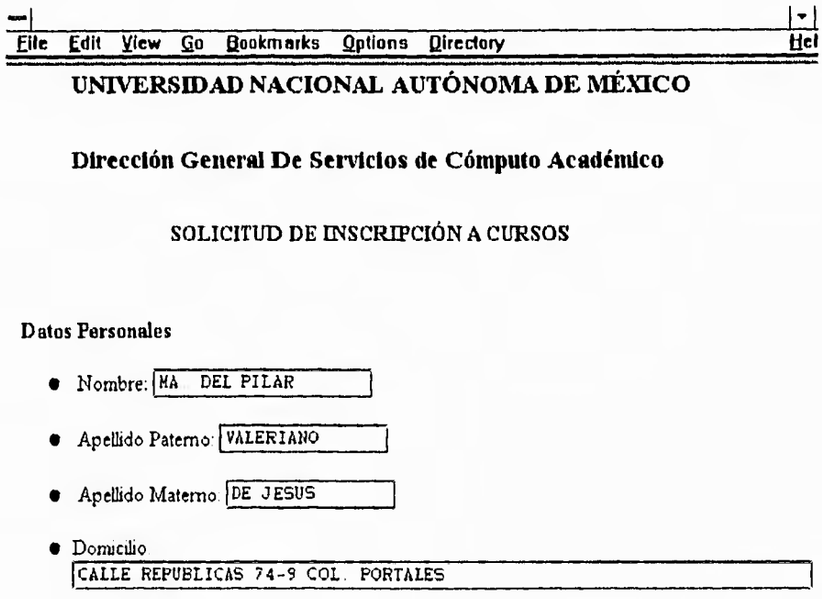
Desde el menú principal hay que acceder al proceso de inscripciones dando un click sobre la palabra que lo indica; en este momento aparecerá una pantalla donde se le pedirá al usuario introducir su RFC, mediante esta información el sistema obtendrá la información necesaria para realizar la inscripción. Si el usuario es nuevo, no existirá problema alguno ya que los mensajes le indicarán que presione sobre ciertas áreas para poder darlo de alta.



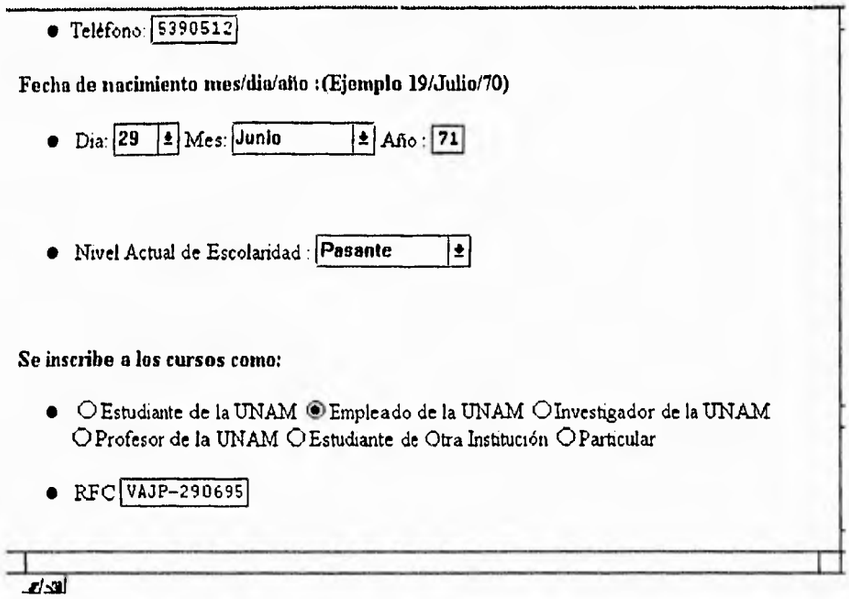
En la figura anterior se muestra la pantalla para introducir el RFC del usuario.

Si el administrador por alguna causa no se percato del mensaje del sistema que le indica presione la liga en el caso de que el usuario no esté dado de alta, y

presiona el botón de aceptar, el sistema automáticamente buscará la información en la tabla y detectará que el RFC es nuevo por lo que le enviará un segundo mensaje par darse de alta. En este momento le pedirá que introduzca sus datos personales y con ello se le otorgará el derecho a inscripción.



Cuando los datos han sido dados de alta y se ha presionado el botón de Aceptar, se muestran los datos completos que ya fueron registrados. Luego, aparece una pantalla para inscribirse a algún curso y continuaremos con el proceso hasta escoger el curso, período y horario deseados. En ese momento, el sistema automáticamente revisará la seriación, para ver si el alumno tiene los conocimientos requeridos, es decir, que haya aprobado los cursos antecedentes. Si la inscripción es rechazada, se le da la oportunidad de seleccionar otro curso, pero si éste es el mismo caso que el anterior, el sistema seguirá rechazando la inscripción. La siguiente figura muestra la continuación de la forma de solicitud de inscripción.

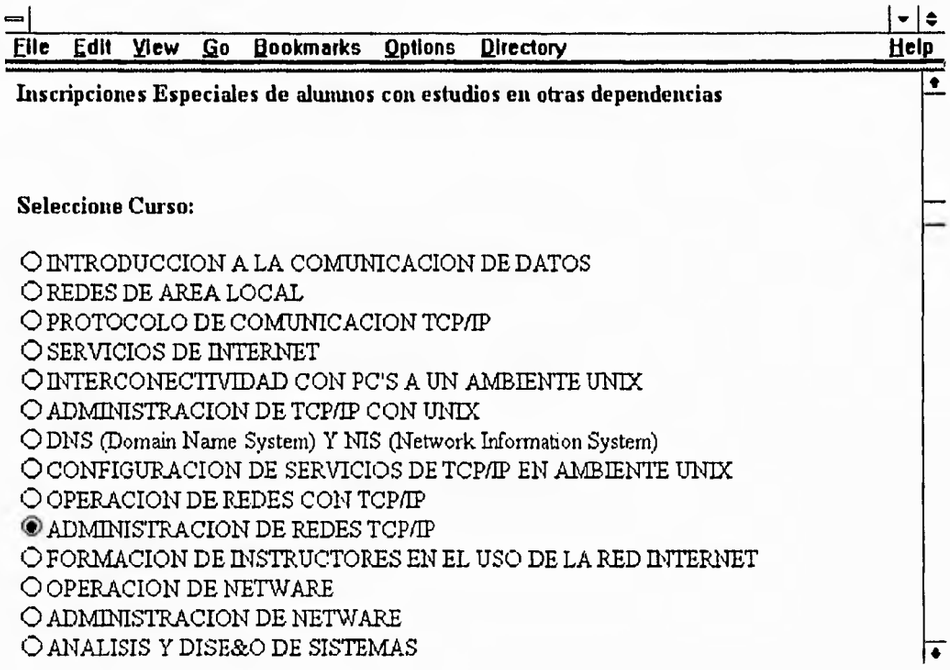


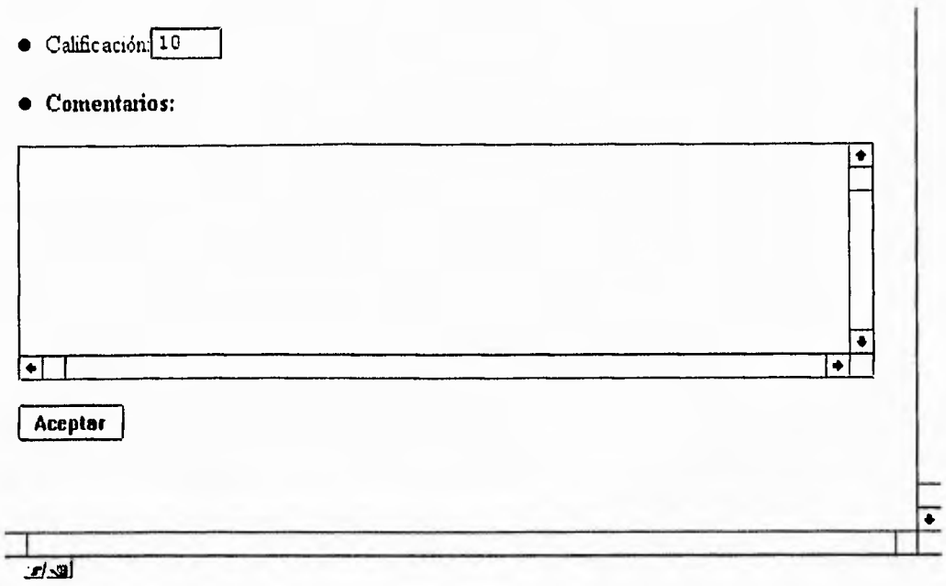
Algo muy importante que hay que recalcar, es que si el alumno ha cursado los antecedentes en otra institución, deberá ser inscrito mediante otro proceso. La opción para inscripciones de alumnos externos se encuentra listada en el menú principal.

Cuando la inscripción ha sido aceptada por tener el requisito de antecedente cubierto, estará dado de alta en el sistema para tomar el curso, pero sólo *temporalmente*, ya que no estará dado de alta formalmente hasta que realice el pago correspondiente al curso. Toda esta información se maneja mediante dos tablas, una con información temporal y la otra con la información definitiva. Esto es para evitar que un alumno inicie el proceso de inscripción y no lo termine, además de ocupar un lugar que puede ser de interés para otra persona.

En el proceso de inscripción a cursos de usuarios externos, el sistema es accesible, ya que recibe solo el RFC del alumno para verificar o dar de alta y se llena la forma respectiva para el antecedente, donde se le pide se le solicita escoger

el curso y la calificación que obtuvo en dicho curso. Con esto podrá pasar al proceso de inscripción normal para verificar la seriación, con lo que quedará inscrito y dado de alta en el sistema. La información capturada en este proceso se almacena en otra tabla, puesto que únicamente se tiene como antecedente, más no se podrán generar constancias a partir de esta información.





Dentro del menú de inscripciones existen otras tres opciones:

- CANCELAR INSCRIPCIONES
- BAJAS DE ALUMNOS
- CONSULTAS

donde el usuario puede llevar un mejor control en el manejo de la información.

Calificaciones

Una vez que ha terminado el curso y el profesor reporta las calificaciones de los alumnos, el paso siguiente es capturarlas. Este es un proceso sencillo, al acceder al menú de calificaciones se encuentran las opciones de Altas, Bajas, Cambios y Reportes. Si lo que el usuario desea es dar registrar las calificaciones que los profesores le entregaron entonces tiene que elegir la opción de Altas. Como en la mayor parte del sistema, primero hay que escoger el área, después el curso y el

período donde se desean dar de alta las calificaciones. El sistema genera la lista de alumnos inscritos en ese curso a partir de la información que se encuentra en la Base de Datos, por lo que el administrador podrá llenar las calificaciones de ellos y quedar listos para la generación de constancias. Si por algún motivo el usuario escogió un curso o período equivocado, el sistema le informará que no hay alumnos inscritos en ese curso.

En los procesos de bajas, cambios o reportes de calificaciones es igual al similar a los procesos anteriormente descritos donde se tiene acceso a estas opciones.

Generación de Constancias

Este es el último proceso manipulado por el usuario administrador, en él se tiene un recuento de la información de alumnos, cursos, períodos, horarios, etc. Como es de suponerse, el proceso de generación de constancias era anteriormente cansado y tedioso, puesto que se tenía que convertir y transferir la información de un equipo a otro. Ahora, con este proceso automatizado es posible generar constancias desde el mismo sistema, donde se tiene la información catalogada y almacenada lista para emitir esta clase de reportes.

Este proceso es muy similar a los anteriores donde hay que agrupar la información. El primer paso entonces es seleccionar un área, un curso, un período y horario, como toda la información se encuentra almacenada, solo hay que indicar al sistema mediante un click que la imprima. En este caso, la generación de constancias se hace con ayuda de impresoras láser de alto volumen conectadas en red. En el capítulo 3, sección 3.1.2 hablamos un poco acerca de estas impresoras. Las constancias llevan un formato, el cual será el mismo para todas, por lo que resta llenarlas al momento de enviar el trabajo de impresión. El formato puede ser

generado desde algún paquete de Windows (por ejemplo: Word, PowerPoint, etc.) como una imagen en formato PostScript. Esta imagen se almacena en el servidor de impresión y al enviar el trabajo, se le indica al sistema (XPM) que el archivo siendo tipo texto (que contiene los datos de los alumnos con sus respectivas calificaciones) estará asociado a la imagen que generamos previamente. El resultado de este proceso, son las constancias de los alumnos, las cuales se generan en algunos minutos.

Como se pudo observar, todos los procesos del sistema son muy fáciles de manipular, las ventanas, menús, botones, mensajes, cajas de dialogo, etc., ayudan en todo momento al usuario permitiéndole manipular una gran cantidad de información, donde el usuario nunca se entera, por ejemplo, de que tipo son los datos, de que tamaño son, etc., y sobre todo dónde están ubicados. Este sistema por lo tanto, creemos que ha cumplido con los objetivos planteados, y ha resuelto los problemas a los que nos enfrentamos en las tareas de inscripciones.

GLOSARIO

ANSI

American National Standards Institute. Grupo que define los estándares de EU para la industria del procesamiento de la información. ANSI participa en la definición de estándares para protocolos de redes.

API

Aplication Programming Interface. Es una interfaz que permite a los programadores hacer llamadas a funciones de un lenguaje de programación.

Aplicación

Programa diseñado para realizar una tarea específica.

Archie

Es un método para búsqueda de archivos en servidores FTP anónimos.

Atributo

Es una propiedad descriptiva de una entidad.

Back end

Es el software de un servidor, capaz de resolver las peticiones de los clientes.

CERN

Laboratorio Europeo de Física de Partículas de Suiza. Los inventores de los conceptos de HTTP y HTML.

CGI

Un CGI (Common Gateway Interface) es un programa ejecutable que sirve como interfaz entre visualizadores de la World Wide Web y servidores HTTP.

Cliente

Programa de computadora que utiliza los recursos dados por un servidor a través de la red. NCSA Mosaic es un ejemplo de un cliente de la World Wide Web.

CSMA/CD

Carrier Sense Multiple Access with Collition Detection. Es una característica del hardware de red que utiliza acceso CSMA combinado con un mecanismo que le permite al hardware detectar cuando dos estaciones están intentando transmitir información simultáneamente. Un ejemplo es Ethernet.

Dato

Unidad mínima de información que es susceptible de una observación sin significado.

DBMS

(DataBase Management System). Sistema Manejador de Bases de Datos que está formado por un conjunto de rutinas de software interrelacionadas donde cada una de éstas, es responsable de una tarea determinada.

Deamon

Programa residente en memoria que ejecuta algún servicio o acción muy específica y que sólo es aplicable a los sistemas operativos UNIX.

Diagrama Entidad-Relación

Es una herramienta de modelado que tiene el propósito de representar las entidades de datos con sus relaciones. Las entidades se representan con un rectángulo etiquetado y las relaciones mediante líneas que interconectan a las entidades relacionadas.

Diagrama de Flujo de Datos

Es una herramienta empleada en la elaboración de modelos que muestra un sistema como una red de procesos conectados entre sí por flujos.

Dominio

Parte de la jerarquía de nombres en sistemas. Sintácticamente, un dominio consiste en una secuencia de nombres u otras palabras separadas por puntos.

Entidad

Son todos aquellos organismos, instituciones, dependencias, personas, etc., que están relacionados con el sistema de información.

Escalabilidad

Característica de hardware o software que permite crecer desde un sistema básico hasta uno más poderoso.

Ethernet

Popular tecnología de redes locales desarrollada por Xerox. Ethernet es la mejor implantación que utiliza teconología de CSMA/CD.

External Viewer

Es un programa externo utilizado por Mosaic cuando no puede manejar un tipo particular de archivo internamente.

FDDI

Fiber Distribution Data Interfase. Estándar reciente de comunicaciones basado en fibra óptica que ha sido establecido por ANSI. FDDI especifica una velocidad de señalización de 100 Mbps y una longitud máxima de 200 kms.

Front End

Término que se refiere a la parte de los clientes que interactúa directamente con el usuario.

FTP

Protocolo de transferencia de archivos (File Transfer Protocol), es un método de transferencia de archivos de y para computadoras en sitios remotos.

GIF

Formato de intercambio de gráficas (Graphics Interchange Format), es un formato para archivos de imágenes.

GUI

Una Interfaz de usuario gráfica (Graphical User Interface) es un programa que permite al usuario interactuar con un sistema mediante la visualización de íconos, imágenes, gráficas, texto, etc.

Gopher

Es un sistema de información distribuido de texto desarrollado en la Universidad de Minnesota

Hardware

Elementos físicos que integran un sistema electrónico.

Hiperliga

Es una liga dada a un documento para poderse conectar con otro documento. Estas ligas generalmente son hechas en hipertexto y están representadas por palabras subrayadas y de otro color o imágenes.

Home Page

Es un documento en HTML que se despliega cuando se inicializa una sesión con Mosaic, es la presentación de inicio de sesión. Sin embargo, se puede diseñar el "home page" que el usuario desee.

Host

Cualquier computadora capaz de ejecutar programas de aplicación y que se encuentra conectada a una red.

HTML

Lenguaje de hipertexto (HyperText Markup Language). Son una serie de reglas que gobiernan la manera que creamos documentos y que pueden ser leídos por un paginador de la World Wide Web. La mayoría de los documentos que son desplegados por Mosaic son del tipo HTML, estos documentos están caracterizados por su extensión .HTML o .HTM. Por ejemplo: homepage.html o homepage.htm.

HTTP

Protocolo de transferencia de hipertexto (HyperText Transfer Protocol), es el protocolo utilizado por los servidores de la Wide World Web.

Internet

La red de computadoras más grande del mundo, conocida también como la red de redes, que conecta a instituciones del gobierno, académicas y comerciales.

Interoperabilidad

Forma de comunicar diferentes arquitecturas de cómputo así como programas de software.

JPEG

Grupo Experto Fotográfico de Unión, por sus siglas en inglés Joint Photographic Expert Group, es un método de almacenamiento de imágenes en un formato digital.

LAN

Local Area Network. Cualquier tecnología física de redes que opera a altas velocidades sobre distancias cortas.

Mainframe

Término aplicado a las computadoras de alto rendimiento con características de un alto nivel transaccional, con gran capacidad para atender a varios usuarios y con requerimientos especiales en cuanto a condiciones ambientales para su funcionamiento.

MPEG

Grupo experto de fotografías en movimiento (Moving Pictures Experts Group), es un método para almacenar archivos de video en un formato digital.

Multimedia

Aplicación que integra video, texto, imágenes, sonidos y gráficas.

NCSA

Es el Centro Nacional de Aplicaciones de Supercómputo (The National Center for Supercomputing Applications). NCSA se encuentra localizado en la Universidad de Illinois en Urbana-Champaign, Estados Unidos.

OSI

Open System Interconnection. Estándar de arquitectura de redes propuesto por ISO, consiste de siete capas.

PostScript

Es un lenguaje de descripción de páginas, desarrollado por la Compañia Adobe Systems. Es un lenguaje manejado por algunas impresoras láser.

Protocolo

Conjunto de convenciones y normas para establecer un diálogo a distintos niveles de implantación de una arquitectura de redes.

Prototipo

Es un modelo del sistema que permite conocer las necesidades del usuario, plasmándolas en interfaces del sistema.

RedUnam

Proyecto de interconexión de cómputo en la comunidad universitaria.

Relación

Es el vínculo entre entidades.

RISC

Reduced Instruction Set Computer. Arquitectura de procesadores que implantan un conjunto de instrucciones para obtener un mejor desempeño en el equipo de cómputo. Concepto que se utiliza en estaciones de trabajo.

Servidor

Es un programa que proporciona atención a los clientes que lo soliciten a través de la red Internet

SGML

Lenguaje de Marcado Estándar Generalizado (Generalizad Standard Generalized Markup Language), es un estándar internacional, un esquema codificado para la creación de información textual, HTML es un subconjunto de SGML.

Software

Conjunto de programas y aplicaciones que forman parte de una computadora.

SQL

Es un lenguaje de consulta estructurado (Structured Query Languaje), que permite al usuario formular operaciones relacionales en una base de datos.

Tabla

Es una estructura de datos que posee un encabezado y un cuerpo, el encabezado es un tipo determinado de columnas y tipos de datos, el cuerpo es un conjunto de renglones de valor cambiante y cuyo tipo es el definido para cada columna en el encabezado.

TCP/IP

Protocolo de Control de Transmisión/Protocolo de Internet (Transmission Control Protocol/Internet Protocol), es una serie de reglas que establecen el método con el cual los datos van a ser transmitidos a través de la red Internet.

TELEPAC

Red pública de datos basada en el protocolo X.25.

TELNET

Es el protocolo estándar de la Internet del servicio de conexión remota de terminales. Telnet permite a un usuario en una máquina interactuar con un sistema remoto de tiempo compartido en algún otro lugar como si la terminal del usuario estuviera conectada directamente al sistema remoto.

Token Ring

Un tipo de tecnología de red que controla el acceso al medio de transmisión a través de un paquete llamado token. Una computadora puede transmitir un paquete cuando posee el token.

Topología

Es la distribución geométrica y física de una red de cómputo.

Unix

Sistema Operativo interactivo de tiempo compartido desarrollado en 1969 por Ken Thompson, después de que los laboratorios Bell abandonaran el proyecto Multics. Unix existe en varias plataformas de cómputo.

URL

Localizador Uniforme de Recursos (Uniform Resource Locator), es la dirección de alguna fuente de información. El URL contiene cuatro partes, el tipo de protocolo, el nombre de la máquina, el directorio y el nombre del archivo. Por ejemplo:

http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/NCSAMosaicHome.html

Workstation

Estación de trabajo. Cualquier combinación de dispositivos de cómputo que provee al usuario de un gran poder de procesamiento y velocidad y un medio de almacenamiento permanente y de rápido acceso. Generalmente conectado a una red y con un sistema operativo poderoso.

World Wide Web = WWW = W3 = The Web

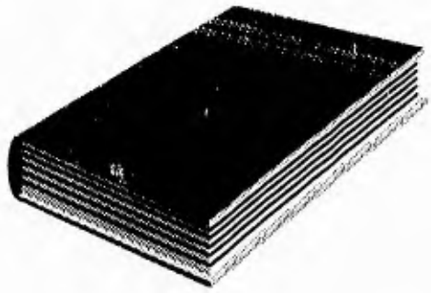
Es un sistema de información distribuida en hipermedia desarrollado por el CERN para proveer a la comunidad de Internet una manera sencilla de tener acceso a la información.

x.25

Es un estándar internacional del CCITT para el envío de paquetes a través de redes públicas de datos.

X11

Conjunto estandarizado para el manejo gráfico de pantallas, aplicaciones, etc. que son totalmente independientes del hardware utilizado para dicho propósito. Fue desarrollado por el MIT y pertenece al dominio público.



Bibliografía

- A guide to SYBASE and SQL Server

  Mc. Goveran Dale

  Ed. Addison Wesley

  1992. Estados Unidos

- The World Wide Web

  John December, Neil Randall

  Ed. Sams Publishing

  1994. Estados Unidos

- The Mosaic Handbook for the X Window System

  Dale Doughertg, Richard Koman & Paula Ferguson

  Ed. O'Reilly & Associates, Inc.

  1994. Estados Unidos

- Internetworking with TCP/IP Vol. I

  Douglas E. Comer, David L. Stevens

  Ed. Prentice Hall

  1991. Estados Unidos

- The Internet Book

  Douglas E. Comer

  Ed. Prentice Hall

  1995. Estados Unidos

- Redes de Ordenadores

  Andrew S. Tanenbaum

  Ed. Prentice Hall

  2da. Edición. 1989. Estados Unidos

- Distributed Computer Environments

  Daniel Cerutti, Dona Rierson

  Ed. Mc. Graw Hill

  1993. Estados Unidos

- The Multimedia Library

  James Cabecerras

  Ed. Academic Press

  1990. Londres, Inglaterra

- Conéctate al Mundo de Internet

  Ed Krol

  Ed. McGraw-Hill O'Reilly & Associates Inc.

  2da. Edición. 1995. México, D.F.

- El Lenguaje de Programación C

  Brian W. Kernighan, Dennis M. Ritchie

  Ed. Prentice Hall

  2da. Edición. 1992. México, D.F.

- C Manual de Referencia

  Herbert Schildt

  Ed. McGraw-Hill

  2da. Edición. 1991. Madrid, España

- Multimedia Technology Combining Sound, Text, Computing, Graphics and Video

  Bohdan O. Szuprowicz

  Ed. Computer Technology Research Corp.

  1992. Estados Unidos

- Revista "ELECTRONIC PRODUCTS"

  Artículo:

  "Touchscreens are changing the face of computers"

  Autor: Ben Phillipps

  Noviembre de 1994. Estados Unidos

- Revista "BYTE" (A McGraw-Hill Publication)

  Artículo:

  "The 12 Top GUI's"

  Autores: Frank Hayes y Nick Baran

  Julio de 1989. Estados Unidos

- Revista "Ciencia y Desarrollo"

  Artículo:

  "La era digital"

  Autores: Silvia Castillo A., Humberto Hernández R. y Raúl Rojas G.

  Septiembre-Octubre de 1990, México.

- Revista "Soluciones Avanzadas"

  Artículos:

  "Sistemas de bases de datos"

  "Interfaces de programación en red: API's, Sockets, RPC's, y esas cosas..."

  "Algunos protocolos de aplicación en redes TCP/IP..."

  "Redes para uso académico: BITNET e Internet"

  "Introducción a redes de computadoras"

  "X-Window y los sistemas de ventanas"

  Enero-Febrero de 1993, México

- Revista "Soluciones Avanzadas"

  Artículos:

  "Conceptos generales sobre redes locales"

  "Redes locales tipo Ethernet"

  "Redes de área local tipo Token Ring"

  "Introducción a la realización de programas en X-window"

  Marzo-Abril 1993. México

- Revista "Personal Computing"

  Artículo:

  "Multimedia en México. ¿Para quién?"

  Mayo de 1994, México

- Manuales de SYBASE

  - Advanced System Administration Students Guide

  - Transact SQL

  - Open Client DB-Library Student's Guide

- Manuales de SUN

  - Solaris 2.2 User's Guide

  - Sun Video User's Guide

- HyperText Markup Language Specification 2.0. Internet Draft

  Tim Berners-Lee, CERN

  Noviembre 28, de 1994. Suiza

- HyperText Transfer Protocol Specification. Internet Draft

  Tim Berners-Lee, CERN

  Noviembre 5, de 1994. Suiza

- Uniform Resource Locator Specification. Internet Draft

  Tim Berners-Lee, CERN

  Noviembre 5, de 1994. Suiza