12

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES.

CAMPUS ARAGÓN

SISTEMA DE ADMINISTRACIÓN DE CUENTAS DE CORREO ELECTRÓNICO DE REDUNAM.

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

P R E S E N T A N:

HUMBERTO CALIFORNIA GONZÁLEZ
ALEJANDRO GUÍZAR DEL RAZO.

ASESOR:
ING. JOSÉ LUIS LEGORRETA GARCÍA

MÉXICO

NOVIEMBRE 2001

AGREDECIMIENTOS

A mis padres:

Por todo el apoyo que me han dado siempre para cumplir con mis metas. Gracias por esas atenciones, sacrificios y esfuerzos que han hecho para ayudarme a terminar una carrera universitaria. Este trabajo de Tesis fue realizado especialmente para ustedes, ya que es la culminación de la herencia que ustedes me han dado.

A mis hermanas Sandra y Rocío:

Por darme esa motivación para que terminara con este trabajo de Tesis. Por transmitirme su entusiasmo para seguir adelante. Por todas las cosas que hemos compartido. Cuenten con mi apoyo siempre. Muchas gracias hermanas.

A mi novia:

Gracias Rocío, por la atención que has tenido para que yo concluya con este trabajo de Tesis. Por tu paciencia y tu tiempo que has cedido para la realización del mismo. Te estaré siempre agradecido mi amor.

A Alejandro Guízar:

Por ser un gran amigo. Por todos los momentos que hemos convivido juntos desde la preparatoria. Finalmente concluimos con este proyecto que como verás nos costo mucho trabajo terminarlo. Gracias a ti y a tu familia por esa amistad y apoyo incondicional.

A mi Asesor:

Ing. José Luis, le agradezco su paciencia y dedicación que tuvo para dirigir esta Tesis. Gracias por el tiempo invertido.

A mis amigos de siempre (René, Marco A. Ziga , Nora, Alejandro Espejel ("James"), Marco Antonio Mejía, Olga, Salomón) :

Por los grandes momentos que hemos pasado juntos. Como olvidar el gran grupo que con algunos de ustedes hemos formado desde la preparatoria. Algunos otros se integraron en la universidad, pero no por eso dejan de ser mis grandes amigos. Gracias por sus consejos y voluntad transmitida para dar el último paso y terminar esta tesis.

En general, gracias a todas las personas que contribuyeron para la realización de este trabajo. Es difícil escribir todos sus nombres aquí, pero cuando lean esto, sabrán que tiene todo nuestro agradecimiento.

Humberto California González

AGREDECIMIENTOS

A mis padres:

Que siempre me han mostrado su apoyo y su preocupación hacia mí , les dedico este proyecto de Tesis, el cual no es un logro mío sino también de ustedes y del cual les estaré por siempre agradecido, ya que sin esa motivación que siempre me brindaron, mi camino a esta meta hubiera sido mas difícil . Se que este es apenas el comienzo de esta carrera , que la ruta para descubrir mis objetivos la estoy iniciando pero se que siempre contaré con ustedes y lo mas importante, ustedes siempre contarán conmigo , lo cual quiero que sea algo que lo tengan presente, siempre en mente. Si alguna vez mi carácter fue o es un poco fuerte, espero me comprendan. Por todo esto y por todo lo que representan para mi, les agradezco este esfuerzo que han hecho para que yo cumpla estos logros.

A mi hermano Luis:

Por apoyarme en muchos momentos durante mi carrera , por ser mi amigo y por sacarme de dudas muchas veces , te agradezco tu paciencia de tenerme como hermano y sabes que siempre contarás conmigo así como yo espero siempre contar contigo. Gracias Luis.

***A mis abuelos**:*

Por estar conmigo desde mi niñez y por darme ese cariño que solo ustedes saben dar y que siempre lo he tenido en mente, sinceramente quiero que sepan que cuenten con mi cariño y aprecio hacia ustedes por estar conmigo y por los consejos que me han dado y que en verdad los tomé en cuenta. Por siempre agradecido , su nieto Alejandro.

***A mis tíos**:*

Por la ayuda que me brindaron de todo tipo durante mi etapa académica , les quiero agradecer por todo esto tiempo y su atención que tuvieron hacia mi. Cuenten siempre con mi apoyo.

A Humberto:

Amigo, ya me conoces de años y creo que yo a ti también y quiero agradecerte todos estos años tu ayuda y apoyo que siempre me brindaste, durante toda la preparatoria, carrera y ahora en la tesis. Espero sigamos por mucho tiempo ayudándonos , ahora no solo en el aspecto académico sino también como amigos que siempre hemos sido. A ti y a tu apreciable familia les agradezco su amistad y apoyo.

A mi Asesor:

Ing. José Luis, le agradezco su paciencia y dedicación que tuvo para esta Tesis , que después de tantos meses y contratiempos, por fin esta terminada. Espero que sigamos en contacto.

A mis amigos de siempre: (Diana, René, Marco A. Mejía, Marco A. Ziga , Nora, Alejandro ("James")) :

Amigos de siempre y no solo por el tiempo que hemos convivido juntos durante todos estos años, con unos más o con otros menos , sino por el tiempo que se que seguiremos en contacto que espero sea mucho o por lo menos eso intentaré. Gracias por su apoyo en todo este tiempo.

Alejandro Guízar del Razo

SISTEMA DE ADMINISTRACIÓN DE CUENTAS DE CORREO ELECTRÓNICO DE REDUNAM

INDICE

Capítulo 4 - La comunicación entre la interfaz gráfica y el servidor (socket)

INTRODUCCIÓN

En la actualidad, el hombre está tratando de realizar de manera automática trabajos que ha estado haciendo de forma rutinaria. Principalmente en el área de la computación y las comunicaciones siempre están surgiendo cosas nuevas, por lo que se requiere tener procesos automáticos para realizar determinadas tareas, o bien que faciliten el trabajo y disminuyan el tiempo que se invierte para que esa tarea se lleve a cabo, esto puede ser mediante un sistema que le permita a cualquier persona realizar esa tarea sin que tenga la necesidad de conocer o seguir todo un proceso paso a paso. De esta manera solo tiene que ejecutar ciertas acciones y el sistema realizará el proceso internamente y proporcionará un resultado.

Con base a esto, la persona que realizaba todos los procesos para cumplir con una tarea, reducirá el número de los mismos y podrá aprovechar el tiempo restante aprendiendo o haciendo otras cosas.

La automatización es un término, que últimamente es muy usado, del cual el hombre ha aprendido a sacar ventajas, para realizar su trabajo de manera más rápida y eficiente.

Precisamente, esta es la palabra clave, para que la labor de un administrador de un sistema de correo electrónico que maneja más de 21000 buzones de usuario, sea más enfocada a la Administración del Sistema Operativo del servidor y no propiamente a la administración de las claves de correo electrónico, que con un sistema bien diseñado e implementado, puede hacer que otras personas le ayuden a administrar esas cuentas.

Es aquí, donde surge la necesidad en el Departamento de Administración de Servidores de la D.G.S.C.A. para desarrollar un proyecto que permita a otras personas auxiliarle al administrador del servidor central de correo electrónico (servidor.unam.mx) a administrar dichas cuentas.

Debido a esto, presentamos aquí un trabajo que en 5 capítulos, resuelve las necesidades de la institución.

Capítulo 1. Antecedentes Históricos. Como todo proyecto requiere de antecedentes, aquí tratamos de mostrar como ha ido evolucionando toda la infraestructura de lo que hoy es RedUNAM.

Capítulo 2. La Base de Datos. En este proyecto presentamos, de manera detallada como fue el análisis y diseño estructurado, así como el modelo Entidad Relación de la base de datos. En este proyecto se trabajará con el manejador de Base de Datos Sybase.

Capítulo 3. La Interfaz Gráfica. Aquí se mostrarán todos los elementos que son utilizados para desarrollar un entorno gráfico que permita a un usuario final visualizar y llevar de forma amigable la administración de cuentas de un servidor de correo electrónico.

Capítulo 4. La comunicación entre la interfaz gráfica y el servidor (socket). En este apartado se mostrará el medio por el cual se van a comunicar el servidor de base de datos, el servidor de Web y el servidor de correo electrónico.

Capítulo 5. La integración de los componentes del sistema. Es aquí donde se mostrará uno a uno la configuración de los componentes de este sistema, así como la manera en que se integran.

Con estos capítulos pretendemos cubrir los objetivos y alcances que demanda el presente trabajo.

OBJETIVOS

- Analizar la demanda de cuentas para obtener el servicio de correo electrónico en REDUNAM , y el procedimiento que se debe para dar mantenimiento a estos procesos de administración de las cuentas,

- Identificar las funciones y requerimientos necesarios para llevar a cabo una eficiente administración.

- Desarrollar un sistema que permita administrar las cuentas de correo electrónico de REDUNAM al personal del Área de Atención a Usuarios de D.G.S.C.A. a través de una interfaz gráfica por medio del WWW.

- Documentar el desarrollo de este sistema para una mejor comprensión del mismo.

CAPITULO 1

ANTECEDENTES HISTÓRICOS

CAPITULO 1

Antecedentes Históricos

1.1 El Origen de la Red Integral de Telecomunicaciones de la UNAM.

LA UNAM Y EL DESARROLLO DE LAS REDES DE COMUNICACIONES.[1]

El final de los años 60's y el principio de la década de los 70's marcaron para la UNAM, la etapa de inicio de las comunicaciones telefónicas y de datos. Es en ese periodo, cuando se realizaron las primeras conexiones de teletipos hacia una computadora central, utilizando líneas telefónicas de cobre, de la recién instalada red telefónica dentro de la Institución.

Esta tecnología fue usada en el interior de la UNAM y difundida al exterior, por ello se efectuaban una gran cantidad y diversidad de conexiones, de terminales de caracteres, de graficación e impresión, hasta la interconexión de estaciones de trabajo remotas, todas ellas manejando líneas telefónicas. A partir de la segunda parte de la década de los 80's surge en la UNAM la búsqueda de cambios en las comunicaciones.

Así en 1987, la UNAM establece la primera conexión a **BITNET**, mediante enlaces telefónicos, desde la Ciudad Universitaria hasta el Instituto Tecnológico de Estudios Superiores de Monterrey (ITESM) y de ahí hasta San Antonio, Texas en los EUA.

Posteriormente, la UNAM buscó consolidar su enlace a esa red internacional mediante la computadora IBM 4381, la cual sirvió como residencia del correo electrónico y otros servicios de BITNET; dentro de ese proceso se inició la conexión de terminales IBM con emulación 3270, estableciéndose además un enlace con la Red TELEPAC de la SCT, bajo la finalidad, nunca lograda, de brindar este servicio a nivel nacional. No fue sino hasta 1989, cuando la UNAM a través del Instituto de Astronomía establece un convenio de enlace a la red de la **NSF** en EUA, el cual se realizó utilizando el satélite mexicano Morelos II entre el Instituto de Astronomía en la UNAM y el UCAR-NCAR con residencia en Boulder Colorado, además, se llevó a cabo el primer enlace para conectar las redes de área local, entre el Instituto de Astronomía y la Dirección General de Servicios de Cómputo Académico, utilizando enlaces de fibra óptica .

A partir de ese momento se inició dentro de la UNAM una revolución en las comunicaciones, así como la adquisición masiva de computadoras personales y su interconexión e intercomunicación en redes de área local, principalmente en las dependencias del subsistema de la investigación científica; lo cual permitió desarrollar la infraestructura de comunicaciones con fibra óptica, y establecer más enlaces satelitales hacia Cuernavaca, Morelos, y San Pedro Mártir en Ensenada Baja California Norte, a la par del primer enlace de microondas de alta velocidad entre la Torre II de Humanidades y la Dirección General de Servicios de Cómputo Administrativo, en la colonia del Valle, Ciudad de México.

Con esto último, se estableció en definitiva el final de la era del teleproceso, para dar paso a las redes de computadoras y sus enlaces a través de fibra óptica. En 1990 la UNAM, fue la primera Institución en Latinoamérica que se incorpora a la red mundial

[1] http://www.nic.unam.mx/redunam/historia.html

Internet, que enlaza a millones de máquinas y decenas de millones de usuarios en todo el mundo.

Internet es una red, producto de un proyecto del Gobierno de los Estados Unidos que data de 1970, y en sus primeras etapas (como parte de un programa de investigación militar de **ARPA**) se logra demostrar la viabilidad de las comunicaciones entre computadoras, por medio de la **conmutación de paquetes**; lo cual creó la red **ARPANET**, que enlazó en sus primeros años varias decenas de sitios en una red nacional dedicada a la comunidad de investigación en computación. El concepto de **conmutación de paquetes** se extendió en muy pocos años para incluir redes satelitales y redes basadas en radio. Su ininterrumpido desarrollo que no tiene límite a la fecha, contempla como elemento fundamental el diseño de una arquitectura para comunicar redes que permitan la coexistencia de paquetes de comunicación de diferentes tipos bajo el protocolo TCP/IP; mismo que se mantiene como estándar en la actualidad, dado su funcionalidad y posibilidad de adaptación a los requerimientos que se van presentando.

A finales de los 80's se da la apertura al uso comercial, en tanto se limitaba a proporcionar servicios a la comunidad académica.

Creación de Red Integral de Telecomunicaciones de la UNAM.

A finales de 1989 se estableció un ambicioso proyecto que debía sustituir los antiguos conmutadores para renovar totalmente el sistema telefónico de la UNAM, de acuerdo con los estándares más modernos y con capacidad de crecer conforme a las necesidades de la institución. Para este proyecto que constituye la parte fundamental del Programa Institucional en informática, en la Dirección General de Servicios de Cómputo Académico se creó la Dirección de Telecomunicaciones Digitales cuyo objetivo sería la creación de la Red Integral de Telecomunicaciones de la UNAM, la cual debería ser capaz de transmitir indistintamente datos e imágenes entre las dependencias universitarias independientemente de su ubicación geográfica.

Ante la necesidad de integrar los diferentes servicios y recursos de cómputo como soporte de desarrollo eficiente para la investigación y la docencia, surge el Laboratorio de **REDUNAM** en 1990 (proyecto del Departamento de Redes y Comunicaciones de la DGSCA) como un espacio para el estudio, análisis de comunicación, **topologías** de redes, **protocolos** y servicios, entre otras actividades. La Red Integral de Telecomunicaciones de la Universidad Nacional Autónoma de México se inaugura oficialmente en 1992. Entre sus principales características destacan hoy en día:

- Transmisión indistinta de datos y video, mediante sistemas digitales basados en normas internacionales que rigen actualmente.

- Integración a la red de las principales instalaciones de la Universidad.

Esto significa, que a nivel bachillerato, licenciatura, postgrado e investigación, alrededor del 95% de sus miembros se encuentran en instalaciones cubiertas por la red, en varias regiones del país desde Ensenada, Baja California; hasta Puerto Morelos en Quintana Roo.

El sistema está conformado por 32 nodos operacionales de telefonía enlazados entre sí mediante fibra óptica, enlaces satelitales y de microondas.

Posee una infraestructura instalada para 13,000 servicios telefónicos alimentados por 2,400 troncales digitales conectadas via fibra óptica con las centrales telefónicas públicas.

En 1998 el puente de la Escuela Permanente de Extensión en San Antonio (EPESA) ya cuenta con el enlace dedicado con la Universidad de Texas A&M en *College Station*. Oficialmente, a partir del 25 de Junio, la UNAM forma parte de la Red TTVN[2] (*Trans Texas Video Network*) que tiene acceso a más de 100 salas en los 13 campus de TAMU además de otras universidades y escuelas.

Asimismo el puente de la EPESA cuenta con el enlace directo de la Universidad Autónoma de Nuevo León (UANL) en Monterrey. La conexión de parte de la UANL se encuentra en su Centro Medico que tiene enlaces de circuito cerrado con hospitales y clínicas regio-montanas afiliadas con la UANL.

Además, se cuenta con una red complementaria de respaldo de más de 1000 servicios, basada en telefonía celular y l7 líneas telefónicas directas. La red enlaza a cerca de 10,000 computadoras de la UNAM entre sí y alrededor de 15 millones de computadoras en el resto del mundo.

Se lleva a cabo la instalación de 3,500 servicios nuevos (**BDI** y otros) que se suman a la infraestructura actual, y la actualización de los equipos de datos.

Se realiza una ardua labor para integrar a las principales instalaciones de la UNAM a nivel metropolitano y nacional; a la par de atender los campus de Hermosillo, Ensenada, Martínez de la Torre, Cuernavaca, Juriquilla y Morelia.

Como resultado de una labor ininterrumpida, se cuenta con sistemas de tarificación. Renovar los servicios telefónicos de la UNAM con las tecnologías más modernas y eficientes, implica brindar a la Institución el soporte necesario para el mejoramiento de sus actividades sustantivas.

Así de manera general se presenta a continuación los enlaces para conexión a Internet con los que actualmente cuenta la Red Integral de Telecomunicaciones de la UNAM :[3]

Cantidad	Conexión	Proveedor	Velocidad
1	E3	Avantel	2.048 Mbps
2	E1	Avantel	""
1	E3	Uninet	34.368 Mbps
6	E1	Uninet	""
1	E1	Uninet	""
1	E1	Uninet	""
2	T1	Verio	1.544 Mbps
1	T1	MCI World Com	""

Fig. 1.1

[2] http://ttvn.tamu.edu

[3] Departamento de Redes. D.G.S.C.A. 2001

1.2 Infraestructura y Servicios de RED UNAM :

RED UNAM se define como el proyecto desarrollado para la transmisión de datos entre las Facultades, Institutos, Centros de difusión, Coordinaciones y demás Dependencias que conforman la UNAM

Entre las principales razones del Desarrollo de RED UNAM y de su invención están :

- Descentralización de los servicios de computación
- Modernización de las bases de datos
- Adopción de UNIX y Lenguajes de Cuarta Generación
- Servicios vía RedUNAM y vía telefónica
- Creación de módulos computarizados de atención a estudiantes
- Sistematización computarizada para oficinas
- Fortalecimiento del Programa de Becarios para generación de los recursos humanos requeridos
- Programa de capacitación para personal administrativo
- Sustitución de microfichas por archivos digitalizados
- Servicio de red en oficinas administrativas
- Descentralización del presupuesto para mantenimiento y adquisición de equipo de cómputo
- Modernización de los sistema de impresión

Para poder lograr estos propósitos y llevar a cabo la realización de varios proyectos que son de importancia para el mejor funcionamiento de todos los servicios que ofrece REDUNAM, es necesario tener una estructura capaz de solventar los requerimientos que se deban satisfacer. De esta forma es como se ha creado la dirección de Telecomunicaciones el cual se encargará de cumplir con estas necesidades .

A continuación se muestra un organigrama de la estructura de las diferentes subdirecciones y coordinaciones en que se divide la Dirección de Telecomunicaciones UNAM para una mejor optimización de los recursos y servicios que ofrece al público en general, así como a la comunidad universitaria.[4]

[4] Notas Sobre la Historia de Red UNAM. Departamento de Redes . DGSCA UNAM 1997

Fig. 1.2

De esta forma RED UNAM ofrece una amplia gama de servicios que son de gran utilidad para todos los universitarios así como para dependencias e instituciones que los requieran .

Los servicios de RED UNAM son muy amplios y se dividen de la siguiente manera: [5]

1. En primer lugar aquellos servicios de los que pueden hacer uso los usuarios. Los principales son :

- **Correo Electrónico :**

Es uno de los servicos de mayor demanda, el cual permite la comunicación entre usuarios de cualquier parte del mundo a través de envio o recepción de mensajes de texto en un buzón electrónico . Los sistemas de correo electrónico tienen el enfoque cliente-servidor, esto es, dos programas cooperan para transferir un mensaje de correo electrónico desde la computadora del transmisor hasta el buzón del receptor.Cuando un usuario envía un mensaje, el programa en la computadora del remitente se vuelve cliente. Contacta un programa de servidor de correo electrónico en la computadora del receptor y transfiere una copia del mensaje . El servidor almacena el mensaje en el buzón del receptor. [6]

RedUNAM cuenta con varios servidores de correo, dos de los cuales están destinados al público universitario y académico en general : servidor.unam.mx y correo.unam.mx .

El Protocolo Simple de Transferencia de Correo (**SMTP** o *Simple Mail Transfer Protocol*), es el protocolo estándar de Internet para enviar y recibir correo electrónico . SMTP en sí no es un problema de seguridad, pero lo pueden ser los servidores de SMTP .

[5] www.unam.mx

[6] Internet . Douglas E. Comer. Prentice Hall. México 1995 . Pag 147

Un programa que entrega correo a usuarios con frecuencia necesita ejecutarse como cualquier usuario que recibe correo. Esto lo puede hacer un blanco para algún atacante. El servidor SMTP más común en UNIX es ***Sendmail***.

CORREO ELECTRONICO

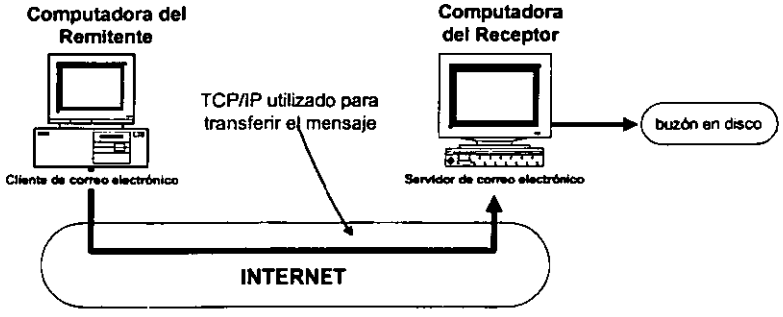


Fig. 1.3

- **Gopher :**

Este servicio es un rastreador interactivo que se emplea por medio de menus jerárquicos que permiten buscar información en RedUNAM e Internet a través de conexiones transparentes al usuario.

Cuando se ejecuta *gopher*, aparece un menú de opciones. Estas por lo general consisten en frases cortas. Así, cada opción en un menú de gopher denota un archivo de información o una referencia a otro menú.El usuario ve la lista de operaciones en el menú y escoge una de ellas . Si la opción seleccionada corresponde a un archivo de informacion, el gopher recupera el archivo y muestra su contenido; si la opción seleccionada corresponde a otro menú, el sistema gopher recupera el nuevo menú y permite al usuario seleccionar una opción de él.

Gopher utiliza también la interacción cliente – servidor; el usuario llama al software *cliente gopher* en la computadora local e interactua con él para seleccionar opciones del menú y controlar la búsqueda de información. El cliente gopher contacta uno por uno a los servidores gopher. Siempre que el usuario selecciona un menú, el cliente gopher utiliza Internet para recuperar y mostrar la información.

En la UNAM se cuenta con una gran variedad de servidores Gopher, por ejemplo: Condor y Cuk que contienen información de interés general y concerniente a la UNAM.

GOPHER

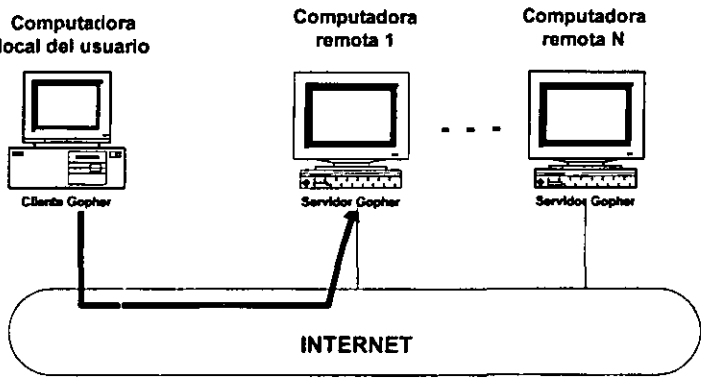


Fig. 1.4

- **FTP :**

El protocolo de Transferencia de Archivos (FTP : *File Transfer Protocol* , es el protocolo estándar de Internet para este propósito) proporciona el servicio para la transferenica de archivos a, o desde una computadora remota.

Para utilizar el FTP, el usuario llama a una aplicación del FTP en su computadora local. Esta aplicación acepta una serie de comandos de manera interactiva. Durante la sesión interactiva, el FTP responde a cada comando que introduce el usuario. Al iniciar una sesión de FTP , el usuario identifica una computadora remota e instruye al FTP para establecer una conexión. El FTP utiliza TCP/IP para ponerse de acuerdo con la otra computadora. Así, una vez establecida esta comunicación, el usuario interactua directamente con la computadora remota.El usuario puede obtener una lista de archivos disponibles en la computadora remota o copias de uno o más archivos así como actualizar al servidor remoto agregando archivos o directorios , siempre y cuado cuente con los permisos necesarios .

FTP utiliza un acceso cliente-servidor. El usuario llama un programa FTP en la computadora, lo instruye para contactar a una computadora remota y solicita la transferencia de uno o más archivos. El programa local de FTP es un cliente que utiliza TCP para conectarse con un servidor FTP en la computadora remota. Cada vez que se solicita transferecina de un archivo, cliente y servidor cooperan para transmitir una copia de los archivos a través de internet.

RedUNAM cuenta con servidores de archivos que contienen información diversa y que son accesibles a través de FTP ANONIMO, esto es , tiene el convenio de dejar el acceso libre pero controlado al usuario que ha ingresado como "*anonymous*".

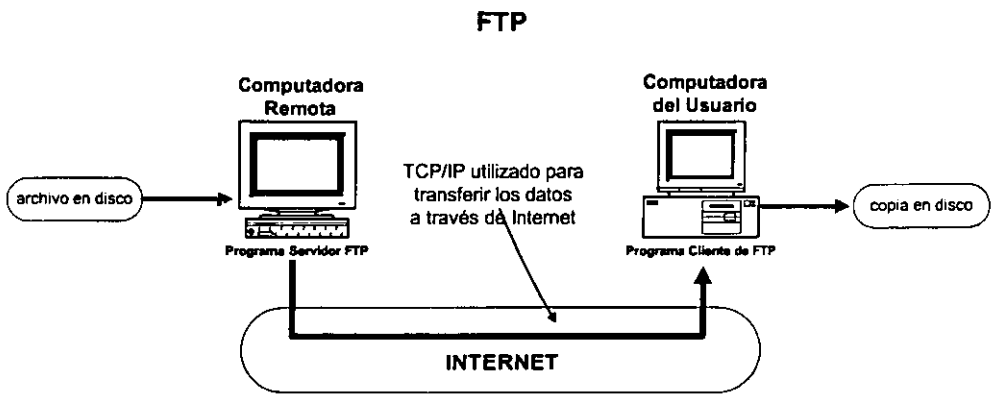


Fig. 1.5

- **Telnet (Acceso remoto) :**

Es un servicio remoto permite que el usuario interactue con programas de aplicación que corran en una computadora ubicada en otro sitio.

El acceso remoto sigue el paradigma cliente-servidor . Esto es, cuando el usuario en una computadora local desea accesar a un sistema remoto, llama a un programa de aplicación local para el servicio de acceso remoto e introduce el nombre de la computadora remota que desea contactar. La aplicación se convierte en un cliente que utiliza TCP/IP para conectarse, a través de Internet, con un servidor en la computadora remota. El servidor envía exactamente la misma solicitud de acceso utilizada en las terminales convencionales. Ya establecida la conexión entre el cliente y el servidor, el software permite al usuario interactuar directamente con la computadora remota. Cuando el usuario presiona una tecla o mueve el "ratón", la aplicación cliente envía los datos a través de la conexión a la máquina remota. Cuando el programa de aplicación de la máquina remota produce una salida, el servidor la envía al cliente.

El estándar de Intenet para el servicio de acceso remoto se encuentra en un protocolo conocido como TELNET. Este protocolo especifica de manera exacta como interactúa un cliente de acceso remoto con un servidor de acceso remoto.

Se pueden establecer sesiones remotas a grandes computadoras para aprovechar sus altas capacidades de cálculo y otros recursos que difícilmente se encuentran en máquinas menores.

La UNAM cuenta con varios sistemas UNIX para los cientos de proyectos de investigación y recopilación de información que se desarrollan en los distintos institutos y centros de difusión.

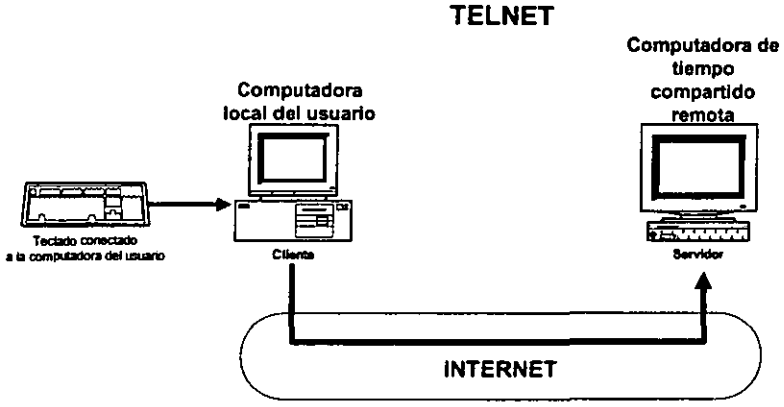


Fig. 1.6

- **World Wide Web (WWW) :**

Es uno de los servicios más interesantes y completos que se pueden encontrar en Internet . Este servicio aprovecha la tecnología de multimedia para ofrecer una presentación de la información mucho más interesante mediante el uso de imágenes, texto y sonido desde cualquier punto de Internet a su computadora.

El *WWW* basa su funcionamiento en el protocolo *HTTP* (*HyperText Transfer Protocol*, Protocolo de Transferencia de Hipertexto), el cual permite la transferencia de hypertexto en la red. Este protocolo permite el uso de otros servicios como *FTP, gopher, news, telnet, e-mail*[7].

Existen una gran cantidad de Servidores *WWW* en la UNAM, que ofrecen información de gran interés.

Así como los servicios anteriores, el *World Wide Web* utiliza la interacción cliente – servidor, esto es, el usuario comienza la interacción al llamar a un *browser* o navegador (puede ser : *Netscape Comunicator, Internet Explorer, StarOffice*, etc). El browser es un cliente que utiliza Internet para ponerse en comunicación con un servidor remoto y obtener una copia de la página para ser presentada en la pantalla con información adicional que describe el contenido.

En la información adicional, un servidor WWW envía al *browser* la descripción de la página y como será presentada así como los *URL's* o ligas para cada opción que el usuario pueda seleccionar en ella.

Cuando el *browser* recibe una página de un servidor remoto, la despliega y espera a que el usuario seleccione alguna parte de ella , ya sea a través de una liga de texto, de una liga de imagen o realizando alguna petición que requiera parámetros adicionales para realizar algun proceso.

Actualmente los servidores de *WEB* nos ofrecen una gran cantidad de servicios y herramientas para interactuar con ellos a través de sus páginas, de tal forma que nos es casi imposible no encontrar cosas nuevas cada día. Entre lo que más destaca está lo que hoy en día se le conoce como comercio electrónico, en el cual el usuario a través de imágenes, sonidos y video es incitado a comprar algún producto que alguna compañía le ofrece. Si el usuario le interesa, podrá pedir este producto a través de alguna forma en la

[7] http://webdia.cem.itesm.mx/ac/rtrejo/Interfaz/www.html

que deberá llenar los datos necesarios para esa transacción , entre ellos, el número de su tarjeta de crédito con el cual la compañía podrá realizar el cobro. Todas estas transacciones deben de realizar a través de un servidor seguro que encripte esta información. Una herramienta para este tipo de servidores es la llamada *Secure Socket Layer* (***SSL***).

A través de la *WWW* podemos también obtener cualquier cantidad de información, aplicaciones , programas, tutoriales, consultar y/o administrar bases de datos, desplegar videos, sonidos, comunicación con otras personas en cualquier parte del mundo, foros de discusión de casi cualquier tema, etc.

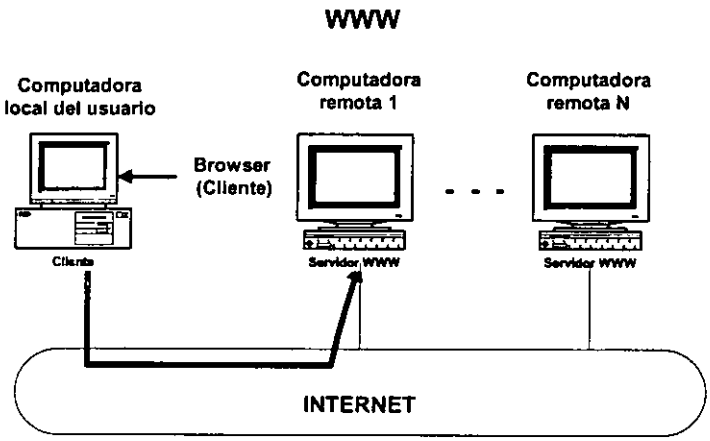


Fig. 1.7

2. Todos los servicios anteriores son a su vez soportados por otros servicios que permiten la comunicación entre los diversos *hosts* (máquinas que fungen como clientes y como servidores). Estos servicios son administrados de la siguiente manera :

- **NOC** (Centro de Operación de la Red / ***Network Operation Center***)

El Centro de Operación de RED UNAM, se encarga de monitorear el comportamiento de la red, así como la operación de ésta y darle el mantenimiento necesario. El objetivo primordial del NOC es el mantener en óptimas condiciones la operación de la red y de los dispositivos conectados a ella. Entre sus actividades realizadas para llevar a cabo ese objetivo está la detección, determinación, atención y solución de fallas en la red. En otras palabras, es el encargado de mantener funcionando de manera eficiente la interconexión de las redes locales, los enlaces de área amplia y la "Columna Vertebral" o *Backbone* de la Red Universitaria.

- **NIC** (Centro de Información de la Red / ***Network Information Center***)

El Centro de Información se encarga de distribuir la información de los servicios de red, soportarlos dentro de la RedUNAM y dar la capacitación necesaria a los usuarios sobre estos servicios.

Servicios administrados por El Centro de Información de RedUNAM (*NIC*) : [8]

- La asignación de direcciones *IP*

El *Asignar* una dirección *IP* es autorizar a los responsables de una Dependencia o Institución conectada a la UNAM el uso de un rango de direcciones *IP* pertenecientes o asignadas a RedUNAM. Cualquier solicitud o trámite de asignación debe ser realizado mediante los formatos y procedimientos establecidos por NICUNAM

- La asignación de dominios

El Sistema de Nombres de Dominio dentro de Internet, se basa en la estructura del espacio de nombres de dominio. Esta estructura es de forma arborecente y jerárquica semejante a un sistema de archivos.

Asignar es autorizar a los responsables de una Dependencia la utilización de un dominio con el objetivo de asociarlo a sus direcciones *IP*, así como la posibilidad de enviar solicitudes de ALTAS y/o BAJAS de nombres, alias y/o **mail exchangers** relacionados al mismo. Cualquier solicitud o trámite de asignación debe ser realizado mediante los formatos y procedimientos establecidos por NIC UNAM.

- El servicio de nombres

En Internet, es fundamental y conveniente que todos los equipos de cómputo que tengan una dirección *IP*, tengan asociado un nombre y dominio. Esto permite, entre otras cosas, el envio de correo electrónico, acceso a páginas de web, etc.

Actualmente RedUNAM cuenta con cuatro Servidores de Nombres encargados de la resolución de diversos dominios, principalmente UNAM.MX:

dns1.unam mx [132.248.204. 1] (Nodo DGSCA)
dns2.unam.mx [132.248.10 . 2] (Nodo DGSCA)
dns3.unam.mx [132.248.64 .250] (Nodo **IIMAS**)
dns4.unam.mx [132.248.237.250] (Nodo ZONA-CULTURAL)

Estos servidores son administrados y operados única y exclusivamente por el Centro de Información de RedUNAM (NICunam)

[8] ftp://ftp.nic.unam.mx/Politicas/DirIP-Politicas

1.3 Departamento de Administración de Servidores

Como ya se mencionó, la Dirección de Telecomunicaciones se encarga de operar y administrar la Red Integral de Telecomunicaciones de la UNAM (**RIT** UNAM). En la parte de datos (RED UNAM), se encarga también de administrar, a través de su Departamento de Administración de Servidores, los servicios centrales de Internet para la Universidad:

Estos servicios son otorgados tanto a dependencias internas de la UNAM como a externas, así como a usuarios individuales que comprenden tanto a personal de la misma Universidad como a particulares externos que simplemente quieren hacer uso de la infraestructura y servicios de la UNAM.

El departamento de Administración de Servidores se encarga de la administración de los servidores centrales de la red universitaria de datos, **RedUNAM**. Su objetivo es mantener en óptimas condiciones de operación y funcionamiento, servicios como:

- Correo Electrónico Central .

  Este servicio, utilizado para enviar y recibir mensajes, concentra la mayor cantidad de usuarios de RedUNAM. Es proporcionado por un servidor central llamado servidor.unam.mx . Actualmente, se está dando servicio a más de 21,000 usuarios y en promedio se procesan 65 000 correos diarios.

- Lista de distribución de Correo Electrónico .

  Otro servicio que proporciona servidor.unam.mx es el de listas de discusión a través de ***Majordomo***. Actualmente existen más de 60 listas, que incluyen aproximadamente a 3 000 usuarios dentro del propio servidor y otros sitios a nivel mundial. Estas permiten a un conjunto de usuarios intercambiar ideas, de tal forma que al enviar un mensaje a una lista, *Majordomo* se encarga de distribuirlo a las cuentas de cada uno de los integrantes de dicha lista.

- World Wide Web Central de la UNAM .

  La UNAM proporciona su servicio de páginas WWW a través de un servidor central http://www.unam.mx . Sin embargo, este contiene a más de 180 sitios de información de propósitos general.

- Almacenamiento Masivo .

  Este servicio consiste en proporcionar al usuario un espacio de almacenamiento secundario y siempre que sea posible, se intenta adaptar el procedimiento de acceso a las necesidades del mismo usuario.

Otros servicios importantes que se administran en el departamento son :

- FTP Anónimo de la UNAM
- Gopher Central
- Sistema de impresión Laser

Estas actividades, permiten a los integrantes del Departamento de Administración de Servidores, especializarse en la administración de equipos UNIX; la instalación y mantenimiento de servicios en Internet; el desarrollo de aplicaciones en redes propietarias; administración de sistemas de bases de datos; el diseño e implementación de esquemas de seguridad para servidores en red, entre muchas otras.

Es por esto, que una de las actividades de este departamento es la asesoría gratuita a todas las dependencias universitarias en la administración de servidores instalados en RedUNAM.

Las facilidades en cuanto a la obtención de dichos servicios ha provocado que sean muy populares sobre todo entre la comunidad universitaria. Para obtener una cuenta de correo electrónico y una de acceso vía módem solo basta ser empleado, investigador, profesor o tener algún cargo en la Universidad para tener estos servicios de manera gratuita. Los estudiantes tienen tarifa especial al comprobar estar estudiando en la UNAM.

El servicio desde sus comienzos ha crecido exponencialmente por lo que se ha necesitado la evolución de los medios con que se crean y administran las cuentas de correo electrónico y acceso vía módem. Cuando ya no fue posible que el propio administrador de los servicios fuera quien tuviera contacto directo con el usuario se creó un área especial para esas tareas la cual sirvió de intermediario entre el usuario y el administrador.

Aún cuando las tareas ligadas a la atención del usuario ya no eran realizadas por el administrador, la creación y manejo de tantas cuentas de correo electrónico y acceso vía módem ya no podían ser manejadas por él a la par de las labores de administración y mantenimiento de los servicios y las máquinas.

Se crearon entonces herramientas que facilitaran esas tareas. El departamento encargado en ese tiempo del servicio creó un sistema con una interfaz gráfica muy básica a través del Web el cual pretendió automatizar las labores relacionadas a la creación y administración de claves de correo y acceso vía módem.

Este sistema no realizaba más que las tareas básicas de alta, baja y renovación de claves y su interfaz era muy simple ya que solo contenía las "cajas" para la inserción de los datos. Cabe mencionar que las operaciones que realizaba al momento de la inserción, baja o renovación de alguna clave no era *limpia* ya que, debido a la mala planeación, diseño, codificación e implantación del algoritmo utilizado, los datos en el servidor se corrompían, esto es, dejaba de haber coherencia en los archivos de sistema que regulan el acceso a los usuarios, y constantemente se tenían problemas con el sistema y por consiguiente con el usuario el cual se quejaba frecuentemente de no poder accesar a su cuenta o de pérdidas de información.

El servidor en la cual radican las cuentas de correo electrónico, había pasado por una administración deficiente y la falta de organización y compromiso de la gente que administró estos servicios había causado una notable problemática para su administración.

Para dar una mejor idea de la magnitud de estos servicios, actualmente se administran, más de 21,000 cuentas de correo electrónico, la gráfica muestra las estadísticas de crecimiento de dicho servicio:

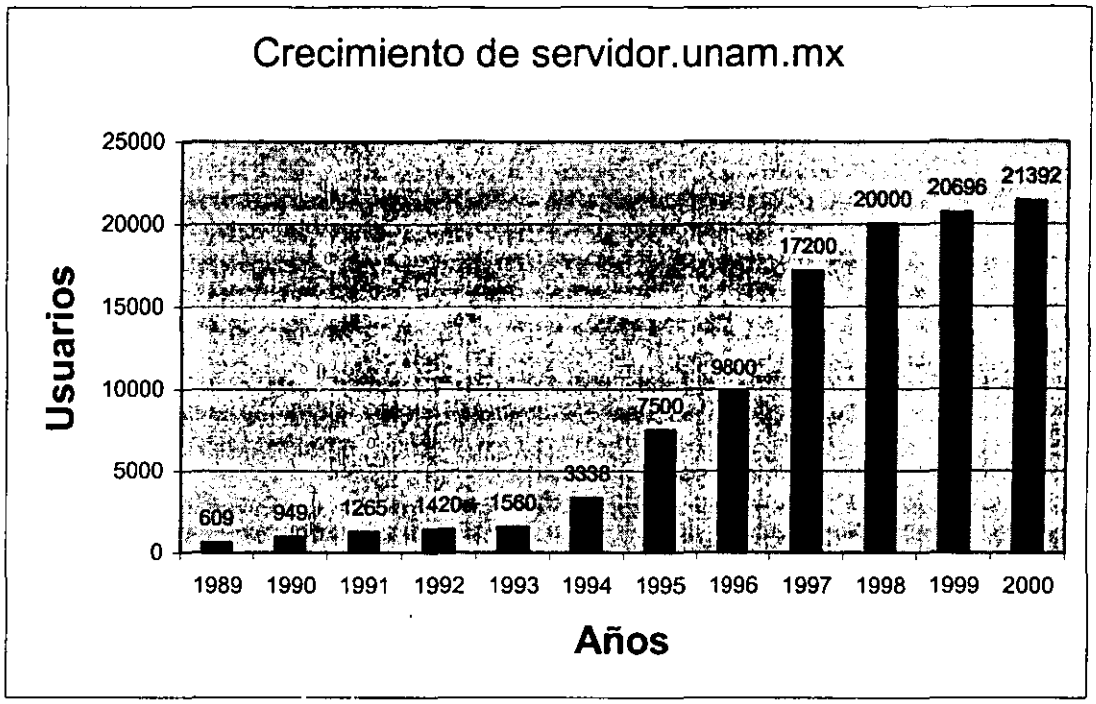


Fig. 1.8

Si dichas claves cuentan con una vigencia de un año, esto significa que se debe efectuar la renovación de 100 claves de correo electrónico diariamente, esto sin considerar el crecimiento anual de estos servicios, estimado para el último año en aproximadamente 5,000 usuarios.[9]

Para dar respuesta a la demanda de atención por parte del usuario, la Dirección de Telecomunicaciones creó el Departamento de Proyectos Especiales, que a través de su área de Atención a Usuarios recibe las solicitudes de los usuarios y da respuesta con la ayuda de una base de datos y formas de solicitud impresas.

Debido al reducido presupuesto asignado para la contratación de personal en la dependencia, el área de Atención a Usuarios sólo cuenta con tres personas encargadas de realizar altas, renovaciones, cambios y consultas de claves de correo electrónico y acceso vía módem, así como registrar los pagos correspondientes. Resulta evidente la necesidad de contar con una herramienta que permita la automatización de estas actividades con el fin de proporcionar un servicio más rápido y eficiente al usuario.

[9] Departamento de Servidores . D.G.S.C.A. 2000

1.4 La administración de servicios de correo electrónico en otras Instituciones

La importancia de conocer la infraestructura y los servicios de correo electrónico que ofrecen otras instituciones educativas es con el fin de poder comparar la magnitud del problema que se aborda para la administración de estos servicios. Es así como se puede tener una idea más exacta de las necesidades.

A continuación se presentan las características de algunas Instituciones educativas que cuentan con los servicios de Administración de Correo Electrónico y en algunos casos de servicios para cuentas de acceso vía módem, así como de los servicios que estas dependencias ofrecen:

Servicios de IPN-Red [10] :

Correo Electrónico (*E-mail*) , Acceso remoto (*Remote login*) , ***Finger*** , *FTP* anónimo (*Anonymous FTP*) , *Gopher* , ***Archie***, *Telnet, World Wide Web, Veronica y Judghead.*

Servicios de Internet del Colegio de México [11]:

En cuanto a telecomunicaciones se refiere, el Colegio de México cuenta con un enlace de microondas para la conexión a *Internet*. Este recurso permite, tanto a nivel nacional como internacional, usar equipo de cómputo y programas de aplicación de alguna Institución académica y poder integrarse a grupos de trabajo sobre temas específicos, así como tener acceso al correo electrónico y a bancos de datos.

El Colegio cuenta con un sistema de cableado estructurado que permite conectar todas las computadoras e impresoras con las que cuenta la Institución, en una red local que da la facilidad de compartir recursos y obtener información interna, nacional e internacional.

El equipo principal que integra la red local está conformado de la siguiente manera: un servidor de correo *SUN SPARC* 20 con sistema operativo *Solaris* 2.6, esta máquina cuenta con 32 MB de memoria *RAM* y 6 GB de disco duro.

Existe un equipo *Alpha server* 2100 con 128 MB en memoria *RAM* y 20 GB en disco duro, además de dos procesadores con una velocidad de 275 MHZ. Un *cluster* Digital con dos servidores *Pentium* II de 266 Mhz y 12 GB de disco duro. 4 servidores *Pentium* de 166 mhz con 64 MB *RAM*, el sistema operativo instalado en estos servidores es Windows NT 4.0. Además se cuenta con tres servidores *Netware* 4.1 con las mismas características.

Estos equipos prestan el servicio de consulta y uso de Bases de Datos, manejo de correo electrónico, conexión a Internet, así como la administración de los servicios de impresión y del software de uso común a más de 400 equipos e impresoras que se conectan a la red local.

[10] http://www.ipn.mx/ - 2000
[11] http://www.colmex.mx/areas_apoyo/computo/computo.htm - 2000

Servicios de Internet de la UAM [12]:

Antecedentes :

Anteriormente el servicio de acceso a la TeleUAM vía módem sólo permitía el uso del servicio de conexión de terminal remota (*telnet*) hacia equipos multiusuarios ubicados dentro de la ***UAM***.

El servicio de conexión a la TeleUAM vía ***PPP*** permite que los usuarios, desde sus domicilios, exploten casi todos los servicios de red, soportados por TCP/IP, que pueden explotar desde el interior de la UAM.

El servicio de acceso a la TeleUAM vía *PPP* comenzó a operar el 2 de Enero 1996.

Servicios de Internet que ofrece:

Una vez que se le asigne la cuenta y clave de acceso, el usuario deberá obtener tanto los programas de comunicación como los programas de aplicación que al ser instalados en su computadora le permitirán acceder a los servicios de red que la TeleUAM ofrece hacia el interior de la UAM, tales como:

Telnet, Ftp, Email, WWW y Gopher.

Servicios de Internet que ofrece la UVM [13] :

La Universidad del Valle de México cuenta con el siguiente equipo para ofrecer sus servicios de Internet :

Tienen 15 campus en todo el país; cada uno de ellos cuenta con un servidor Intel *Pentium* III a 300-450 Mhz, 512 Mb RAM y RAID5 con tres discos de 8 Gb cada uno, haciendo 12 Gb de espacio utilizable.

Corre ***Netscape Messaging Server*** 3.5 bajo NT 4.0, aunque está por cambiar a ***MS Exchange*** 5 debido a problemas de rendimiento. Aunque algunos servidores sólo tienen 800 ó 900 usuarios, algunos tienen hasta 5,000. Cada campus tiene un enlace E0 hasta un nodo central ubicado en Tlalpan y de ahí la salida a Internet es de 2E0.

[12] http://www.uam.mx/ - 2000
[13] http://www.uvmnet.edu/ servicios/servicios.htm - 2000

1.5 El Sistema de Administración de Cuentas

Características de los equipos de los servidores de correo electrónico .

Para Octubre de 1994, existían 2500 usuarios de correo electrónico en los servidores redvax1 y unamvm1, quienes generaban más de 8000 mensajes diarios. La actividad de correo electrónico en esos servidores había aumentado a una tasa del 35% mensual durante los últimos tres meses a esa fecha, lo que excedía las expectativas que se habían planteado.

Debido a esto y a la preocupación por mejorar el servicio, se informó que a partir del 1o de Enero de 1995, se migraría a los usuarios de estos servidores a un equipo de mayor capacidad. Ese equipo cubría las siguientes caracteristicas: ***Sun SPARCserver*** 1000 con 128 Mb RAM y 12 Gb de Disco Duro.

El domicilio electrónico de los usuarios de redvax1 y unamvm1 cambiaría a un domicilio unico de la forma: usuario@servidor.unam.mx.

El nuevo equipo sería **servidor.unam.mx** con dirección IP 132.248.10.1

Actualmente, el servicio de correo electrónico es proporcionado por un servidor central que conserva el nombre de **servidor.unam.mx** . Este es un equipo ***Sun Enterprise*** E3500 con 1024 MB RAM, 4 procesadores ***Ultra Sparc*** II de 400 Mhz y 264 GB de Disco Duro.

Para brindar este servicio se hace uso de *sendmail* 8.9.3 como agente de transporte de correo, además del ***Post Office Protocol (POP)*** e ***Internet Mail Agent Protocol (IMAP)*** para que los usuarios puedan utilizar herramientas tales como *Eudora, Netscape, PCPine, Outlook y Exchange*, entre otros. Además los usuarios pueden hacer uso de los recursos del servidor realizando una sesión remota a través del *Telnet*.

Todos estos equipos están conectados entre sí mediante la red de datos RedUNAM utilizando el conjunto de protocolos TCP/IP. Esta red, está a su vez conectada a la red mundial Internet.

Planteamiento del problema y su solución.

Así, para que la UNAM pueda obtener el máximo provecho de los servicios que ofrece y al mismo tiempo dar un servicio más eficiente, se debe simplificar el tiempo de respuesta en el proceso de altas o modificaciones de cuentas de correo electrónico . Esto es debido a la consideración de las tareas que se realizan en el Departamento de Atención a Usuarios.

El usuario que tiene la necesidad de adquirir una clave de correo electrónico debe llenar una forma de solicitud del servicio y en caso de ser personal externo a la Universidad Nacional Autónoma de México, tendrá que pagar la cuota destinada para el mismo. Los datos de esta forma son capturados en una base de datos por el personal de dicho departamento, posteriormente ésta es llevada al Departamento de Administración de Servidores para que el administrador del servidor de correo electrónico realice la operación correspondiente, esto es utilizando una serie de comandos en sistema operativo UNIX, todo esto ocasiona que el proceso de hacer cualquier movimiento en el servidor, en cuanto a claves se refiere, consuma demasiado tiempo, y si se toma en cuenta el crecimiento demográfico que está teniendo la UNAM y la cantidad de personas que demandan estos

servicios, además de la disminuida cantidad de personal que atiende esta demanda, lo hacen aún más lento, dando origen a un problema que se resolverá.

La solución para este problema es la elaboración de un sistema que cumpla con las siguientes características y pueda funcionar correctamente con las herramientas de software y hardware con las que cuenta la dependencia.

La infraestructura con la cual se prestan los servicios se muestra en el siguiente diagrama:

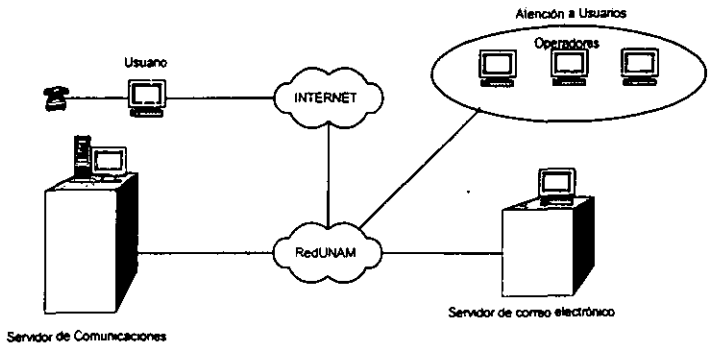


Fig. 1.9

Software :

Para seleccionar el software que vamos a utilizar para el desarrollo del sistema es necesario que éste pueda funcionar de manera adecuada y correcta con el hardware antes mencionado, es por eso que decidimos hacer uso de software de dominio público y de algunas herramientas de desarrollo que se tienen en la dependencia.

Es por esto, que se decide utilizar el compilador de lenguaje de programación C: ***gcc*** , que es un software de dominio público o en su defecto ***CC*** que viene integrado en una instalación completa de un servidor con sistema operativo UNIX, para el desarrollo de lo que será la interfaz gráfica con la que el operador del sistema podrá tener un ambiente amigable de trabajo, además de ser utilizado para la programación de los ***sockets*** de comunicación que se encargarán, precisamente, de la comunicación entre la interfaz gráfica en la que el operador introducirá los datos y el servidor de correo electrónico. Otra de la herramientas a utilizar es el manejador de base de datos ***Sybase***, que será el encargado de hacer todas las tareas de manipulación y organización de la información en una base de datos, en la cual se tendrá un control de cada una de las claves de los servicios proporcionados asi como de los usuarios de dichas claves. Además también se hará uso del lenguaje de programación *HTML* que junto con el lenguaje de programación C y con algunas subrutinas de programación en el lenguaje *JavaScript*, harán que la forma en la que sean introducidos los datos de las claves sea más vistosa al operador; hay que considerar que esta forma es una página de *Web* por lo que es necesario también tener un servidor de páginas *Web* que será puesto en marcha con una utilería llamada ***Apache Web Server*** que también es de dominio público y tiene la capacidad para abastecer la cantidad de peticiones que le sean solicitadas. Finalmente, este servidor de Web debe ser seguro por lo que haremos uso *de Secure Socket Layer (SSL),* también de dominio público, para poder realizar una conexión segura en *Internet*.

Por otra parte, una de las partes más importantes de este sistema es poder contar con la información que permita conocer el estado de los usuarios en cuanto al vencimiento

de las claves de correo electrónico, esto se puede lograr enviando varias notificaciones al usuario unos días antes de que expire su clave. Este proceso desde luego debe ser automático, debido a la cantidad de usuarios. Finalmente el usuario decidirá si quiere renovar el servicio .

Características a Desarrollar.

Después de considerar las herramientas de software que se utilizarán y el hardware que se tiene destinado para la integración de este sistema, se pueden establecer algunos requisitos que éste debe cumplir:

En primera instancia, debe ser una **herramienta fácil de usar**, puesto que el personal que lo operará no cuenta con profundos conocimientos de administración de este tipo de servicios, por lo que no debe presentar ningún procedimiento que cause confusión con el operador.

Por otro lado, debe ser **seguro**, ya que administrará claves que pertenecen a estudiantes, académicos, investigadores, funcionarios, universitarios en un conjunto de servidores conectados a la red Internet por lo que tienen comunicación con el resto del mundo.

Además, tiene que ser **ágil**, por la cantidad de usuarios a los que hay que dar respuesta diariamente, considerando que se contempla un incremento de usuarios en los servicios.

Desde luego, debe ser **confiable**, para asegurar la integridad de la información de los usuarios y la permanencia del servicio proporcionado.

También, debe permitir la **generación de estadísticas** de uso para así poder dimensionar el crecimiento y asegurar el futuro de estos servicios, así como tener un cierto control de los mismos en determinados periodos de tiempo.

Así mismo, debe facilitar la **interacción del operador con el usuario** y así poder resolver sus dudas, o bien canalizar su problema con el área técnica correspondiente, es decir, si el sistema tiene algún problema para realizar la operación solicitada por el operador, el sistema mandará un mensaje indicando que procedimiento se debe seguir o en su defecto indicar el por que no se puede llevar a cabo dicha operación para que el operador consulte o transfiera esta situación directamente con la persona encargada del sistema, en este caso el administrador.

Debe ser **robusto**, por la carga de información que manejará y por las peticiones simultáneas que se llamarán tanto a la base de datos como al servidor de correo .

Lo anterior nos da la pauta para que el sistema sea capaz de bloquear automáticamente la clave de un usuario cuando haya concluido el tiempo por el que contrato el servicio, esto es, no le va a permitir el acceso el servidor a dicha clave hasta que decida renovar el servicio.

Tomando en cuenta lo anterior se puede lograr la integración de un sistema eficiente que va a permitir realizar altas, bajas, cambios y renovaciones de claves de usuarios de los servicios proporcionados así como hacer consultas de estos datos, además de tener un amplio control de las organizaciones o dependencias internas o externas a las que pertenezca el usuario y tener una tabla de ingresos monetarios que percibe la Institución por prestar estos servicios, con lo que podemos garantizar el hecho de que se va a contar con una administración adecuada de estos servicios y el servicio de atención a los usuarios mejorará notablemente.

De esta forma el sistema debe ser capaz de dar **altas** puesto que es el principal servicio demandado en el servidor central de correo electrónico de la UNAM, en este caso el usuario debe proporcionar sus datos personales y el nombre de la clave que desee, de esta forma se podrá tener una información general y por tanto un mejor control de la persona que solicita el servicio, además de que el operador del sistema procesará la clave en el servidor pretendido y el usuario podrá hacer uso de ella casi inmediatamente. Con esto se van a evitar algunos trámites que hacen que el servicio sea lento.

Además se requiere de dar **bajas** debido a que existen muchos usuarios que no usan más su clave de correo electrónico y esto ocasiona que se agoten los recursos de hardware, en este caso almacenamiento en disco duro, que puede ser ocupado por información nueva de recientes altas de claves. Hay que considerar que se le tiene que dar un tiempo razonable (2 meses) al usuario para conservar su información en el servidor, aún cuando éste no haya renovado su servicio, y si no se ha recibido alguna respuesta por parte del interesado, se dará por hecho que el usuario no quiere seguir utilizando su clave en el servidor y el operador del sistema dará de baja la clave, perdiéndose de esta forma tanto los datos de esa clave en el servidor como los datos del usuario en la base de datos, si fuese ésta la única clave que él tuviese. Con esto, a la larga se podrá hacer una depuración del servidor y la información contenida en el mismo estará más organizada.

Otro módulo importante es el de **cambios**, en el que se podrán hacer algunos cambios o modificaciones de los datos del usuario, en el caso de que éste haya cambiado de domicilio, teléfono, etc., con el propósito de tener actualizados los datos del usuario en una base de datos y brindarle un mejor servicio.

Una de las tareas más solicitadas, junto con las altas es la **renovación** de las claves de los servicios y esto es muy importante porque existen muchos usuarios que no quieren quedarse sin servicio, por lo que renuevan sus claves incluso antes de que éstas expiren, ocasionando que éste sea uno de los procesos con más carga en el servidor. En este módulo habrá opciones, que permitirán al operador seleccionar por cuanto tiempo el usuario contrata el servicio. Con esto se actualizará la fecha de expiración de la clave tanto en una base de datos como en el servidor.

Otra actividad muy demandada es el **cambio de *password*** (contraseña) de la clave del usuario en el servidor. Puesto que se utilizan servidores con sistema operativo UNIX, es a veces complejo cambiar su *password* para algunos usuarios que no están familiarizados con este sistema operativo, por lo que requieren que su contraseña sea cambiada por el administrador, lo que permitirá también tener la contraseña actualizada en la base de datos.

Las **consultas** son muy útiles en todos los sistemas, en este caso permitirán al operador tener un amplio control sobre la información del usuario y de su(s) clave(s). Aquí el operador podrá hacer consultas por diversos campos, lo que le permitirá tener de forma más rápida y directa los resultados de una consulta.

Otro módulo importante es el de **Pagos**, en el que se registrarán todos los pagos que efectúen los usuarios por la contratación del servicio, de esta forma se podrá tener un control de los ingresos captados por estos servicios.

Obviamente, será necesario elaborar algunos **Reportes** que permitan conocer información específica de los usuarios, de las claves o de las dependencias, por lo que un módulo que haga esto será muy útil.

Además se requiere que se pueden hacer altas, bajas, cambios y consultas de **Instituciones** o Dependencias, ya que cada día hay más demanda en contratar estos servicios y como está abierto al público en general, llega gente de distintas Dependencias, por lo que es necesario tener un control de las Instituciones a las que la UNAM está prestando los mismos.

Conociendo la infraestructura de RedUNAM, así como los servicios que la conforman y partiendo de las características del equipo con que se cuenta en el departamento de Administración de Servidores de la D.G.S.C.A. se decide la realización de un sistema que facilite la administración de las cuentas de correo electrónico de RedUNAM.

CAPITULO 2

LA BASE DE DATOS

CAPITULO 2

La Base de datos

2.1 Conceptos generales de Bases de Datos [14]

Antes de describir la estructura de la base de datos que almacena la información pertinente de todos los usuarios de correo electrónico, es necesario revisar de manera general algunos conceptos teóricos de bases de datos para entender claramente la forma en que se dispone de la información y la necesidad de migrar la información ya existente hacia una base de datos más confiable y con una estructura adecuada a las necesidades de nuestro proyecto, así como también vislumbrar la inminente necesidad de reorganizar estos datos.

2.1.1 Descripción de Base de Datos

Una base de datos representa a una colección de datos con información relacionada entre si. Este conjunto de datos describen características de algún objeto o cosa.

Una base de datos almacena información (datos) , en una serie de objetos dentro de la base de datos, como tablas, las cuales se relacionan entre si unas con otras. Una tabla es una colección de renglones o registros que han sido asociados con columnas conteniendo registros individuales.

De esta manera se puede tener una base de datos de los empleados de una institución, en la cual se guarde la información pertinente a cada uno de estos empleados, de tal forma que obteniendo estos datos podemos tener una idea de las características de estos como puede ser su nombre, dirección y teléfono, así como el sueldo que percibe en dicha institución, su antigüedad en la institución así como las horas que debe permanecer trabajando en ésta, etc.

También podríamos tener un base de datos de los productos que produce un fábrica de plásticos, por ejemplo. De tal manera podríamos echar un vistazo a esta información y ver que existen diversos artículos que son fabricados con plástico en esta empresa, podríamos hablar de objetos plásticos que tienen nombre, color, tamaño, precio, peso, etc. , es decir, a través de la base de datos podemos almacenar muchas de sus características que los describen en cierto nivel de detalle dependiendo de las necesidades específicas de cada organización y de los fines que se persiguen al almacenar todos estos datos en una base de datos.

Disponer de una base de datos significa el poder tener cantidades grandes de información almacenada de manera organizada para fines administrativos.

De esta forma una base de datos se puede dividir en base de datos planas cuando solo se utiliza una tabla y relacionales que son las bases de datos organizadas mediante distintas tablas relacionadas por campos comunes , las cuales nos dan la ventaja de tener mayor consistencia en los datos y menos redundancia en ellos.

Un punto importante de mencionar en cuanto a las bases de datos, es que éstas nos proveen una manera de tener nuestra información centralizada.

Podemos dividir a un sistema de bases de datos en dos partes: El sistema de manejo de bases de datos (DBMS), el cual es el programa dedicado a la administracion y manejo de la base de datos y el cual describimos a fondo más adelante. El otro elemento es la aplicación de Bases de

[14] Diseño de Bases de Datos. Wiederhold Gio. Stanford University Ed. McGrawHill Pag.1,14

Datos la cual va a estar encargada de otorgar un medio por el cual el usuario o administrador de bases de datos pueda visualizar los datos de una manera u otra o visualizar reportes de la información almacenada por el DBMS, así como también actualizarla.

Estos dos elementos del sistema de bases de datos comunmente residen en la misma computadora o sistema, sin que ello sea una reestricción, ya que actualmente contamos con toda una tecnología de bases de datos que nos permiten tener al DBMS corriendo en una máquina y la aplicación de bases de datos en otra distinta, e inclusive podemos contar con varias máquinas corriendo la misma aplicación haciendo peticiones a través de la red a un mismo DBMS o a varios residiendo en distintas máquinas. Aquí es donde encontramos un nuevo concepto : la tecnología de bases de datos **cliente/servidor**, la cual se describe más adelante.

Las bases de datos no son un elemento nuevo que surja con el modernismo, las computadoras o la tecnología electrónica, por el contrario las bases de datos han existido desde el momento en que se agrupa gran cantidad de información que tiene relación entre sí, no importando si se encuentra en medios electrónicos o en simple papel, la base de datos es la información en si con una estructura definida, lo que cambia es la forma en que es manejada y almacenada.

2.1.2 Definición de DBMS

DBMS o bien "Sistema Manejador de Bases de Datos" , es un programa que nos va a facilitar la manipulación de una o varias bases de datos de manera transparente y simple, de tal modo que no tengamos que preocuparnos por "detalles" como la manera de accesar esa información o los procedimientos exactos por los cuales podemos obtener un dato u otro, sólo nos preocuparemos por hacerle una petición a este programa, en un lenguaje que él entienda, para que realice las operaciones necesarias que resulten en la lectura de un dato en específico o su procesamiento y escritura.

Un DBMS no debe ser confundido con una base de datos, puesto que éste es el manejador de esa base datos.

En una base de datos la información no solamente va a ser almacenada, se pretende adicionar más datos a ésta o actualizarlos, ordenarlos, visualizar parte de estos datos, etc. Todas estas operaciones las podemos realizar a través del DBMS que va a ser el encargado directo de realizarlas cuando el usuario se lo requiera o la aplicación de base de datos se lo indique.

Podemos mencionar los servicios que un DBMS va a otorgar:

- Definición de datos .

En una base de datos vamos a tener toda una estructura para poder almacenar la información. Esta información, como lo habíamos mencionado, va a describir de alguna manera algun objeto. Los datos almacenados que describen a este objeto deben de seguir cierta estructura y deben de estar bien definidos.

Retomando el ejemplo de la fábrica de plastico, podemos mencionar que las características de un cierto objeto plástico que es producido en esta fábrica, son almacenadas en "campos" de un "registro", siendo el registro el conjunto de campos que definen a un objeto, en este caso un objeto de plástico.

Para cada característica de nuestro objeto podemos definir un campo en el cual se almecene una característica en especial. Cada campo debe de tener un tipo, es decir, cada dato debe estar clasificado para tener la información bien clasificada y ordenada.

La definición de datos se refiere ciertamente al requerimiento de establecer el tipo de dato que se almacenará en un campo de un registro.

- Mantenimiento de Datos.

El DBMS debe de contar con un mecanismo que le permita mantener los datos en el disco y encontrar alguno cuando se le haga una petición. A este mecanismo se le refiere como el Mantenimiento de Datos.

- Manipulación de Datos.

Como habíamos ya mencionado, en una base de datos deben de existir los procesos adecuados para insertar nuevos datos, actualizarlos, borrarlos, obtener un dato en específico, etc., es decir, un DBMS nos debe proveer de un mecanismo para manipular los datos contenidos en la base de datos.

- Despliegue de los datos.

También es necesario contar con un servicio que permita la visualización de datos así como del resultado de las operaciones que sobre ellos se realicen. El DBMS debe de permitir este tipo de servicios, si no se cuenta con ellos, entonces la aplicación es quien provee este servicio, si este es el caso , el DBMS es conocido como un "motor de bases de datos".

- Integridad de Datos.

EL DBMS debe de proveer un mecanismo que permita tener confiabilidad de los datos, es decir, que se tenga la certeza que los datos almacenados no esten corruptos debido a su constante manejo asi como a condiciones ambientales como fallas en la energía y otros.

EL DBMS debe proteger los datos de actualizaciones simultáneas, como es el caso en las bases de datos multiusuario en donde varios usuarios pueden en un mismo tiempo hacer modificaciones a los datos almacenados, en estos casos el DBMS debe controlar que solo una actualización en un tiempo tome efecto y notificar al usuario del cambio realizado.

Los datos que son insertados en un campo deben de ser adecuados para ser almacenados en éste o deben de seguir un formato específico. Si se toma como ejemplo una base de datos de los alumnos de una escuela cuyos registros esten conformados por varios campos que definen los datos generales del estudiante y tratáramos de insertar un nuevo estudiante, el campo denominado "nombre" deberá contener caracteres alfabéticos únicamente puesto que es casi imposible que exista un nombre de persona que contenga algun número o caracter especial como una coma o un signo de pesos.

El DBMS puede ser el encargado de validar este tipo de detalles, o bien, podemos dejar esta tarea a la aplicación, la cual a través de programación, validará los datos y los mandara ya correctos al DBMS, este último ya no será el encargado de realizar ese trabajo.

Podemos tener una base de datos en archivos, pero no contaríamos con un DBMS por lo que el manejo de la misma sería extremadamente laborioso y nos haria dependientes del código, por lo que una pequeña modificación en la estructura de los archivos que conforman a la base de datos nos ocasionaría la modificación de varias rutinas y código en general si queremos seguir manteniendo la funcionalidad de nuestra base de datos o nuestro sistema.

Un DBMS, a diferencia del esquema de archivos, puede entregarnos el dato que necesitemos sin necesidad de leer la base de datos completa o el registro entero, es posible indicarle al DBMS qué dato requerimos, que lo procese de cierta manera, que realice alguna operación con un conjunto de datos y que nos entregue el resultado o bien que lo escriba en algún otro registro, etc.

Cuando utilizamos el esquema de archivos, el manejo y procesamiento de la información se vuelve más complicado puesto que el programador o el desarrollador debe de tomar en cuenta todo lo anterior y además usualmente debe de realizar rutinas especiales para tratar ciertos tipos de datos especiales como son las fechas y los datos monetarios los cuales debe de convertir hacia algún formato más digerible, procesarlo, convertirlo de nuevo a su forma original y entregar el resultado, eso sin contar que previo a esto debió de haber solucionado todos los pormenores de lectura y obtención de la información necesaria para llegar a un resultado final.

Un DBMS también tiene herramientas que nos facilitan la creación de reportes y la agrupación de datos para visualizarlos de una manera útil a nuestros fines, en términos generales un DBMS realiza muchas de las tareas de manejo de un a base de datos por nosotros, de tal modo que el Administrador de Bases de Datos (**DBA**), sólo se ocupe de hacerle las peticiones necesarias para obtener los datos requeridos.

2.1.3 Modelos de DBMS

Los modelos de DBMS hacen referencia a la manera de cómo están distribuidos los datos en la base de datos para ser accesados.

Se dice también que un modelo de sistema de manejo de bases de datos es una descripción conceptual de cómo trabaja una base de datos, además de que por medio de el se puede visualizar la relación que existe entre varios datos.

Los principales modelos de DBMS son los siguientes :

- Modelo jerárquico

En este modelo los datos están distribuidos en forma de árbol, los cuales se originan de una raíz y se desprenden por niveles, así una estructura de datos comprendida en este modelo es llamado nodo del cual se pueden desprender más ramas. Si de un nodo no se desprenden más ramas, entonces este nodo es llamado hoja.

Con este modelos podemos ver las relaciones de padre-hijo entre todos los objetos de nuestra base de datos.

El acceso a los datos en este modelo es de manera ordenada y rápida puesto que si es necesario extraer un dato específico, éste va a ser descompuesto en sus componentes para encontrar la ruta por la cual dirigir su búsqueda a través de la estructura de árbol. También con este tipo de modelo es posible establecer índices para realizar los accesos aún más rápidamente.

Una de las principales desventajas de este modelo es la incapacidad de representar la relación muchos a muchos entre varios objetos. Por ejemplo supongamos que tenemos una base de datos de distribuidores de componentes eléctricos, si consideramos que este modelo podemos verlo como una estructura de árbol invertido en donde cada una de las ramas de este árbol penden de un *root* o raíz y cada una de éstas representa cada uno de los distribuidores de los cuales a su vez penden más ramas que representan los productos que comercializa y así también pueden existir más ramas para los modelos específicos de cada tipo de componente eléctrico.

Tomando en cuenta todo lo anterior, resultaría complicado el tratar de represntar una base de datos donde más de un distribuidor comercializa un mismo producto y donde un solo distribuidor comercializa varios tipos de productos, es decir, donde exista una relación muchos distibuidores a muchos productos.

Tratando de representar la situación anterior tendríamos ramas en nuestra estructura de árbol de

las cuales pendieran hojas o inclusive ramas iguales, es decir, tendríamos la misma información en varios lugares distintos, duplicando y complicando aún más la estructura de nuestra base de datos y por lo tanto la administración de ésta.

También resulta complicado el representar relaciones cruzadas en donde, si tenemos una base de datos de empleados, podamos representar la situación en la cual un empleado también sea un jefe de departamento y en la estructura exista la representación lógica de una relación de padre-hijo entre el jefe de departamento y él mismo.

En sí la principal desventaja de este modelo es lo estático de su representación.

- Modelo de Red

El nombre de este modelo de DBMS no tienen que ver con el medio en el cual corre, sino con la propiedad de poder representar a través de él, las relaciones de muchos a muchos entre los objetos de la base de datos.

Con el modelo de red las relaciones que existen entre los diferentes objetos de la base de datos ya no son referidas como padre-hijo, sino que se integra un concepto nuevo en donde las relaciones son conocidas como "sets" o grupos.

2.1.4 SQL Server de Sybase[15]

Deficinición.

SQL Server de **Sybase** es un Sistema Manejador de Bases de Datos (DBMS) relacional que tiene como caracteristicas principales las siguientes:

- Concurrencia, esto es, manejo de múltiples bases de datos, así como múltiples usuarios
- Protección completo de la integridad de la base de datos, a través de mecanismos de seguridad, manejo de transacciones, **triggers**, etc.
- Soporte de la arquitectura Cliente-Servidor.
- Utilización del SQL como vía de consulta y definición de datos a través del Transact-SQL.
- Posee un optimizador de consultas a las bases de datos.

Sybase SQL Server acepta la petición de algún tipo de servicio de un cliente, procesa la petición y regresa el resultado, para que los clientes interactuen con Sybase SQL Server sólo es necesario que conozca el método de emisión de una petición de cualquier tipo e interpretar una respuesta o contestación.

[15] Notas del Curso de SQL. Departamento de Innovación

2.1.5 Características principales :

- Sybase SQL Server hace el manejo de los datos y la memoria.
- Maneja múltiples bases de datos y múltiples usuarios.
- Mantiene el seguimiento de los datos en los discos.
- Realiza el mapeo lógico de la descripción de los datos a un almacenamiento físico de los datos.
- Mantiene los datos y los procedimientos cache en memoria.
- Compila y ejecuta sentencias SQL en procesos **batch**.
- Muestra los resultados a los clientes.
- Optimiza automáticamente las tareas en la base de datos.
- Arquitectura multi-hilos/proceso
- Maneja por si mismo múltiples usuarios.
- No depende del sistema operativo del servidor para realizar multitareas.
- Mejora el desempeño reduciendo la sobrecarga al S.O.

2.1.6 Strucured Query Lenguage (SQL)

El **SQL** (Strucured Query Lenguage) es un lenguaje para consultar y definir datos en un sistema manejador de bases de datos relacional(RDBMS). Entre las pricipales características están las siguientes:

- Lenguaje semánticamente fácil de entender, ya que las sentencias parecen sencillas frases en inglés.
- SQL permite en sus instrucciones manejar un conjunto de registros en lugar de un registro a la vez
- Identifica y utiliza el método más eficiente de búsqueda de los datos solicitados.

Los componentes del SQL son:

El Lenguaje de Definición de Datos(**DDL**), el cual permite crear, modificar y eliminar estructuras de datos como: tablas, bases de datos, indices, etc.

El Lenguaje de Manipulación de Datos(**DML**), el cual permite insertar, modificar y eliminar datos de la base de datos

El Lenguaje de Control de Datos (**DCL**), el cual permite establecer los privilegios de acceso a los datos, esto es, establece la seguridad de la base de datos.

2.1.7 Transact-SQL

Sybase SQL Server utiliza una variante del SQL, el cual cumple con el estándar SQL (ANSI en 1989). Contiene algunas adiciones al estandar:

- Estructuras de control de flujo de progranas, declaración de variables, asignación de parámetros, etc. . Permite crear procedimientos almacenados y triggers.
- Permite la definición de tipos de datos por parte del usuario.
- Amplia la función de la instrucción SELECT.
- Agrega una gran variedad de funciones para el manejo de datos tipo carácter, fecha y numéricos, las contenidas en el estándar SQL.

2.1.8 Elementos de un DBMS (Estructuras de Datos) .

Tablas: Una tabla es una estructura de almacenamiento de un **DBMS** en el cual se almacena la información y consiste en una o mas columnas y cero o mas registros.

Default: Especifica el dato que será utilizado, si el usuario no lo especifica en la inserción o en la actualización de un registro.

Regla: Especifica lo que se permitirá y lo que no se permitirá insertar en una columna.

Estos dos últimos elementos se pueden asociar a una columna en particular, a un número de columnas o a todas las columnas de la base de datos si lo asociamos a un tipo de dato definido por el usuario, el cual se utilizará en las tablas de la base de datos.

Indices: Es una estructura de almacenamiento físico y que ocupa un espacio. Los índices ayudan al Servidor SQL a localizar datos y son transpartentes para el usuario. Su propósito general es proporcionar un acceso más rápido a los datos, aunque en algunos casos su propósito es asegurar que el contenido de un campo sea único.

Permiten que un DBMS tenga acceso a datos más rápido a los datos para las consultas, se utilizan para la optimización de las consultas. El sistema crea esta estructura de datos interna que realiza la selección de filas, cuando la selección se basa en columnas puestas en un índice.

Sybase soporta 2 tipos de Indices:

1. Clustered: Ordena las paginas de los datos mediante la definición del índice y solo puede haber un índice de este tipo.
2. Nonclustered: Estos no ordenan físicamente los datos, sino que mantienen apuntadores o ligas a los registros (pueden existir hasta 249 índices nonclustered por tabla).

Llaves Primarias: Una o más columnas que identifican a un registro, el cual debe ser único y no puede ser nulo.

Llaves Foráneas: Una o más columnas que hacen referencia a la llave primaria de otra tabla.

Estas llaves, se utilizan solo para documentación, de manera que los usuarios tengan referencia de que existen. Los procedimientos del sistema utilizados para definir las llaves son: sp_primarykey y sp_foreingkey.

2.1.9 Modelo Entidad – Relación

Para poder entender de una manera más detallada este concepto, se mencionan a continuación las siguientes definiciones:[16]

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

Modelo de Datos: Es una colección de herramientas conceptuales para describir los datos, las

[16] URL:www.itlp.edu.mx/publica/tutoriales/basedat1/tema1_4.htm

relaciones que existen entre ellos, semántica asociada a los datos y restricciones de consistencia.

Los modelos de datos se dividen en tres grupos:

Modelos lógicos basados en objetos.
Modelos lógicos basados en registros.
Modelos físicos de datos

Modelos lógicos basados en objetos

Se usan para describir datos en los niveles conceptual y de visión, es decir, con este modelo representamos los datos de tal forma como nosotros los captamos en el mundo real, tienen una capacidad de estructuración bastante flexible y permiten especificar restricciones de datos explícitamente.

Entidad: es un objeto que existe y es distinguible de otros objetos. [17]
Las entidades pueden ser:

- concreta, como una persona o un libro
- abstracta, como un concepto o una fecha

Conjunto de entidades: grupo de entidades del mismo tipo, como cuentas, o clientes, etc.
Los conjuntos de entidades pueden no ser disjuntos. Entidad cliente, entidad empleado, y entidad persona que puede ser a su vez cliente, y/o cliente, o ninguna.

Una entidad está representada por un conjunto de atributos, tales como nombre, calle, ciudad, etc.

Dominio: conjunto de valores permitidos para cada atributo. Enteros positivos, cadenas de caracteres, etc.
Atributo: una función que asigna un conjunto de entidades a un dominio.

Cada entidad se describe por medio de un conjunto de pares (atributo, valor del dato).

Tablas: Es la forma de estructurar los datos por medio de filas o registros y columnas o atributos

Llaves: Por definición, las entidades deben poder distinguirse entre sí. Desde el punto de vista de la Base de Datos, son los atributos lo que harán que las entidades puedan distinguirse.

Relación: es una asociación entre varias entidades.

Conjunto de relaciones: grupo de relaciones del mismo tipo. Es una relación matemática de n>=2 conjuntos de entidades (que pueden no ser distintos).
Si E1, E2, ..., En son conjuntos de entidades, entonces un conjunto de relaciones R es un subconjunto de:

{(e1, e2, ..., en) | e1 $ E1, e2 $ E2, ..., en $ En }

Por lo tanto :

[17] URL:usa.ethek.com/basedatos/intro.htm

Para identificar las entidades dentro del sistema debe conocerse el funcionamiento del sistema en estudio, a través de estudios de usuarios, de necesidades de información, de tipos de información, etc.

Para determinar las claves o identificadores de entidades se deben señalar aquellos atributos que identifiquen inequívocamente cada ocurrencia de la entidad.

Al establecer las relaciones entre la entidades, describiendo el grado de las mismas se tienen que estudiar las asociaciones entre las entidades, para definir su importancia dentro del contexto del sistema. Y de esta manera dibujar el modelo de datos, o sea, representar gráficamente el modelo obtenido. Identificar y verificar esto es, la eliminación de las relaciones redundantes y que puedan ser obtenidas a través de combinar otras asociaciones así como describir los atributos de cada entidad esto es, señalar aquellas propiedades de la entidad de interés.

Pero para entender mejor el concepto, se tiene el siguiente ejemplo:

Consideremos una empresa que requiere controlar a los vendedores y las ventas que ellos realizan; de este problema determinamos que los objetos o entidades principales a estudiar son el empleado (vendedor) y el artículo (que es el producto en venta), y las características que los identifican son:

Empleado:	*Artículo:*
Nombre	Descripción
Puesto	Costo
Salario	Clave
R.F.C.	

La relación entre ambas entidades la podemos establecer como Venta.

Por último faltaría representar esta relación de forma gráfica, o sea, crear el modelo E-R gráficamente.

Tablas relacionales[18]

Son tablas que cumplen los siguientes requisitos:

- Cada fila debe ser única, es decir no pueden existir filas duplicadas.
- Cada columna debe ser única
- Los valores de las columnas deben pertenecer al dominio de cada atributo
- Debe tener un solo tipo de fila, cuyo formato está definido por el esquema de tabla o la relación.
- El valor de la columna para cada fila debe ser único.
- No puede contener columnas duplicadas.

Modelos lógicos basados en registros.

Se utilizan para describir datos en los niveles conceptual y físico. Estos modelos utilizan registros e instancias para representar la realidad, así como las

[18] URL: kiyen.face.ubiobio.cl/~cvidal/base_datos...nal/tsld040.htm

relaciones que existen entre estos registros (ligas) o apuntadores. A diferencia de los modelos de datos basados en objetos, se usan para especificar la estructura lógica global de la base de datos y para proporcionar una descripción a nivel más alto de la implementación.

Los tres modelos de datos más ampliamente aceptados son:

Modelo Relacional
Modelo de Red
Modelo Jerárquico

Modelos físicos de datos.

Se usan para describir a los datos en el nivel más bajo, aunque existen muy pocos modelos de este tipo, básicamente capturan aspectos de la implementación de los sistemas de base de datos. Existen dos clasificaciones de este tipo que son:

Modelo unificador
Memoria de elementos

2.1.10 Arquitectura Cliente-Servidor (Historia)

Algunos años atrás, aún cuando en aquel momento se hablaba mucho y se hacía muy poco sobre el tema, se decía que el desarrollo de aplicaciones Cliente / Servidor era inevitable por varias razones:

Existían ya en ese momento servidores razonablemente eficientes y confiables, se había establecido un estándar de hecho para una interfaz Cliente / Servidor: el **ODBC** SQL, adoptado por todos los fabricantes importantes de servidores.

Los primeros trabajos conocidos para la arquitectura Cliente / Servidor los hizo Sybase, que se fundó en 1984 pensando en lanzar al mercado únicamente productos para esta arquitectura. A fines de la década de los 80's el producto fue lanzado, para el voluminoso segmento "low-end" del mercado, en conjunción con Microsoft, teniendo como soporte de la base de datos un servidor OS/2, y como herramienta "front end" básica el Dbase IV de Ashton Tate. El Dbase IV no se mostró como una herramienta adecuada, y los desencuentros comerciales entre Sybase, Microsoft e IBM (en aquel momento socia de Microsoft para el OS/2) hicieron el resto.

En la última década una cantidad muy importante de empresas, en todo el mundo, ha encarado aplicaciones Cliente / Servidor, y que las que lo están haciendo con los planes necesarios y con las herramientas adecuadas, están obteniendo éxitos muy importantes, mientras que los que lo han hecho desaprensivamente, han cosechado fracasos.

Algunas definiciones para esta arquitectura están las siguientes:

•desde un punto de vista conceptual:

"A grandes rasgos es un modelo para construir sistemas de información que se sustenta en la idea de repartir el tratamiento de la información y los datos por todo el sistema informático, permitiendo mejorar el rendimiento del sistema global de información"

•esto nos lleva a una definición en términos de arquitectura:

"Los distintos aspectos que caracterizan a una aplicación (proceso, almacenamiento, control y operaciones de entrada y salida), en el sentido más amplio, están situados en más de una computadora, los cuales están interconectados mediante una red de comunicaciones."

"Una arquitectura es un grupo de componentes funcionales que aprovechando diferentes estándares, convenciones, reglas y procesos, soporta la integración de una amplia gama de productos y servicios informáticos de manera que pueden ser utilizados eficazmente dentro de la Organización."

Los principales componentes del esquema Cliente/Servidor son entonces los Clientes, los Servidores y la infraestructura de comunicaciones. Los Clientes interactúan con el usuario, usualmente en forma gráfica. Frecuentemente se comunican con procesos auxiliares que se encargan de establecer conexión con el servidor, enviar el pedido, recibir la respuesta, manejar las fallas y realizar actividades de sincronización y de seguridad.

Los Servidores proporcionan un servicio al cliente y devuelven los resultados. En algunos casos existen procesos auxiliares que se encargan de recibir las solicitudes del cliente, verificar la protección, activar un proceso servidor para satisfacer el pedido, recibir su respuesta y enviarla al cliente. Además deben manejar los inter bloqueos, la recuperación ante fallas, y otros aspectos afines. Por las razones anteriores la plataforma computacional asociada con los servidores es más poderosa que la de los clientes. Por esta razón se utilizan PC's poderosas, estaciones de trabajo, mini computadoras o sistemas grandes. Además deben manejar servicios como administración de la red, mensajes, control y administración de la entrada al sistema ("login"), auditoría y recuperación y contabilidad. Usualmente en los servidores existe algún tipo de servicio de bases de datos.

Para que los clientes y los servidores puedan comunicarse se requiere una infraestructura de comunicaciones, que es la red, la cual proporciona los mecanismos básicos de direccionamiento y transporte. La mayoría de los sistemas Cliente/Servidor actuales se basan en redes locales y por lo tanto utilizan protocolos no orientados a conexión, lo cual implica que las aplicaciones deben hacer las verificaciones. La red debe tener características adecuadas de desempeño, confiabilidad, transparencia y administración.

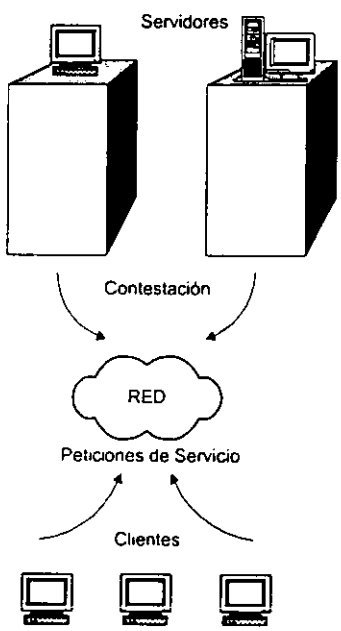


Fig. 2.1 Aplicaciones Cliente/Servidor

2.2 Análisis de la base de datos

Diccionario de Datos.

Tablas del Sistema de Administración de Cuentas de Administración de cuentas de correo electrónico de Red UNAM

Usuarios

Usuarios del Sistema de Administración de Cuentas

Nombre	Tipo	N/NN	PK/FK	Descripción
US_ID	int	NN	Si (PK)	Clave de Identificación del Usuario.
TIP_ID	tinyint	NN	Si(FK)	Identificador del tipo de servicio.
DEP_ID	int	NN	Si(FK)	Identificador de la Dependencia.
NOMBRE	char(16)	NN	No	Nombre del Usuario
PATERNO	char(16)	NN	No	Apellido Paterno del Usuario.
MATERNO	char(16)	NN	No	Apellido Materno del Usuario.
RFC	char(13)	NN	No	RFC del Usuario.
DIRECCION1	char(32)	NN	No	Dirección (1) del Usuario.
DIRECCION2	char(32)	NN	No	Dirección (2) del Usuario.
CP	char(5)	N	No	Código Postal del Usuario.
CIUDAD	char(32)	N	No	Ciudad del Usuario.
TEL_NUM	char(25)	N	No	Número de Teléfono del Usuario.
TEL_DESC	char(25)	N	No	Descripción del Teléfono.
TITULO	char(12)	N	No	Título del Usuario.
PUESTO	char(32)	N	No	Puesto del Usuario.
E_MAIL	char(64)	N	No	E-mail del Usuario.

Claves

Claves de los Usuarios del Sistema de Administración de Cuentas.

Nombre	Tipo	N/NN	PK/FK	Descripción
CLV_ID	int	NN	Si (PK)	Identificador de la Clave del Usuario.
US_ID	int	NN	Si (FK)	Clave de Identificación del Usuario.
PAG_ID	int	NN	Si(FK)	Identificador de Pago.
DIR_ID	int	NN	Si(FK)	Identificador del Directorio.
TIP_ID	tinyint	NN	Si(FK)	Identificador del tipo de servicio.
DEP_ID	int	NN	Si(FK)	Identificador de la Dependencia.
EST_ID	tinyint	NN	Si(FK)	Identificador del Estado de la cuenta (0 - 1).

SER_ID	tinyint	NN	Si(FK)	Identificador del tipo de Servicio. (0 - 1).
NOMBRE	char(8)	NN	No	Nombre de la clave del Usuario.
PASSWORD	char(15)	NN	No	Password de la clave del usuario.
INICIO	datetime	NN	No	Fecha de Inicio de activación de la clave.
MODIFICA	datetime	NN	No	Fecha de Modificación de la clave.
HORAS	int	N	No	Número de Horas
VENCE	datetime	N	No	Fecha de termino de la clave.
FORWARD	char(32)	N	No	Dirección alterna de correo electrónico.
DESCRIPCIO N	char(48)	N	No	Comentario adicional e la clave.

Pagos

Información de las formas y montos de los pagos para las claves de cuentas de correo electrónico.

PAG_ID	int	NN	Si (PK)	Identificador de Pago.
CLV_ID	int	NN	Si (FK)	Identificador de la Clave del Usuario.
FECHA	datetime	NN	No	Fecha en que se realizó el pago.
MONTO	money	NN	No	Monto del pago.
RECIBO	char(32)	NN	No	Número de Recibo.
TIPO_PAGO	char(32)	NN	No	Descripción del tipo de pago.

Tipo_clv

Información del tipo de claves.

TIP_ID	tinyint	NN	Si(PK)	Identificador del tipo de servicio. 0-9
TIPO	char(32)	NN	No	Descripción del tipo de clave 0 - Cortesía. 1- Becario o Servicio Social. 2- Estudiante. 3- Investigador. 4- Académico Tiempo completo. 5- Académico Asignatura. 6- Funcionario. 7- Particular. 8- No lucrativa. 9- Administrativa.

Servicios.

Información del tipo de servicio de la clave.

SER_ID	tinyint	NN	Si(PK)	Identificador del tipo de Servicio. 0-1
SERVICIO	char(32)	NN	No	Descripción del tipo de servicio de la clave. 0 - Vía Módem. 1- Correo Electrónico..

Directorios

Información de los directorios donde quedará alojada la clave de correo electrónico.

DIR_ID	int	NN	Si(PK)	Identificador del Directorio. 0 - 13
Directorio	char(25)	NN	No	Nombre del Directorio. 0 - /usr/users00 1- /usr/users01 2- /usr/users02 3- /usr/users03 4- /usr/users04 5- /usr/users05 6- /usr/users06 7- /usr/users07 8- /usr/users08 9- /usr/users09 10- /usr/users10 11- /usr/users11 12- /usr/users12 13- /usr/users13

Estados

Información del estado de la clave de correo electrónico.

EST_ID	tinyint	NN	Si(PK)	Identificador del Estado de la cuenta 0 - 1.
ESTADO	char(20)	NN	No	Descripción del Estado de la clave 0 - Activa. 1 - Suspendida.

Tipo_dep

Información del Tipo de Dependencia a la que pertenece la clave de correo.

TIP_ID	tinyint	NN	Si(PK)	Identificador del tipo de servicio. 0 - 4 .
TIPO	char(50)	NN	No	Descripción de la

				Dependencia. 0 - Interna 1 - Externa 2 -.Lucrativa 3 - No lucrativa 4 - Particular.

Dependencias

Información de la Dependencia a la que pertenece la clave de correo.

DEP_ID	int	NN	Si(PK)	Identificador de la Dependencia.
TIP_ID	tinyint	NN	Si(FK)	Identificador del tipo de servicio. 0 - 4 .
NOMBRE	char(80)	NN	No	Nombre de la Dependencia.

2.3 Diseño de la base de datos

Diagrama Entidad Relación del Sistema de Administración de Cuentas

A continuación se describe el módelo entidad-relación de la base de datos del sistema de administración de cuentas de correo electrónico de REDUNAM, en el cual se muestra la estructura con la que se compone y la relación que hay entre cada tabla que la conforman.

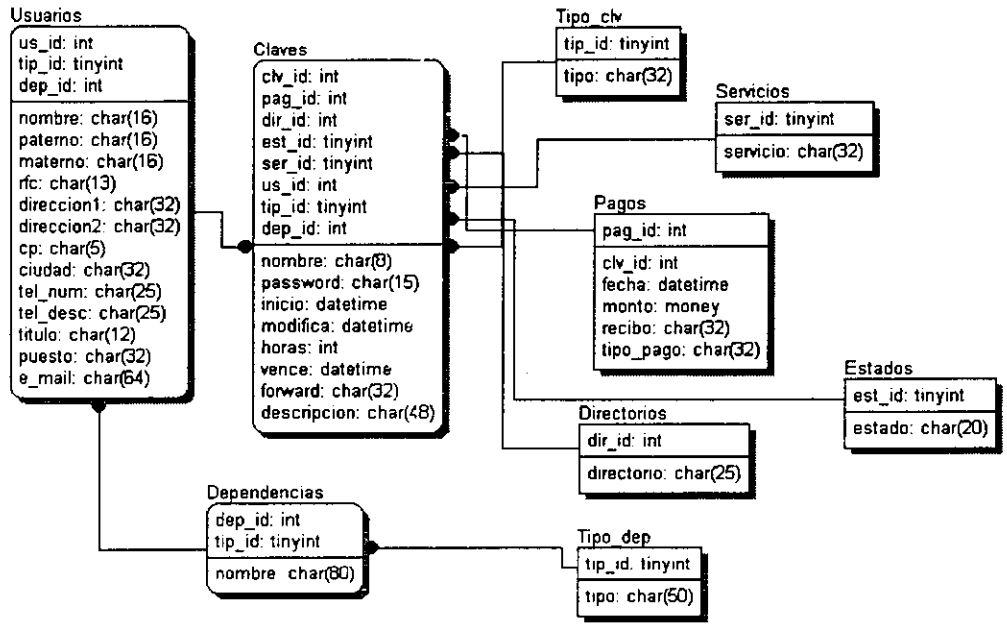


Fig. 2.2 Diagrama Entidad-Relación

Tabla de Usuarios: Será la encargada de almacenar los datos principales de los usuarios (como su nombre lo dice), de los cuales , la clave de identificación del usuario (US_ID) será única para cada usuario y tendrá relación con la tabla de Claves y de esta forma se sabrá que clave es para cada usuario, ya que en esta tabla (Usuarios) están el nombre y apellidos del usuario, su dirección de residencia así como otros datos personales de identificación.
Tabla de Claves: En esta tabla se almacenarán las claves de cada usuario, con lo cual se tendrá el control de las mismas y la información referente a ellas, como lo son principalmente : los identificadores del estado de la cuenta (activa o cancelada), el nombre del usuario, identificador del tipo de dependencia a la que corresponde, fechas de inicio de activación o de alguna modificación de la misma y comentarios acerca de la clave.
Tabla de Pagos: Tabla que registrará la información de formas de pago y sus montos para las claves de cuentas de correo electrónico. Su función también abarca las fechas de realización del pago, así como la descripción del tipo de pago.

Tabla Tipo_clv (tipo clave) : Para el control del tipo de clave para donde será destinada , realizando esto a traves de un identificador y su correspondiente descripción.

Tabla de Servicios: Tabla de información del tipo.de servicio para la clave. Esta base está diseñada para la incorporación del servicio de módem, de esta manera, en esta tabla se controlará el tipo de servicio que el usuario requerirá a través de un identificador de servicio.

Tabla de Directorios: Tabla para el control de la estructura de los directorios donde se alojará el "Home" del usuario a través de un número identificador.

Tabla de Estados: Información del estado de la clave de correo electrónico que a través de un identificador de estado, se diferenciará una cuenta activa o suspendida.

Tabla de Tipo_dep: Para la información del tipo de dependencia a la que corresponde la clave de correo electrónico.

Tabla de Dependencias: Para la información del tipo de dependencia a la que corresponde la clave de correo electrónico además del nombre de la Dependencia.

2.4 Migración de la información.

A continuación se muestran los procedimientos para tener lista la Base de Datos que atenderá al Sistema de Administración de Cuentas de Correo Electrónico de RED UNAM:

Para crear las tablas que requerirá el sistema para su correcto funcionamiento, se elaboró un **script** que cumple con lo especificado en el diagrama entidad relación de la base de datos de SAC (ver **Capítulo 5**).

Ya creadas las tablas y la estructura relacional de la base de datos de SAC se procede a realizar la migración de la información de una base de datos existente que contenía registros que sirvieron para una más rápida actualización de datos para la nueva base de datos. Esta migración se realizó por medio de los siguientes scripts :

sp_claves : Procedimiento de inserción de claves : Este procedimiento inserta las claves de correo de la base de datos existente a la base de datos nueva.

```
create proc sp_claves
as
declare @cve char (15), @#maquina tinyint, @fec_ini datetime
declare @fec_ven datetime, @horas_restan int
declare claves_crsr cursor for
select DA.cve, DA.#maquina, CVE.fec_ini, CVE.fec_ven, CVE.horas_restan from DATO
S DA, CVE_CORREO_TROUTER CVE
open claves_crsr
fetch claves_crsr into @cve, @#maquina, @fec_ini, @fec_ven, @horas_restan
select @clv_id=1
while @@sqlstatus=0
begin
    insert claves select distinct @clv_id, 0, 0, 0, 0, convert(tinyint,DA.@#
maquina), DA.@cve, "DESCONOCIDO", CVE.@fec_ini, Getdate(), CVE.@horas_restan, CV
E.@fec_ven, NULL, NULL from DATOS DA, CVE_CORREO_TROUTER CVE where DA.cve = CVE.
cve and DA.#maquina = CVE.#maquina
    print "Registro %1! copiado %2!",@clv_id,@cve
    select @clv_id=@clv_id+1
    fetch claves_crsr into @cve, @#maquina, @fec_ini, @fec_ven, @horas_restan
end
close claves_crsr
deallocate cursor claves_crsr
return
```

sp_dependencias: Procedimiento de inserción de dependencias. Este procedimiento inserta las dependencias de la base de datos existente a la base de datos nuevas

```
create proc sp_dependencias
as
declare @num_dep int, @nom_depen char(80),@dep_id int
declare dep_crsr cursor for
select dep_id, nombre from prueba
open dep_crsr
fetch dep_crsr into @num_dep, @nom_depen
select @dep_id=1
while @@sqlstatus=0
begin
      insert dependencias select @dep_id,0,nombre from prueba where nombre = @
nom_depen
      print "Registro %1! copiado. %2!",@dep_id,@nom_depen
      select @dep_id=@dep_id+1
      fetch dep_crsr into @num_dep, @nom_depen
end
close dep_crsr
deallocate cursor dep_crsr
return
```

sp_depid: Procedimiento de inserción de claves de dependencias. Este procedimiento inserta las claves de dependencias de la base de datos existente a la base de datos nuevas

```
create procedure sp_depid
as
declare @dep_id int
declare dep_crsr cursor
for select d.dep_id from dependencias d
open dep_crsr
fetch dep_crsr into @dep_id
while @@sqlstatus=0
begin
      update usuarios set dep_id=(select distinct d.dep_id from dependencias d
, prueba p, DATOS DA where d.nombre = p.nombre and p.num_dep =DA.#dependencia)
      print "Registro con dep_id= %1! actualizado. ", @dep_id
      fetch dep_crsr into @dep_id
end
close dep_crsr
deallocate cursor dep_crsr
return
```

sp_usid : Procedimiento de inserción de claves de usuarios. Este procedimiento inserta las claves de usuarios de la base de datos existente a la base de datos nuevas

```
create procedure sp_usid
as
declare @us_id int
declare @nombre char(15),@ap_pat char(15),@ap_mat char(15),@rfc char(13),@cve ch
ar(15)
declare @#dependencia int,@nom_depen char(80)
declare us_crsr cursor
for select nombre, ap_pat, ap_mat, rfc, cve from DATOS
open us_crsr
fetch us_crsr into @nombre, @ap_pat, @ap_mat, @rfc, @cve
while @@sqlstatus=0
begin
      update claves set us_id=(select distinct u.us_id from usuarios u,DATOS D
A where u.nombre = @nombre and u.paterno=@ap_pat and u.materno=@ap_mat and u.rfc
=@rfc ) where nombre=@cve
      print "Registro con us_id %1! y login %2! actualizado. ",@us_id, @cve
      fetch us_crsr into @nombre, @ap_pat, @ap_mat, @rfc, @cve
end
close us_crsr
deallocate cursor us_crsr
return
```

sp_usuarios4 : Procedimiento de inserción de datos de usuarios. Este procedimiento inserta la información de los usuarios de la base de datos existente a la base de datos nuevas

```
create procedure sp_usuarios4
as
  declare @nombre char(15),@ap_pat char(15),@ap_mat char(15),@rfc char(13)
  declare @direccion char(60),@puesto char(30),@telefono char(15)
  declare @#dependencia int,@descrip_usu char(50)
  declare @nombre2 char(15) ,@ap_pat2 char(15),@ap_mat2 char(15),@rfc2 char(13)
  declare @count int,@us_id int
  declare user_crsr cursor for
  select distinct nombre,ap_pat,ap_mat,rfc from DATOS
  declare more_crsr cursor for
  select distinct nombre,ap_pat,ap_mat,rfc,direccion,puesto,telefono,
      #dependencia,descrip_usu from DATOS where nombre=@nombre and
      ap_pat=@ap_pat and ap_mat=@ap_mat and rfc=@rfc
  open user_crsr
    fetch user_crsr into @nombre,@ap_pat,@ap_mat,@rfc
    select @us_id=1
    while @@sqlstatus=0 begin
      select @count=(select count(*) from DATOS where nombre=@nombre and
              ap_pat=@ap_pat and ap_mat=@ap_mat and rfc=@rfc)
      if (select @count)>0 begin
        if @count=1 begin
          insert usuarios select distinct @us_id,0,
            convert(char(16),nombre),convert(char(16),ap_pat),
            convert(char(16),ap_mat),rfc,convert(char(32),direccion),
            NULL,NULL,NULL,convert(char(25),telefono),NULL,NULL,convert(ch
ar(32),puesto),NULL from DATOS
            where nombre=@nombre and ap_pat=@ap_pat and ap_mat=@ap_mat and
 rfc=@rfc
          print "A Reg. %1! copiado",@us_id
        end


        else begin
          open more_crsr
            fetch more_crsr into @nombre2,@ap_pat2,@ap_mat2,@rfc2,
                @direccion,@puesto,@telefono,@#dependencia,
                @descrip_usu
            insert usuarios values(@us_id,0,convert(char(16),@nombre2),con
vert(char(16),@ap_pat2),convert(char(16),@ap_mat2),@rfc2,convert(char(32),@direc
cion),NULL,NULL,NULL,convert(char(25),@telefono),NULL,NULL, convert(char(32),@pu
esto),NULL)
          close more_crsr
          print "B Reg. %1! copiado",@us_id
        end
        select @us_id=@us_id+1
        fetch user_crsr into @nombre,@ap_pat,@ap_mat,@rfc
      end
    end
  close user_crsr
  deallocate cursor more_crsr
  deallocate cursor user_crsr
return
```

Asi, teniendo en cuenta los conceptos y caracteristicas de las base de datos relacionales; en éste capítulo se explicó la forma en que se obtiene el análisis, diseño y construcción de la base de datos que conformará al Sistema de Administracón de Cuentas de REDUNAM.

CAPITULO 3

LA INTERFAZ GRÁFICA

CAPITULO 3

La Interfaz Gráfica

3.1 Conceptos Generales

3.1.1 Internet

Internet es una red mundial de computadoras, integrada por cientos de miles de computadoras que tienen la capacidad de comunicarse entre sí. Esto implica un complejo sistema de comunicaciones que permite conectar una computadora con cualquier otra conectada a la red, sin importar la distancia entre una y otra.

A fines de la década de los 60, el Gobierno de los Estados Unidos crea la ARPA (*Advanced Research Projects Agency*, Agencia de Investigación de Proyectos Avanzados). El objetivo de esta agencia es la investigación de nuevas tecnologías. En 1969, el Departamento de Defensa comisiona a la ARPA la creación de **ARPAnet**, cuya función principal era enlazar todos los sistemas de cómputo militares, y además permitir la investigación en protocolos de red. Este proyecto fue el nacimiento de Internet.

En 1973, se concluye que es necesario un protocolo estandard para la comunicación en red, y es creado el protocolo **TCP/IP**, el cual se sigue utilizando actualmente para en transporte de información en Internet.[19]

El protocolo TCP/IP (*Transmission Control Protocol/Internet Protocol*, Protocolo de Control de Transmisión/Protocolo Internet), es una colección de protocolos que permite el intercambio de información entre diferentes plataformas. Un protocolo es un conjunto de normas que permiten a dos o más computadoras establecer una comunicación en la que se realiza una transferencia de información. El protocolo TCP/IP, entre otras cosas, permite identificar una máquina mediante una dirección (conocida como dirección IP), detectar y corregir errores durante la comunicación, y conseguir que independientemente de la máquina (*PC, MAC, Amiga, Alpha,...*) y del sistema operativo utilizado (*Windows 95, Windows 3.1x, MAC OS, DOS, Unix,...*) todos los equipos conectados a la red puedan entenderse y por tanto comunicarse[20].

En 1983, había sólo 500 computadoras conectadas a Internet. En 1986, la **NSF** (*National Science Foundation*, Fundación Nacional de Ciencia) se une a Internet a través de la red *NSFnet*, la cual conectaba cinco centros de supercómputo con una conexión de 56 *Kbps* y cuyo objetivo era puramente académico. A fines de ese año, había ya 5000 computadoras conectadas a Internet, y NFSnet crece su línea a un *T1* de 1.5 *Mbps*. En 1990, la ARPA termina sus operaciones en Internet y ARPAnet deja de existir, dejando la operación de Internet a la NSFnet. El interés de instituciones académicas y de algunos corporativos, y la fusión con otras redes existentes en otros países, propiciaron el crecimiento masivo de las computadoras conectadas a Internet. En 1992 había ya más de 1,000,000 de nodos, y en 1994 pasaban ya los 4,000,000.

El gran auge de Internet ha permitido pasar los límites militares, académicos y científicos para los que fue creado, llegando a las empresas, instituciones de todo tipo, y por supuesto, al usuario doméstico.

---

[19] Programming JavaScript for Netscape 2.0. Tim Ritchey / New Riders Publishing, Indianapolis, Indiana, USA 1996. Página10

[20] http://businessglobal.com/Formacion%20empresarial/Marketing%20en%20WWW/conceptos_basicos_www/internet_que_es.html

La filosofía de funcionamiento de Internet se basa en el modelo cliente/servidor. Este modelo está formado por tres partes: el cliente, el servidor y la red. Un cliente es una aplicación que frecuentemente corre en la computadora del usuario final. Un servidor es una aplicación que corre en la computadora que proporciona la información o servicio. El cliente puede ser ajustado para que funcione como interfaz entre la computadora en la que corre y el servidor. La red es el medio de comunicación entre el cliente y el servidor.[21]

En Internet existen diversos tipos de servicios, y por lo tanto existe un sinnúmero de servidores, entre los cuales están:

- Servidores de Correo Electrónico. Son computadoras que tienen dos funciones básicas: entregar los mensajes enviados por los usuarios del servidor, y entregar a cada uno de ellos los mensajes recibidos. Su operación es análoga al correo postal.

- Servidores ***Web***. Son equipos que almacenan documentos **HTML**, los cuales son solicitados por clientes Web (comúnmente llamados **navegadores** o ***browsers***). Estos documentos mostrados por el cliente son documentos hipertexto, y pueden o no incluir gráficos, sonidos, animaciones, etc

- Servidores de ***News***. Administran grupos de noticias y se encargan de recibir, enviar y organizar mensajes de cada foro de discusión.

- Servidores ***FTP***. Servidores para la transferencia de archivos remotos.

- Servidor ***DNS***. Estos servidores realizan la traducción de los nombres de dominios a direcciones ***IP*** y viceversa.

3.1.2 WWW

El ***World Wide Web*** (WWW) es uno de los servicios de Internet y es un sistema de información global, interactivo, dinámico, distribuido, gráfico, basado fundamentalmente en documentos llamados páginas, que combinan texto, imágenes, sonido, animaciones y vínculos de **hipertexto**, llamados **hipervínculos.**

La idea en la que se basa el hipertexto es que en lugar de leer un texto siguiendo una estructura rígida y lineal (como un libro), es posible avanzar de un punto a otro fácilmente, obtener más información, regresar al primer punto, brincar hacia otros temas y desplazarse (navegar) por el texto según los intereses que tenga en determinado momento.

Una de las ventajas del WWW y la razón por la que ha llegado a ser uno de los servicios de Internet más populares, es su capacidad para presentar en pantalla tanto texto como gráficos, sonido y vídeo dentro de la misma página. Antes del WWW, el uso de Internet comprendía simples conexiones basadas sólo en texto y para navegar por sus varios servicios se debían utilizar interfaces basadas en menús.

El WWW basa su funcionamiento en el protocolo **HTTP** (*HyperText Transfer Protocol*, Protocolo de Transferencia de Hipertexto), el cual permite la transferencia de hypertexto en la red. Este protocolo permite el uso de otros servicios como FTP, ***gopher, news, telnet, e-mail***[22].

Debido a la gran flexibilidad de este servicio, se ha constituido en uno de los servicios más grandes y más importantes del Internet, al grado de que ha superado a otros medios de comunicación, como la radio y la TV, en términos de aceptación y divulgación entre la gente, ya que sólo le ha tomado algunos años (unos 6 o 7) ser parte esencial en la vida contemporánea.

[21] HTML and CGI Unleashed. John December and Mark Ginsburg. Sams.net Publishing. 1995. Pág. 12

[22] http://webdia.cem.itesm.mx/ac/rtrejo/Interfaz/www.html

3.1.2.1 HTML

El **HTML** (*HyperText Markup Language* , Lenguaje de Marcado de Hipertexto), es un lenguaje que sirve para crear y especificar la estructura (encabezados, listas, párrafos, etc) de documentos hipertexto en el WWW.

El HTML es definido por el **SGML** (*Standard Generalized Mark-Up Language*, Lenguaje Normal Generalizado de Marcado), el cual es una norma internacional (**ISO** 8879) para marcado de textos. Por si mismo, el SGML es un **meta-lenguaje**, es decir, un lenguaje para definir lenguajes. La meta del SGML es dar formato a la información en línea para su eficiente distribución, búsqueda y recuperación electrónica, haciendo énfasis en la estructura del documento, y no en la apariencia.

La **DTD** (*Data Definition Type*, Definición de Tipo de Documento) HTML, con su estructura elemental, está principalmente orientada a describir los elementos estructurales que aparecen en el hipertexto[23].

Un documento HTML consiste principalmente de texto y etiquetas usadas para definir la información de un documento y para marcar su estructura. Los simbolos < y > se usan para formar las etiquetas que delimitan los elementos, los cuales identifican la estructura del documento. Ejemplos de estas etiquetas son <HTML> (la etiqueta inicial de todo documento HTML) y </HTML> (al final del documento), las cuales indican que todo lo que se encuentre entre ellas es un documento HTML. Entre estas dos etiquetas se encuentran varios tipos de elementos, como son el título del documento (definido por las etiquetas <TITLE> </TITLE>), el encabezado (<HEAD> y </HEAD>), y el cuerpo del documento (<BODY> y </BODY>).

3.1.1.2 URL

Para hacer referencia a un documento o recurso disponible en Internet, se hace uso de un **URL**. Un URL (*Uniform Resource Locator*, Localizador de Recursos Uniformes) consiste en una cadena de caracteres que únicamente identifican un recurso. Un URL es como un número de catálogo para un recurso.

El formato básico para la mayoría de los URLs es[24]:

esquema://host:port/path

donde:

esquema es una regla o protocolo para enviar u obtener información, como ftp, http, nntp, gopher, telnet, etc

host es la computadora en la cual residen los recursos

port es un número que identifica el servicio solicitado al servidor; este número es proporcionado si el servicio está instalado en un puerto diferente del establecido para ese servicio.

path es la localización del recurso dentro del *host*

---

[23] http://burn.ucsd.edu/~oli/html-2/htmldef.html

[24] HTML and CGI Unleashed. John December and Mark Ginsburg. Sams.net Publishing. 1995. Pág. 25

3.1.2.3 Navegadores

Para poder observar los documentos hipertexto que se encuentran en el WWW, y algunos otros recursos o servicios de Internet, es necesario contar con una aplicación cliente que sea capaz de interpretar la estructura del documento. Estas aplicaciones son llamadas navegadores o *web* ***browsers***

Los navegadores funcionan de la siguiente forma: una vez obtenido el URL al cual va a conectarse, establece la comunicación con él y obtiene el documento o recurso. Para documentos de hipertexto, lo anterior significa la posibilidad de comunicarse con el servidor utilizando el protocolo **HTTP**. Como en **WWW** también maneja información de servidores FTP y **Gopher**, correo electrónico y demás tareas semejantes, el navegador también debe tener la capacidad para comunicarse en el lenguaje de tales herramientas.

No obstante, el principal uso de un navegador es el despliegue de documentos de WWW. Cada una de las páginas que se cargan en Web representa un documento, escrito en lenguaje HTML, que incluye el texto del documento, una estructura, los vínculos hacia otros documentos, imágenes y otros medios. El navegador se comunica con el servidor de Web en Internet, y abre documentos desde ese servidor. Si el documento es un archivo HTML, lo interpreta según el código HTML contenido en el documento, le da formato y lo despliega. Si el documento contiene imágenes o vínculos hacia otros documentos, también ejerce control sobre esos elementos[25].

Entre los navegadores más conocidos tenemos:

- *Netscape* (http://home.netscape.com)
- *Internet Explorer* (http://www.microsft.com)
- *StarOffice* (http://www.sun.com/staroffice)
- *Mosaic NCSA* (ftp://ftp.ncsa.uiuc.edu/Mosaic)
- *Lynx* (ftp://ftp2.cc.ukans.edu/pub/lynx)
- *MacWeb y WinWeb* (ftp://ftp.einet.net/einet/mac/macweb, ftp://ftp.einet.net/einet/mac/winweb).
- *Cello* (ftp://ftp.law.cornell.edu/pub/LII/Cello/Cello.zip)

3.1.2.4 *CGI*

CGI (*Common Gateway Interface*, Interfaz de Entrada Común), es una especificación para permitir a un servidor de WWW ejecutar un programa externo y devolver los resultados al navegador. Mediante esta interfaz es posible correr cualquier programa disponible en el servidor, de modo que no hay límite para las tareas que se pueden realizar. El programa *CGI* obtiene información del usuario a través de una forma HTML que se despliega en la pantalla de un navegador. Cuando la forma se encuentra llena, el usuario lo envía utilizando un botón. El servidor recibe los datos, los pasa al programa externo y éste devuelve los resultados de la operación, que pueden ser el resultado de un cálculo o de la consulta a una base de datos, para que el servidor los regrese a su vez al usuario.

Esta especificación define una serie de reglas que deben cumplir tanto las aplicaciones como los servidores para hacer posible la presentación de resultados de programas ejecutables en tiempo real a través de servicios de información estandarizados. Al tratarse de una interfaz, no existe ninguna dependencia con el lenguaje de programación empleado y, del mismo modo, no presentará mayor dificultad programar un *CGI* en un lenguaje u otro, salvo las complicaciones del propio lenguaje.

En principio, cualquier lenguaje capaz de tener una salida estándar es susceptible de ser utilizado para desarrollar *CGIs*, ya sea interpretado o compilado. En la red existen *CGIs*

[25] http://webdia.cem.itesm.mx/ac/rtrejo/Interfaz/browsers.html

desarrollados *en C, C++, Fortran, Perl , Tcl , Visual Basic, AppleScript* y cualquier ***shell*** de *UNIX.*

Uno de los lenguajes más utilizados en el mundo UNIX es sin duda *Perl.* Es un lenguaje interpretado resultado de una mezcla de *C, sed, awk y sh,* y que se presta muy bien al desarrollo de *CGIs.* El lenguaje C es también muy utilizado en la programación de *CGIs,* ya que tiene una ventaja sobre ***Perl***: la compilación ofrece un mayor nivel de seguridad, ya que permite asegurarse de que el código fuente de un *CGI* no será nunca accesible a la red ni si quiera por error, cosa que no ocurre con un lenguaje interpretado.

Las aplicaciones del *CGI* son numerosas: herramientas de búsqueda, formas de registro, libros de visitas, contadores de accesos, sistemas de bases de datos, grupos de discusión, personalización de páginas, tiendas y catálogos electrónicos, y más. La gran mayoría de las aplicaciones que existen en la *WWW* actualmente, dependen exclusivamente del *CGI*[26].

Un documento *HTML* es estático, permanente, que no se ajusta a las necesidades de acceso a información en constante actualización, consulta a base de datos o seguimiento, control o recuperación de resultados de un determinado proceso. Por ello, a veces es necesario acceder a información que se genera en tiempo real. Así pues, es preciso contar con algún tipo de interfaz como la que define *CGI.* Esta interfaz define una forma cómoda y simple de ejecutar programas que se encuentran en la máquina en la que se aloja el servidor *WWW.* Para el cliente presenta una ventaja en el aspecto de la seguridad, ya que no tendrá que ejecutar ningún programa de efectos desconocidos en su sistema local. El acceso se realiza a través de cualquier cliente de *WWW* y la comunicación se realiza según el protocolo *HTTP.*

Hay que tomar ciertas precauciones al usar *CGIs,* ya que el simple hecho de tener un programa *CGI* implica que cualquier persona que tenga acceso a Internet, puede ejecutarlo. Para empezar, todos los programas *CGI* deben ser almacenados en un directorio exclusivo reservado para ellos, generalmente llamado *cgi-bin,* el cual no debe tener permisos de acceso a los usuarios del sistema. Cuando un servidor WWW recibe como petición un documento alojado en este directorio, entenderá que se trata de un programa ejecutable.

Para ejecutar un *CGI,* la forma más sencilla es utilizando el URL dirigido al archivo ejecutable, pero la más común es que sea invocado desde un documento HTML utilizando una forma (a través de la directiva <FORM>), especificando el URL del programa. Dicho programa debe estar ubicado en el directorio *cgi-bin,* aunque es posible que se sitúe en otro directorio siempre y cuando tenga permisos de ejecución. En este último caso, los archivos deben tener una extensión especial (normalmente *.cgi*) y el servidor *WWW* debe estar configurado para permitir la ejecución desde un directorio distinto a *cgi-bin.*

Las formas (también llamadas formularios) se usan para presentar una interfaz formada por campos de texto, listas, botones y otros elementos que permiten recibir entradas por parte del usuario. Cuando un usuario rellena los campos de una forma y presiona el botón adecuado, la forma envía los datos al servidor para su procesamiento. Por lo regular, el servidor preparará un documento *HTML* utilizando la información suministrada por el usuario y la devolverá a éste para su visualización; no obstante, las aplicaciones de las formas son muy variadas: ordenar un producto, enviar comentarios, hacer una operación bancaria, inscribirse en una base de datos, etc...

Las formas están compuestas por dos partes elementales: un documento *HTML* que contiene a la forma, y un *CGI* que procesará la información que captura la forma. La forma está definida por las directivas <FORM> y </FORM> dentro del documento *HTML.* La directiva <FORM> tiene un atributo llamado ACTION, el cual contiene el *URL* del *CGI* que va a procesar la forma. Entre las directivas <FORM> y </FORM>, se encuentra el cuerpo de la forma, es decir, el texto y los campos de ingreso (listas, botones, cajas de texto, etc), entre los cuales se encuentran los botones para enviar la forma y para limpiarla. Los campos de ingreso en una forma pueden ser principalmente de tres tipos: *INPUT,* que aplica a tipos de entradas como campos de texto y

[26] http://emision.uson.mx/tips/Cgi/cgi.htm

diferentes tipos de botones; *SELECT*, que aplica a listas o menúes desplegables, y *TEXTAREA*, que aplica a un área en donde el usuario puede introducir una o más líneas de texto.

La información que la forma envía al *CGI* se sustenta en el protocolo *HTTP* y se puede realizar a través de dos métodos definidos, que son *GET* y *POST*, el cual es indicado en el atributo *METHOD* de la etiqueta *FORM*[27]. La diferencia entre los métodos *GET* y *POST* es la forma en que se pasa la información al *CGI*. Con el método GET se pasa la información en la línea de comando, lo cual significa que necesita ser corta (algunos sistemas restringen la longitud de la línea de comando, lo que produce que se trunquen líneas de comando largas; en la mayoría de los sistemas *UNIX*, por ejemplo, esta restricción es 128 o 256). Usando el método *POST*, fuertemente recomendado para todas las aplicaciones, la información de entrada es codificada en un bloque de datos que se pasa a través de la entrada estandard (*STDIN*) al *CGI*. Este método evita todas las restricciones en la longitud de la línea de comando y ofrece un mecanismo de transmisión de los datos mucho más confiable entre las formas y los *CGI* relacionados[28].

Como se comentó anteriormente, la forma más común de pasar argumentos al *CGI* es mediante una forma en un documento *HTML*. En el momento en el que la forma se envía al servidor (esto se hace al oprimir el botón que tiene asociado el proceso *SUBMIT*), el cliente (el navegador) solicita al servidor la ejecución del *CGI* asociado, con los valores de las variables que fueron asignados a cada uno de los elementos de la forma. La transmisión de datos entre el cliente y el servidor sigue ciertas reglas: todos los espacios en blanco son convertidos al signo +, y todos los caracteres se transforman a un número en hexadecimal que equivale al ***ASCII*** del caracter, precedido por el signo %.

Además de las variables que recibe de la forma, el *CGI* maneja otras variables conocidas como variables de entorno. Estas variables están disponibles cuando el servidor recibe la petición de ejecución del *CGI*, y se clasifican en dos categorías generales: la primera agrupa aquellas que son independientes de la petición del cliente y que tienen el mismo valor sin importar la petición. Estos valores son propiedades del servidor y también son conocidas como metainformación. La segunda categoría involucra a aquellas variables que dependen de la petición del cliente, de las cuales la gran mayoría son específicas del cliente, pero algunas dependen del servidor al cual fue enviada la petición[29].

Las variables que no dependen de la petición del cliente son:

SERVER_SOFTWARE	Nombre y versión del servidor *WWW*.
SERVER_NAME	Nombre, dirección *IP* o **alias *DNS*** del servidor encargado de ejecutar el *CGI*.
GATEWAY_INTERFACE	Versión de la especificación *CGI* soportada por el servidor.

Las variables que dependen de la llegada de la petición son:

SERVER_PROTOCOL	Protocolo de comunicaciones utilizado por el cliente.
SERVER_PORT	Número de puerto por el que se gestiona la comunicación.
REQUEST_METHOD	Al solicitar la ejecución de un *CGI* a través de una forma *HTML*, esta variable adoptará los valores *POST* o *GET*, dependiendo del modo establecido para la transmisión de parámetros.
SCRIPT_NAME	Ruta virtual del *CGI* cuya ejecución ha sido solicitada.
QUERY_STRING	Toda la secuencia de caracteres que siguen al URL que hace referencia al *CGI*.
REMOTE_HOST	Nombre del cliente que solicita la ejecución del *CGI*.
REMOTE_ADDR	Dirección *IP* del cliente.

[27] http://www.cicei.ulpgc.es/gsi/curso_cgi2/cgi.htm

[28] http://a01-unix.gsyc.inf.uc3m.es/~esther/lrg9798/faq.html#11

[29] HTML and CGI Unleashed. John December and Mark Ginsburg. Sams.net Publishing. 1995. Pág. 394

AUTH_TYPE	Tipo de autorización asignada al usuario remoto.
REMOTE_USER	Nombre del usuario que, desde el cliente, ha solicitado la ejecución del *CGI*.
CONTENT_TYPE	Tipo de datos enviados por el cliente al servidor.
CONTENT_LENGTH	Tamaño del buffer de datos enviados por el cliente al servidor.
HTTP_USER_AGENT	Navegador que es utilizado por el cliente para hacer la petición.

Una vez ejecutado el *CGI*, éste envía un documento con los resultados al cliente que solicitó su ejecución. La salida está formada por un encabezado y por la información que se mostrará en el navegador del cliente.

3.1.2.5 JavaScript

JavaScript es un lenguaje interpretado, al igual que *VisualBasic*, *Perl*, *TCL*, etc., pero posee una característica que lo hace especialmente idóneo para trabajar en el *WWW*, ya que son los programas navegadores los que interpretan (y por tanto ejecutan) los programas escritos en *JavaScript*. De esta forma, es posible transmitir documentos *HTML* a través del *WWW* que incorporan el código fuente de un programa, convirtiéndose de esta forma en documentos dinámicos, y dejando de ser simples fuentes de información estáticas[30].

Los programas en *JavaScript* no son la primera forma que conoce el *WWW* para transformar información, dado que el uso de *CGIs* está ampliamente difundido. La diferencia básica que existe entre un programa *CGI* y uno escrito en JavaScript es que el *CGI* se ejecuta en el servidor *WWW*, mientras que el programa en Javascript se ejecuta en el cliente (es decir, en el navegador). Por regla general, el *CGI* necesita unos datos de entrada (que normalmente se proporcionan mediante una forma), los procesa y emite un resultado en forma de documento *HTML*. Esto implica tres transacciones en la red. La primera carga el documento en donde se encuentra la forma, la segunda envía los datos al servidor, y la tercera recibe la nueva página que ha generado el *CGI*.

Por el contrario, los programas escritos en *JavaScript* se ejecutan en el navegador del cliente, sin necesidad de que intervenga el servidor. De esta forma, una sola transacción basta para cargar el documento en el que se encuentra tanto la forma para los datos de entrada, como el programa en JavaScript que proporciona los resultados. Sin embargo, esto no significa que los *CGI* vayan a ser substituidos por *JavaScript*.

Las dos principales características de *JavaScript* son, por un lado que es un lenguaje basado en objetos, y por otro, es además un lenguaje orientado a eventos, debido por supuesto al tipo de entornos en los que se utiliza (*Windows* y sistemas X-*Windows*). Esto implica que gran parte de la programación en *JavaScript* se centra en describir objetos y escribir funciones que respondan a movimientos del ratón, pulsación de teclas, apertura y cerrado de ventanas o carga de un documento, entre otros eventos. Otras características de *JavaScript* son:

- Es interpretado por el cliente, no se genera código binario.
- Está integrado al código del documento *HTML*.
- No declara los tipos de datos de las variables.
- Es dinámico, ya que las referencias a objetos se revisan al tiempo de ejecución.
- Es seguro, ya que no se puede escribir al disco duro[31].

Por otra parte, *JavaScript* no es un lenguaje de propósito general. No permite un control absoluto sobre los recursos de la computadora. Cada programa en *JavaScript* solo tiene

[30] http://kal-el.ugr.es/~victor/curso_javascript/js_intro.html

[31] http://www.mty.itesm.mx/dcic/centros/ciete/tutor_js/requisito.htm

acceso al documento *HTML* en el que está integrado,y si acaso, a las ventanas del navegador dentro del cual se está ejecutando el programa en *JavaScript*.

Como se comentó anteriormente, *JavaScript* no es un sustituto de los *CGI's*. Hay ciertas cosas que no se pueden hacer en *JavaScript*, especialmente las relacionadas con el acceso a archivos. Sin embargo, es muy útil para depurar errores en los datos antes de pasárselos al *CGI* que los trata, por lo que el uso combinado de *JavaScript* y *CGIs* redunda en un mejor manejo de los datos y un menor número de transacciones en la red cuando se usan los *CGIs*.

Para que un navegador pueda leer el código *JavaScript*, éste debe de ser incluido dentro del código del documento *HTML*, y existen dos formas de hacerlo[32]: incrustar el código en el documento *HTML*, o cargarlo de un archivo separado. Además, es necesario saber cómo ocultar el código *JavaScript* a navegadores que no pueden interpretarlo. Esto es necesario ya que, actualmente, sólo los navegadores *Netscape Navigator* y *Microsoft Internet Explorer* son capaces de interpretar JavaScript.

Para incluir el programa *JavaScript* en un documento *HTML*, se incrusta el código entre las etiquetas <SCRIPT> y </SCRIPT>, especificando únicamente el lenguaje que se está utilizando:

```
<SCRIPT LANGUAGE="JavaScript">
Aquí va el código del script
</SCRIPT>
```

Estas etiquetas pueden ir en cualquier parte del código *HTML*, pero es recomendable que se coloque entre las etiquetas <HEAD> y </HEAD> cuando se utilizan funciones que son llamadas con manejadores de eventos, ya que permite que la función sea leída antes de ser llamada.

Por otra parte, para cargar el código *JavaScript* desde un archivo separado, se utiliza el atributo SRC de la etiqueta <SCRIPT>:

```
<SCRIPT LANGUAGE="JavaScript" SRC="JavaScriptCode.js">
</SCRIPT>
```

Pero hay que tener dos cosas en mente, al utilizar un archivo separado con el código *JavaScript*:

- La extensión del archivo debe ser .js.
- El atributo *LANGUAGE* puede ser omitido si se indica el tipo de fuente en la extensión del archivo.

La ventaja de usar archivos fuentes en lugar de incrustar el código en el documento *HTML*, es que el código *JavaScript* puede ser modificado sin alterar el documento *HTML*. Simplemente se reemplaza el viejo archivo .js por uno nuevo.

[32] Programming JavaScript for Netscape 2.0. Tim Ritchey / New Riders Publishing, Indianapolis, Indiana, USA 1996. Página 46

3.2 Funcionalidad y descripción del Sistema de Administración de Cuentas de Correo Electrónico de Red UNAM

3.2.1 Introducción .

A continuación se describe el comportamiento de los elementos o módulos con que cuenta el sistema. Este análisis es de importancia para el diseño del sistema, ya que proveen una oportunidad de involucrar al usuario desde el inicio y de asentar de una manera clara sus requerimientos, lo cual asegura un sistema con mayor robustez y que resuelva de mejor manera las necesidades solicitadas. Asi pues, se describe el análisis de acuerdo a las siguientes funciones.

La documentación de la funcionalidad se divide en :

Intención :	Explica de manera breve el propósito del módulo.
Actor o Actores:	Especifica el área usuaria del módulo
Precondición:	Son los elementos o puntos que deben de estar cubiertos antes de interactuar con el módulo.
Descripción:	Es la narrativa, dividida en pasos, de la interacción que habrá entre el usuario y el sistema. En esta sección, existen oraciones entre corchetes ("< >") que indican los casos de excepción , esto es , los casos en que el flujo del sistema puede variar.
Poscondición:	Establece lo que sucede una vez que se ejecuta el módulo. Que es lo que sucede como consecuencia del uso del módulo.
Excepciones:	Es la sección que explica lo que el sistema realiza cuando sucede cada excepción.
Módulos empleados:	Enlista los módulos que son empleados por el módulo descrito.
Restricciones :	Menciona todas aquellos aspectos que deben tomarse en cuenta para el desarrollo del módulo, por ejemplo : Tiempo de respuesta.

3.2.2 Análisis y diseño de la interfaz gráfica.

1 Inicio

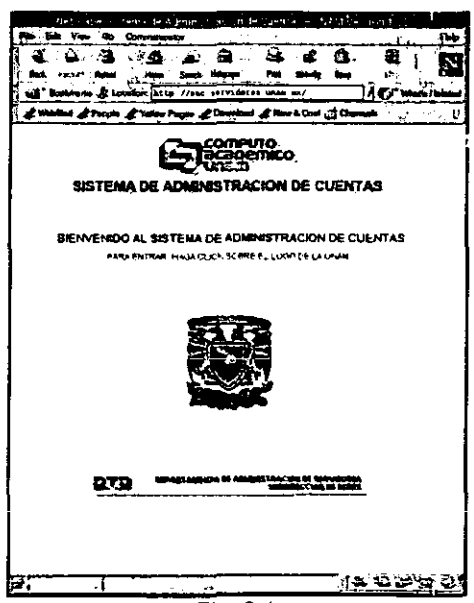


Fig. 3.1

Intención : Presentación de Bienvenida y el Ingreso al Sistema de Administración de Cuentas de Correo Electrónico.

Actores : Atención a Usuarios, Administrador del Sistema.

Precondición : El usuario debe de tener asignada una dirección *IP* válida para poder tener los privilegios para esta funcionalidad.

Descripción : Inicio – Se despliega la forma de Bienvenida e ingreso al sistema.

1. El Usuario del sistema da click en el logo de la UNAM. - El sistema valida el acceso al sistema a través de la dirección *IP* de la computadora que solicita el servicio. Excepciones : A

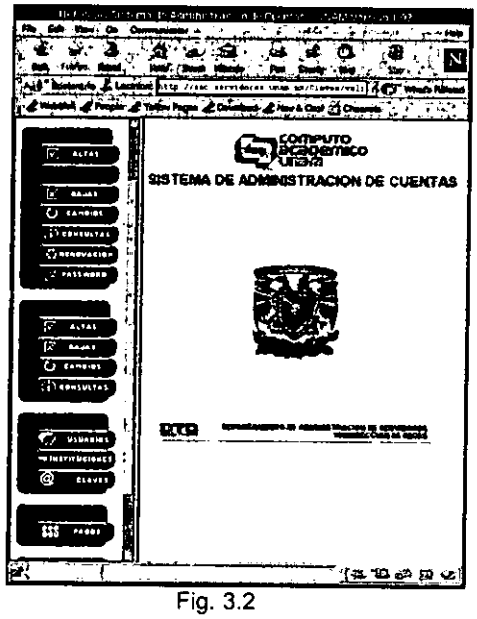


Fig. 3.2

Poscondición : El usuario del sistema accede de manera completa al Sistema.

Excepciones :

Paso	Condición	Acción del sistema	¿Continua?
1.	No se accede al sistema por IP no válida	Se despliega un mensaje de error en el acceso al sistema.	No

Restricciones : Que se cuente con dirección *IP* válida.

2 Módulo de Altas

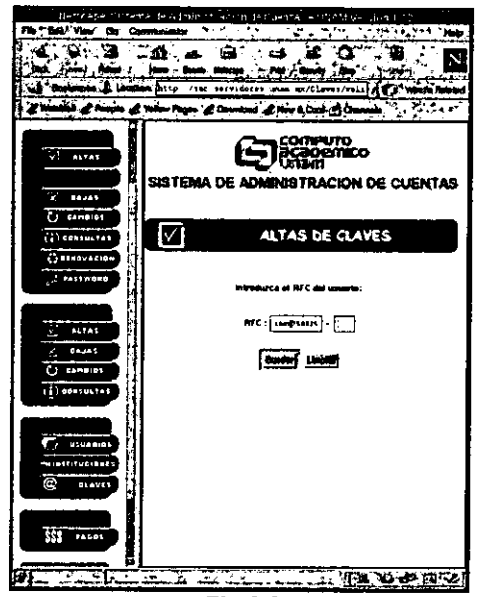


Fig 3.3

Intención : Registrar una nueva cuenta de Correo Electrónico así como los datos relacionados a esta cuenta en el servidor de correo electrónico y en la Base de Datos.

Actores : Atención a Usuarios, Administrador del Sistema.

Precondición : El usuario tienen los privilegios para esta funcionalidad. Tener los datos a introducir en el módulo para agilizar esta operación.

Descripción :

1. Inicio . El Usuario del sistema presiona el botón de Altas (Color Blanco) . El sistema despliega la pantalla con los campos de RFC listos para recibir la captura de estos datos del usuario a dar de Alta.

2. El Usuario del sistema llena los campos del RFC y oprime el Botón de Buscar .El sistema busca esos datos en la Base de Datos. Excepciones : A , B y C.

3. El sistema tendrá 2 opciones de regreso de información :

   3.1 El sistema encuentra datos del RFC ingresado . Se despliega la pantalla con los datos seleccionados . El Usuario del sistema presiona el botón "Aceptar".

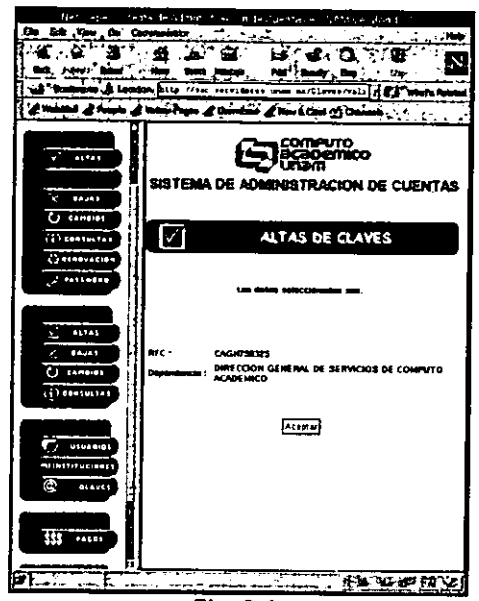


Fig. 3.4

3.1.1 El sistema despliega la pantalla con los datos generales del usuario así como la información de la o las cuentas de correo electrónico que tenga ya dadas de alta. El usuario del sistema presiona el botón de "Aceptar".

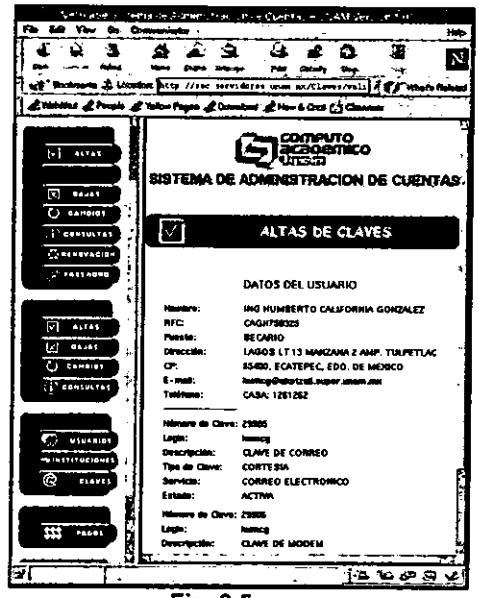


Fig. 3.5

3.1.2 El sistema brinda la opción de dar de alta una clave más al usuario seleccionado: Punto 4.

3.2 El sistema no encuentra los datos del RFC ingresado . Se despliega la pantalla de elección de la dependencia de procedencia.

3.3 El usuario del sistema escribe la palabra de referencia para una búsqueda mas rápida. El usuario del sistema presiona el botón de "Siguiente".

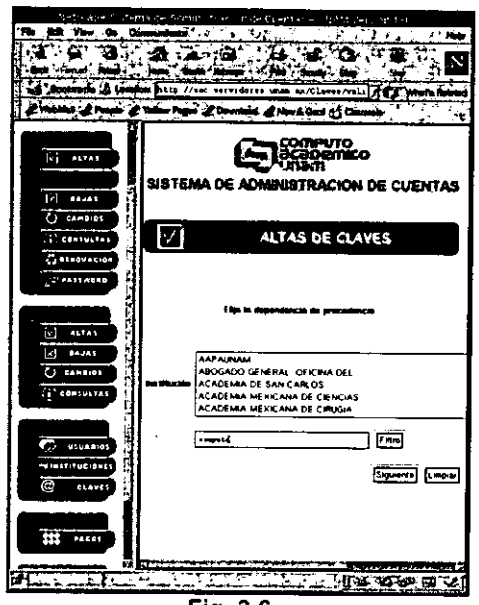


Fig. 3.6

3.4 El usuario del sistema presiona el botón "Filtro" . El sistema busca y encuentra las Instituciones que resultan de la equivalencia a la palabra escrita y se despliegan en el recuadro correspondiente a la Institución. El usuario del sistema elige la Institución . Presiona el botón "Siguiente". Excepción A.

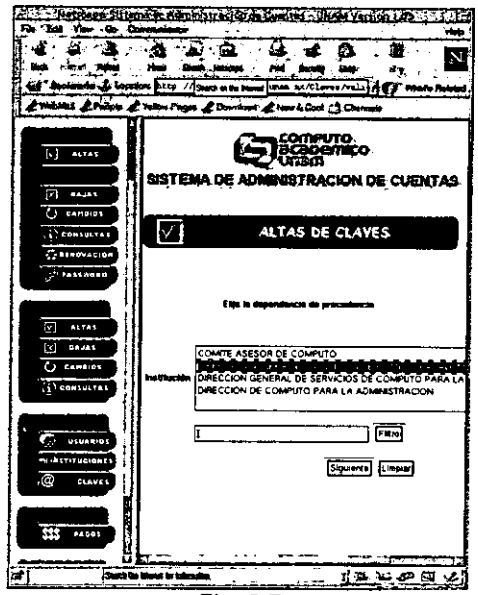


Fig. 3.7

4. El sistema despliega la pantalla para complementar el alta del servicio para el usuario seleccionado. El usuario completa el proceso de Alta y presiona el Botón de "Siguiente" . El sistema actualiza la nueva cuenta en el servidor de correo y actualiza la Base de Datos con los datos correspondientes . Excepciones : A y B.

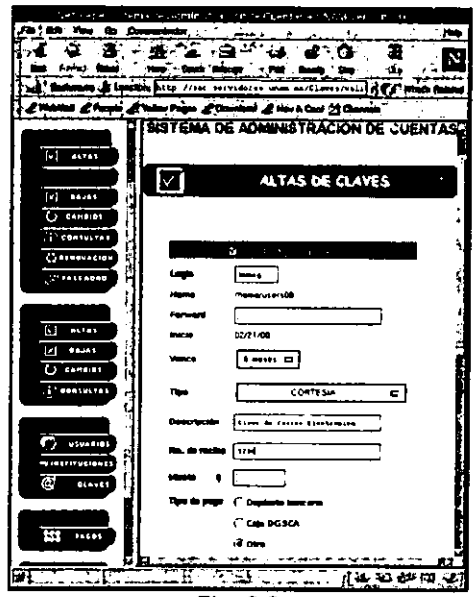


Fig. 3.8

5. Se despliega la pantalla de Impresión del Comprobante. El sistema entrega el comprobante del movimiento realizado y finaliza el proceso de Altas.

Fig. 3.9

Poscondición : El Sistema actualiza la Base de Datos y el servidor de correo con los cambios efectuados por causa de la Alta registrada.

Excepciones :

Paso	Condición	Acción del sistema	¿Continua?
2.	A - Error de Sintaxis del RFC	El sistema despliega un mensaje de error de RFC erróneo	Si
	B - El usuario presiona el Botón "Limpiar"	El sistema limpia todos los caracteres de los campos del RFC .	Si
	C - Campo de RFC Vacío	El sistema lo interpreta como Usuario nuevo. Continua el Paso 3	Si
4.1	A - No se encontró la Institución	El sistema no despliega Instituciones	Si
5 ó 5.1	A - No se puede registrar la clave en el Servidor.	Se despliega un mensaje de error de registro de clave en el servidor.	Si

Restricciones : La clave de usuario a registrar no debe de existir.

3 Módulo de Bajas.

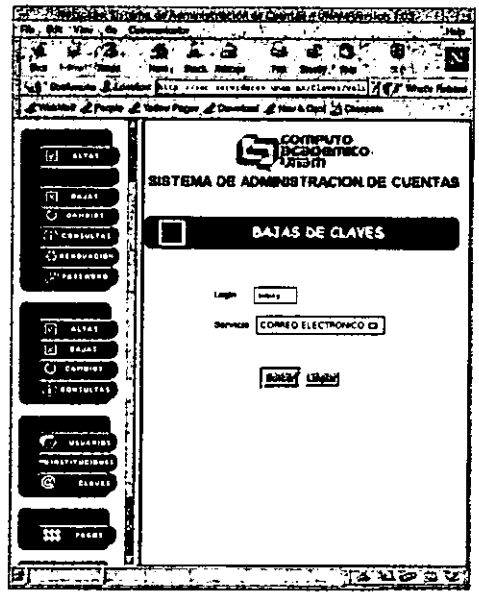


Fig. 3.10

Intención : Eliminar una cuenta de Correo Electrónico así como los datos relacionados a esta cuenta en el servidor de correo electrónico y en la Base de Datos.

Actores : Atención a Usuarios, Administrador del Sistema.

Precondición : El usuario tienen los privilegios para esta funcionalidad . Pasar previamente por un trámite administrativo: la cuenta deberá de haber pasado por un lapso de 3 meses sin uso, así como de una notificación al usuario de que su cuenta será dada de baja para así poder realizar esta acción.

Descripción :

1. Inicio . El Usuario del sistema presiona el botón de Bajas. El sistema despliega la pantalla para el módulo de Bajas .

2. El usuario del sistema escribe el **Login** que será dado de baja del sistema. Presiona el botón de "Buscar" . Excepción A

3. El sistema despliega la pantalla con los datos del Login ingresado.
   El usuario presiona el botón "Borrar". El sistema elimina el Login seleccionado y los datos correspondientes del Servidor de correo electrónico y del Base de Datos. Excepción A.(usuario en sesión)

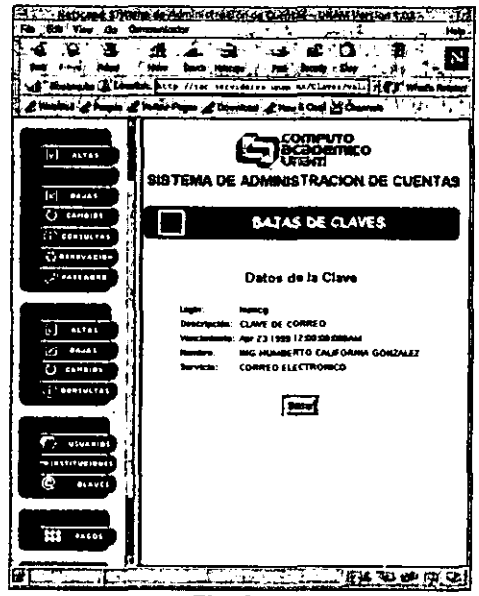


Fig. 3.11

4. El sistema despliega pantalla de Baja realizada.
   Finaliza el proceso de Bajas

Poscondición : El Sistema actualiza la Base de Datos y el servidor de correo con los cambios efectuados por causa de la Baja registrada.

Excepciones :

Paso	Condición	Acción del sistema	¿Continua?
2.	**A - No se encontró el Login solicitado**	**El sistema despliega un mensaje de error de Login no encontrado**	**Si**
3.	**A - La clave a dar de baja está en sesión**	**El sistema despliega un mensaje de error de Usuario en sesión .**	**Si**

Restricciones : El usuario no debe de estar en sesión para realizar la acción.

4 Módulo de Cambios

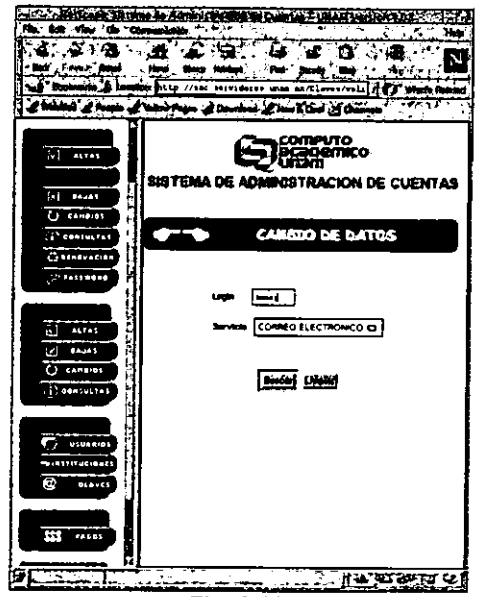


Fig. 3.12

Intención : Cambiar la información de cuenta de Correo Electrónico de la Base de Datos y/o en el servidor de correo.

Actores : Atención a Usuarios, Administrador del Sistema.

Precondición : El usuario tienen los privilegios para esta funcionalidad .

Descripción :

1. Inicio . El Usuario del sistema presiona el botón de Cambios. El sistema despliega la pantalla para el módulo de Cambios .
2. El usuario del sistema escribe el Login al que se le efectuará el cambio. El usuario presiona el botón "Buscar" . El sistema despliega la pantalla con los datos del Login ingresado.
   Excepciones: A y B.

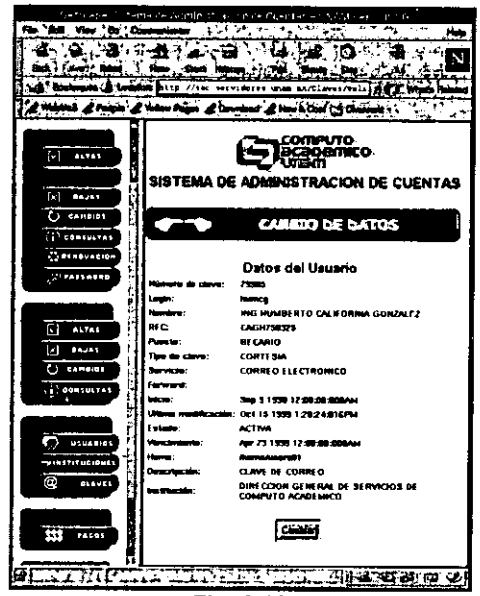


Fig. 3.13

3. El usuario presiona el botón "Cambiar". El sistema despliega la pantalla con los datos del usuario editables .

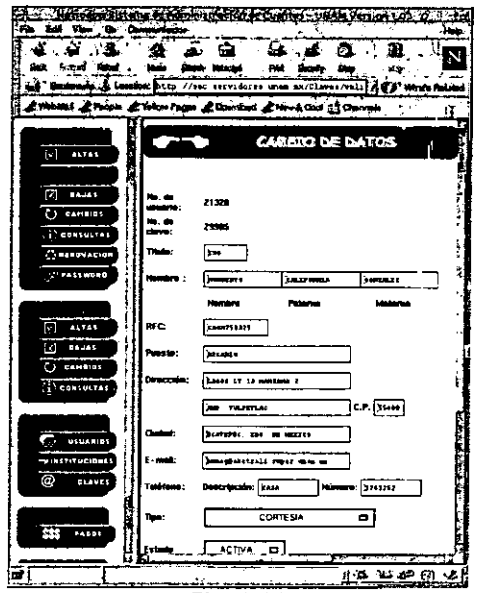


Fig. 3.14

4. El usuario del sistema realiza los cambios correspondientes y presiona el botón "Aceptar" . El sistema actualiza la información en la Base de Datos y/o el servidor de correo, según sea el cambio y regresa un mensaje de Actualización Realizada. Finaliza el módulo de cambios. Excepción A

Poscondición : El Sistema actualiza la Base de Datos y/o el servidor de correo dependiendo de los cambios efectuados .

Excepciones :

Paso	Condición	Acción del sistema	¿Continua?
2.	A - No se encontró el Login solicitado	El sistema despliega un mensaje de error de Login no encontrado	Si
	B – El usuario del sistema presiona el botón "Limpiar"	El sistema limpia todos los caracteres del campo Login.	Si
4.	A - No se puede llevar a cabo el cambio por falta de algún dato necesario para la modificación.	El sistema despliega un mensaje de notificación al usuario del sistema .	Si

Restricciones : No se puede hacer cambio de Login.

5 Módulo de Consultas

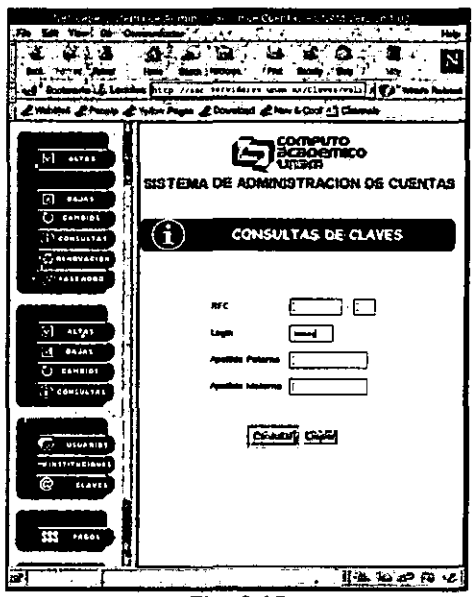


Fig. 3.15

Intención : Consultar la información de una cuenta de Correo Electrónico .

Actores : Atención a Usuarios, Administrador del Sistema.

Precondición : El usuario tiene los privilegios para esta funcionalidad.

Descripción :

1. Inicio . El Usuario del sistema presiona el botón de Consultas. El sistema despliega la pantalla para el módulo de Consultas .
2. El usuario del sistema ingresa los datos de búsqueda en los campos disponibles del módulo, presiona el botón "Consultar" . El sistema despliega la pantalla con todos los datos del usuario. Excepciones : A y B .

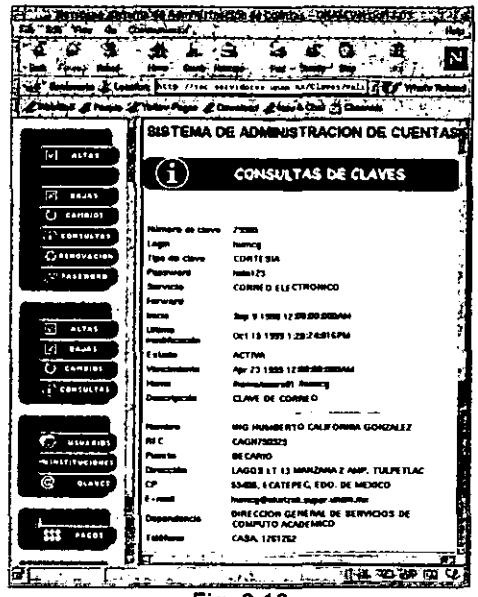


Fig. 3.16

Poscondición : El Sistema presenta la información del usuario solicitada por el Usuario del sistema.

Excepciones :

Paso	Condición	Acción del sistema	¿Continua?
2 .	A - No se encontró el Login solicitado.	El sistema despliega un mensaje de error de Login no encontrado	Si
	B – El usuario del sistema presiona el botón "Limpiar"	El sistema limpia todos los caracteres de todos los campos de búsqueda.	Si

Restricciones : Ninguna

6 Módulo de Renovación

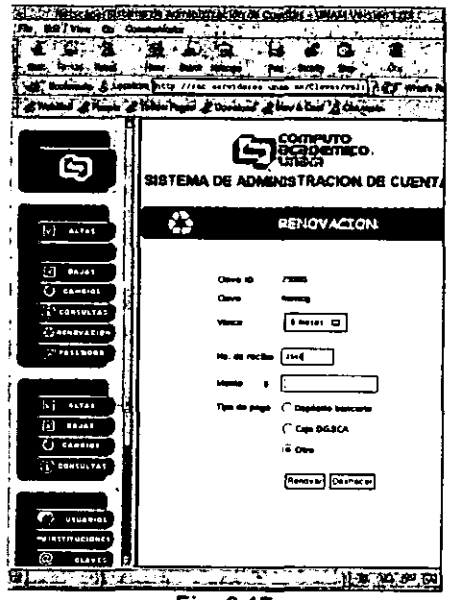


Fig. 3.17

Intención : Actualizar el periodo de expiración de una cuenta de Correo Electrónico.

Actores : Atención a Usuarios, Administrador del Sistema.

Precondición : El usuario tiene los privilegios para esta funcionalidad. Cuenta a renovar existente.

Descripción :

1. Inicio . El Usuario del sistema presiona el botón de Renovación. El sistema despliega la pantalla para el módulo de Renovación.
2. El usuario del sistema ingresa el Login a Renovar. presiona el botón "Buscar". El sistema despliega la pantalla con los datos del usuario a Renovar (clave ID, Clave, Nuevo Vencimiento, No. de Recibo, Monto y Tipo de pago) . Excepciones : A .
3. El Usuario captura la información necesaria para realizar la renovación. Presiona el botón "Renovar" . Excepciones : A y B.

Poscondición : El Sistema indica al usuario que ha sido realizada la renovación.

Excepciones :

Paso	Condición	Acción del sistema	¿Continua?
2.	A - No se encontró el Login solicitado.	El sistema despliega un mensaje de error de Login no encontrado	Si
3.	A - El usuario del sistema presiona el botón "Deshacer"	El sistema limpia todos los caracteres de todos los campos de búsqueda.	Si
	B. - El usuario del sistema no capturó todos los campos necesarios para realizar la renovación.	El sistema solicita al usuario la captura del o de los campos para terminar el proceso.	Si

Restricciones : El sistema solo podrá renovar de acuerdo a los periodos que se tiene en el combo Vence correspondiente y el tipo de pago debe de ser alguno de los que el sistema indica en el módulo.

7 Módulo de Cambio de *Password*

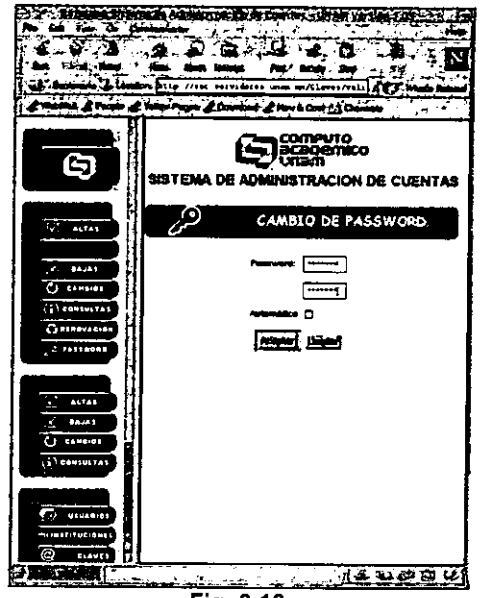


Fig. 3.18

Intención : Cambiar la contraseña de una cuenta de Correo Electrónico .

Actores : Atención a Usuarios, Administrador del Sistema.

Precondición : El usuario tiene los privilegios para esta funcionalidad.

Descripción :

1. Inicio . El Usuario del sistema presiona el botón de *Password*. El sistema despliega la pantalla para el módulo de Cambio de *Password*.

2. El usuario del sistema ingresa el Login a modificar, presiona el botón "Buscar". El sistema despliega la pantalla con los datos del login a modificar su *password*. Excepciones A

3. El usuario del sistema, presiona el botón "Cambiar" y el sistema muestra la pantalla final para el cambio de *password*. La pantalla presenta la opción de Cambio de *Password* Automático (por el sistema) o Cambio de *Password* personalizado (sugerido por el dueño de la cuenta de correo electrónico) el cual pide la repetición del mismo para asegurar el proceso.El usuario presiona el botón "Aceptar" .Excepciones A y B.

Poscondición : El Sistema indica al usuario que ha sido realizado el cambio de *password* mostrando en pantalla el nuevo *password* y generando un comprobante con los datos del usuario, incluyendo su nuevo *password*.

Excepciones :

Paso	Condición	Acción del sistema	¿Continua?
2.	A - No se encontró el Login solicitado.	El sistema despliega un mensaje de error de Login no encontrado	No
3.	A – *Password* Personalizado: El usuario del sistema no capturó alguno de los campos del sistema o no coinciden ambos campos"	El sistema indica que no hay coincidencia en ambos campos.	No
	B. - El usuario del sistema presiona el botón "Limpiar"	El sistema limpia todos los caracteres de todos los campos de este módulo	Si

Restricciones : El sistema no permite ***password*** con menos de 6 caracteres y mayores a 8 .

8 Módulo de Pagos

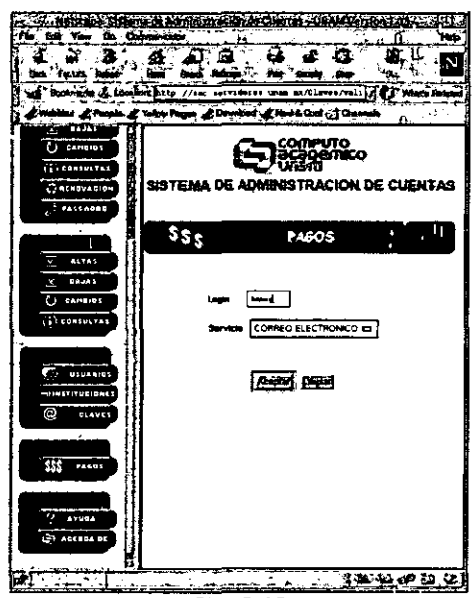


Fig. 3.19

Intención : Actualizar el Monto por concepto de cuentas de correo electrónico activas en servidor.unam.mx

Actores : Atención a Usuarios, Administrador del Sistema.

Precondición : El usuario tiene los privilegios para esta funcionalidad. Monto de Cuenta a actualizar existente. Pago por servicio realizado.

Descripción :

1. Inicio . El Usuario del sistema presiona el botón de Pagos. El sistema despliega la pantalla para el módulo de Pagos.
2. El usuario del sistema Ingresa el Login a modificar, presiona el botón "Aceptar". El sistema despliega la pantalla con los datos del login a modificar su pago. Excepciones A.
3. El usuario actualiza el monto del dueño del correo, el número de recibo y si tipo de pago. Presiona el botón "Aceptar". Excepciones: A

Poscondición : El Sistema indica al usuario que ha sido realizado su pago correspondiente generando un comprobante con los datos del usuario, incluyendo su pago realizado.

Excepciones :

Paso	Condición	Acción del sistema	¿Continua?
2.	**A - No se encontró el Login solicitado.**	**El sistema despliega un mensaje de error de Login no encontrado**	**No**
3.	**B. - El usuario del sistema presiona el botón "Limpiar"**	**El sistema indica que no hay coincidencia en ambos campos.**	**No**

Restricciones : El tipo de pago debe de ser alguno de los que el sistema indica en el módulo.

9 Módulo de Reportes

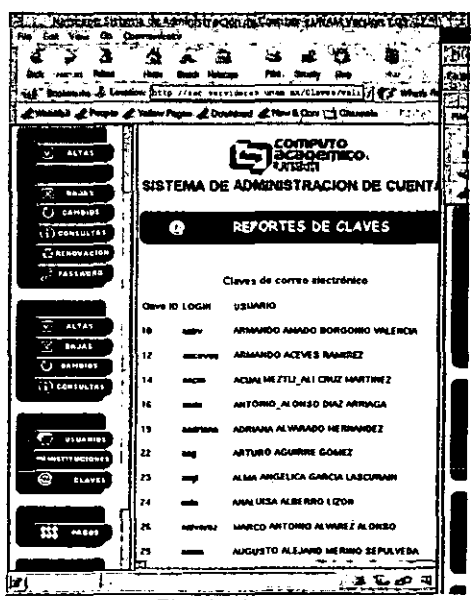


Fig. 3.20
Por Claves o Usuarios

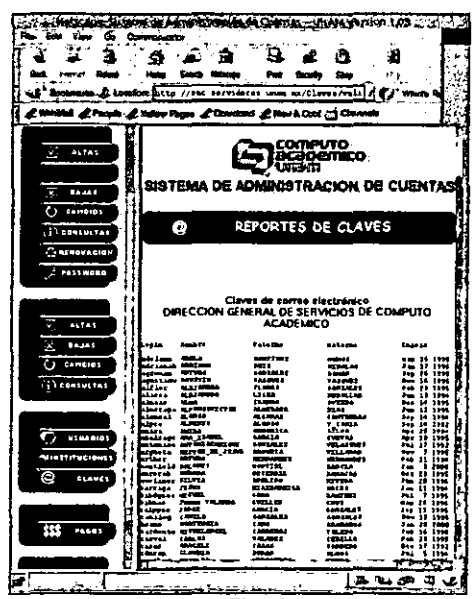


Fig. 3.21
Por Instituciones

Intención : Obtener la información de las cuentas de Correo Electrónico , de las instituciones o de los usuarios de Servidor.

Actores : Atención a Usuarios, Administrador del Sistema.

Precondición : El usuario tiene los privilegios para esta funcionalidad.

Descripción :

1. Inicio . El Usuario del sistema presiona el botón del tipo de Reporte a obtener. El sistema despliega la pantalla para el módulo de Reporte seleccionado.
2. Para el Reporte de Claves y de Usuarios, el sistema despliega una pantalla para seleccionar la letra con la que inician las cuentas a reportar. El sistema despliega los datos principales de los usuarios y claves de acuerdo a la letra seleccionada. Excepciones: A.
3. Para el reporte de Instituciones, el usuario especifica el filtro por el cual buscará . El sistema desplegará los usuarios y claves correspondientes a esta Institución. . Excepciones: A.

Poscondición : El Sistema indica al usuario el resultado del reporte seleccionado y con los patrones de búsqueda elegidos por él.

Excepciones :

Paso	Condición	Acción del sistema	¿Continua?
2 .	**A - No se encontró el patrón de búsqueda**	**El sistema despliega un mensaje de aviso Indicando que no se encontró el patrón de búsqueda.**	**No**
3.	**B. No se encontró el patrón de búsqueda**	**El sistema despliega un mensaje de aviso Indicando que no se encontró el patrón de búsqueda**	**No**

Restricciones : Ninguna.

10 Módulo de Instituciones

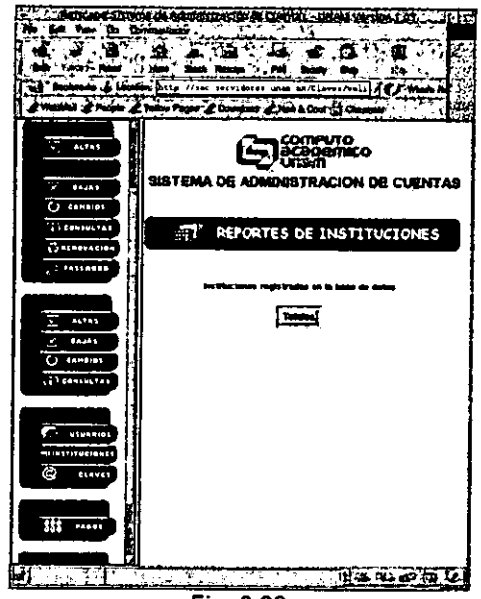


Fig. 3.22

Intención : Actualizar y Consultar la información de las Instituciones que cuentan o contarán con usuarios de correo electrónico en servidor.unam.mx

Actores : Atención a Usuarios, Administrador del Sistema.

Precondición : El usuario tiene los privilegios para esta funcionalidad.

Descripción :

1. Inicio . El Usuario del sistema elige el tipo de operación a realizar del menú correspondiente a Instituciones.
2. Si la elección es una Alta , el sistema pide el nombre de la Institución a Añadir. El sistema solicita si es Institución Interna o Externa. Presiona botón Aceptar. Excepciones: A.
3. Si es una Baja , a través de un filtro, el usuario del sistema selecciona la Institución a eliminar. Presiona botón Aceptar Excepciones A y B.
4. Para cambios de Instituciones, el usuario del sistema busca la Institución a modificar . Permite cambiar el nombre y el tipo de la Institución. Presiona botón Aceptar. Excepciones: A.
5. Para las consultas, el sistema despliega en pantalla todas las Instituciones registradas en el sistema.

Poscondición : El Sistema actualiza la información de las Instituciones para los módulos de Altas, Modificación y Bajas.

Excepciones :

Paso	Condición	Acción del sistema	¿Continua?
2.	A- No se puede registrar la Institución en el Servidor.	El sistema despliega un mensaje de error indicando el problema del registro.	No
3.	A. No se encontró la Institución buscada.	El sistema no regresa información	Si
	B. El sistema detecta usuarios pertenecientes a la Institución a eliminar	El sistema avisa al usuario la existencia de usuarios de la Institución.	No
4.	A. No se encontró la Institución buscada.	El sistema no regresa información	Si

Restricciones : Para el módulo de bajas, no deben existir usuarios pertenecientes a la Institución a eliminar.

Continuando con el análisis, diseño y construcción del sistema, en este capítulo se definieron las herramientas que se emplearon, así como la funcionalidad del sistema a nivel modular, especificando cada una de las pantallas, funciones y operación de cada uno de sus componentes y la relación que habrá entre ellos.

CAPITULO 4

LA COMUNICACIÓN ENTRE LA INTERFAZ GRÁFICA Y EL SERVIDOR (SOCKET)

CAPITULO 4

La comunicación entre la interfaz gráfica y el servidor (socket)

A continuación se explica la forma en que la interfaz gráfica se comunicará con el servidor de correo electrónico y nuestra base de datos , todo esto a través del socket, el cual será nuestra herramienta que hará la función de enlace entre estos elementos a través de la red.

4.1 Conceptos generales

4.1.1 Redes de comunicaciones [33]

Durante los últimos 20 años, las computadoras y las redes han tenido un gran avance y mucho éxito, ya que se han utilizado varias herramientas que han hecho que la productividad se multiplique y que el trabajo sea más eficaz, tanto en las empresas como en los usuarios individuales. Esto ha permitido atender necesidades privadas o comerciales, por ejemplo las operaciones bancarias.

La definición más sencilla que podemos establecer para una red de computadoras es la siguiente: " es un grupo de computadoras (y terminales, en general) interconectadas a través de uno o varios caminos o medios de transmisión ". En los inicios de las redes generalmente se utilizaba la línea telefónica para esta transmisión, aunque últimamente ya se están utilizando otro tipo de enlaces como satélites y fibra óptica.

La finalidad de las redes es transferir e intercambiar datos entre computadoras y terminales. Este intercambio de datos forma parte de nuestras vidas, como por ejemplo los cajeros automáticos.

Topologías de Red

La configuración de una red se conoce como topología. La topología es la forma (la conectividad física) de la red. Para diseñar una red hay que plantearse tres objetivos principalmente:

- Proporcionar la máxima confiabilidad posible, para garantizar la recepción correcta de todo el tráfico.
- Encaminar el tráfico entre el **ETD** (equipo terminal de datos) transmisor y el receptor a través del camino más óptimo dentro de la red.
- Proporcionar al usuario final un tiempo de respuesta óptimo y un caudal eficaz máximo.

A continuación se enlistan las topologías de red más comunes:

- Topología jerárquica o en árbol
- Topología en bus
- Topología en estrella
- Topología en anillo
- Topología en malla

[33] Black, Uyless; Redes de Computadoras; Madrid , España, Ed. Ra-Ma, 1989; Cap. 1

Topología jerárquica o en árbol.

El software que controla la red es relativamente simple, y la topología proporciona un punto de concentración de las tareas de control y resolución de errores. Generalmente el ETD que se encuentra en el nivel más alto de la jerarquía es el que controla la red.

Aunque esta topología resulta interesante por ser fácil de controlar, puede presentar problemas en cuanto a la aparición de cuellos de botella, lo cual plantea serios problemas de confiabilidad.

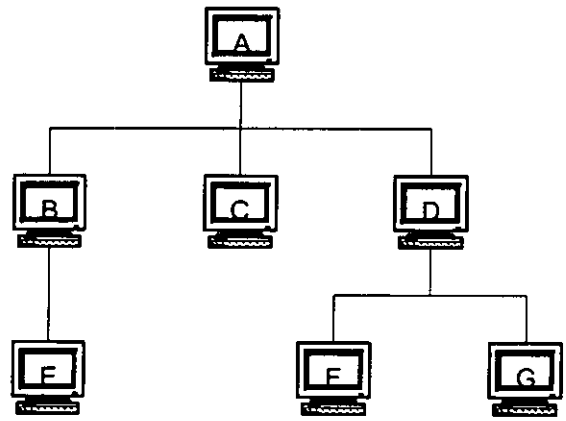


Fig. 4.1 Topología jerárquica o en árbol

Topología en bus.

Se usa generalmente en las redes de área local. Aquí es relativamente fácil controlar el flujo del tráfico entre los distintos ETD, ya que el bus permite que todas las estaciones reciban las transmisiones, es decir, una estación puede difundir la información a todas las demás. Su principal limitación es que sólo existe un canal de comunicaciones para todos los dispositivos de la red.

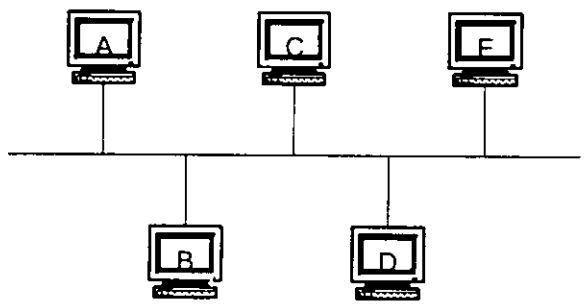


Fig. 4.2 Topología en bus

Topología en estrella.

Es una de las más empleadas en los sistemas de comunicación de datos. Su software no es complicado y su flujo de tráfico es sencillo. Todo el tráfico emana del núcleo de la estrella, el nodo central (A), que tiene el control total de los ETD conectados a él y tiene la capacidad de detectar las averías. Una red en estrella puede sufrir saturaciones y problemas en caso de avería del nodo central.

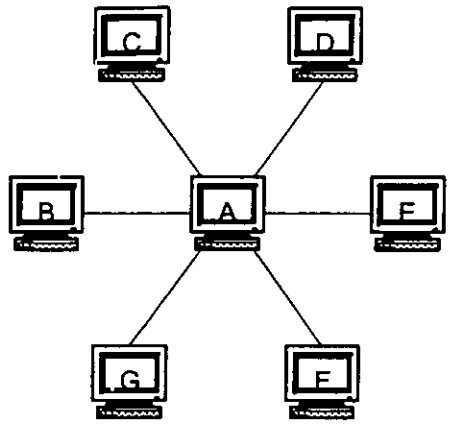


Fig. 4.3 Topología en estrella

Topología en anillo.

Se llama así por el aspecto circular del flujo de datos, que generalmente fluyen en una sola dirección, y cada estación recibe la señal y la retransmite a la siguiente del anillo. En esta topología son muy raros las colisiones que se presentan con frecuencia en los sistemas en estrella o en árbol. La lógica necesaria para poner en marcha una red de este tipo es simple. Cada componente lleva a cabo tareas muy sencillas: aceptar los datos, enviarlos al ETD conectado al anillo o retransmitirlos al próximo componente del mismo. El problema más importante que se presenta en esta topología es que todos los componentes del anillo están unidos por un mismo canal y si falla el canal entre dos nodos, la red se interrumpe.

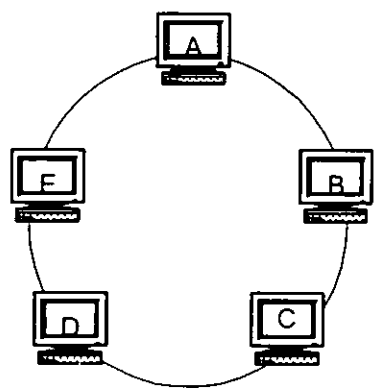


Fig. 4.4 Topología en anillo

Topología en malla.

Ofrece una multiplicidad de caminos entre los ETD y ECD (equipo de conmutación de datos), lo que le permite orientar el tráfico por trayectorias alternativas en caso de que algún nodo este averiado u ocupado. Muchos usuarios prefieren la confiabilidad de una red malla a otras alternativas a pesar de que su realización es compleja y cara.

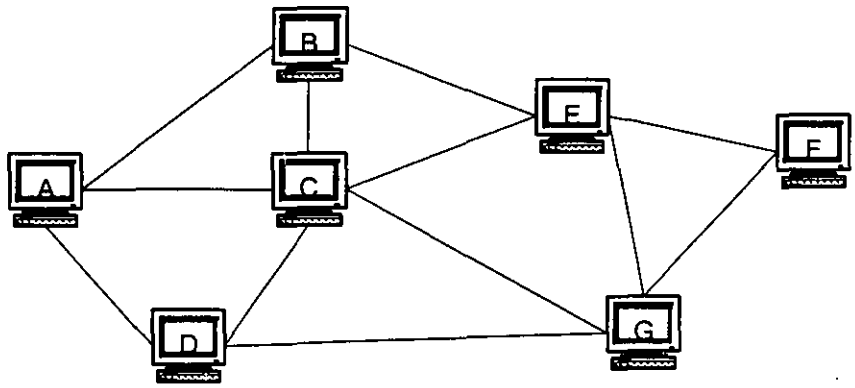


Fig. 4.5 Topología en malla

4.1.2 Conjunto de protocolos TCP/IP [34]

Protocolos

Establecen una descripción formal de los formatos que deberán presentar los mensajes para poder ser intercambiados por equipos de cómputo; además definen las reglas que ellos deben seguir para lograrlo.

Los protocolos están presentes en todas las etapas necesarias para establecer una comunicación entre equipos de cómputo, desde aquellas de más bajo nivel (como la transmisión de flujos de bits en un medio físico) hasta aquellas de más alto nivel (como el compartir o transferir información desde una computadora a otra en la red).

Tomando al modelo **OSI** (Open Systems Interconection) como referencia podemos afirmar que para cada capa o nivel que él define existen uno o más protocolos interactuando. Los protocolos son entre pares (peer-to-peer), es decir, un protocolo de algún nivel dialoga con el protocolo del mismo nivel en la computadora remota.

Conjunto de Protocolos TCP/IP
Origen

Fueron desarrollados como parte del proyecto DARPA a mediados de la década de los setentas, dando lugar a la red **ARPANET**.

Su objetivo principal fue que computadoras cooperativas compartieran recursos mediante una red de comunicaciones.

ARPANET dejó de funcionar oficialmente en 1990.

En 1973, la Agencia de Proyectos de Investigación Avanzada para la Defensa (**DARPA**), de los Estados Unidos, inició un programa para la investigación de tecnologías que permitieran la transmisión de paquetes de información entre redes de diferentes tipos y características. El proyecto tenía por objetivo la interconexión de redes, por lo que se le denominó "Internetting", y a la familia de redes de computadoras que surgió de esta investigación se le denominó "Internet". Los protocolos desarrollados se denominaron el Conjunto de Protocolos TCP/IP, que surgieron de dos

[34] http://www.lpis.com/ayudas/tcp.html

conjuntos previamente desarrollados; los Protocolos de Control de Transmisión (Transmition Control Protocol) e Internet (Internet Protocol).

Su relación con el Modelo OSI

<table>
<tr><td>Aplicación</td><td rowspan="3">TELNET</td><td rowspan="3">FTP</td><td rowspan="3">SNMP</td><td rowspan="3">SMTP</td><td rowspan="3">DNS</td><td rowspan="3">HTTP</td></tr>
<tr><td>Presentación</td></tr>
<tr><td>Sesión</td></tr>
<tr><td>Transporte</td><td colspan="6">TCP</td></tr>
<tr><td>Red</td><td colspan="6">IP</td></tr>
<tr><td rowspan="2">Enlace de Datos</td><td colspan="5">802.2</td><td>X.25</td><td>LLC/SNAP</td></tr>
<tr><td>802.3</td><td colspan="3">802.5</td><td colspan="2">LAPB</td><td>ATM</td></tr>
<tr><td>Física</td><td>Ethernet</td><td colspan="2">Token Ring</td><td>FDDI</td><td colspan="2">Línea Síncrona WAN</td><td>SONET</td></tr>
</table>

Fig. 4.6

TCP = TRANSFER CONTROL PROTOCOL
IP = INTERNET PROTOCOL

En la actualidad, las funciones propias de una red de computadoras pueden ser divididas en las siete capas propuestas por ISO para su modelo de sistemas abiertos (OSI). Sin embargo la implantación real de una arquitectura puede diferir de este modelo. Las arquitecturas basadas en TCP/IP proponen cuatro capas en las que las funciones de las capas de Sesión y Presentación son responsabilidad de la capa de Aplicación y las capas de Liga de Datos y Física son vistas como la capa de Interfase a la Red. Por tal motivo para TCP/IP sólo existen las capas Interfase de Red, la de Intercomunicación en Red, la de Transporte y la de Aplicación. Como puede verse TCP/IP presupone independencia del medio físico de comunicación, sin embargo existen estándares bien definidos a los nivel de Liga de Datos y Físico que proveen mecanismos de acceso a los diferentes medios y que en el modelo TCP/IP deben considerarse la capa de Interfase de Red; siendo los más usuales el proyecto IEEE802, **Ethernet, Token Ring y FDDI**.

Modelo de capas de TCP/IP

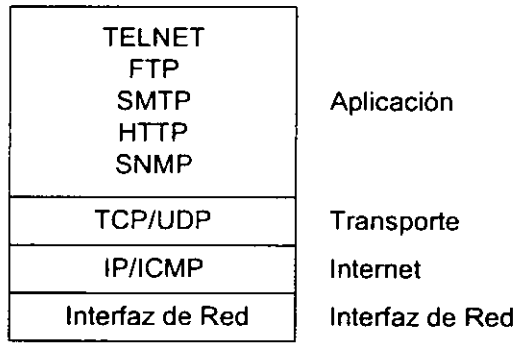


Fig. 4.7

Descripción del Modelo de Capas de TCP/IP[35]

- **Capa de Aplicación.**

Invoca programas que acceden servicios en la red. Interactúan con uno o más protocolos de transporte para enviar o recibir datos, en forma de mensajes o bien en forma de flujos de bytes.

- **Capa de Transporte.**

  - Provee comunicación extremo a extremo desde un programa de aplicación a otro.
  - Regula el flujo de información. Puede proveer un transporte confiable asegurándose que los datos lleguen sin errores y en la secuencia correcta.
  - Coordina a múltiples aplicaciones que se encuentren interactuando con la red simultáneamente de tal manera que los datos que envíe una aplicación sean recibidos correctamente por la aplicación remota, esto lo hace añadiendo identificadores de cada una de las aplicaciones. Realiza además una verificación por suma, para asegurar que la información no sufrió alteraciones durante su transmisión.

- **Capa Internet.**

  - Controla la comunicación entre un equipo y otro, decide que rutas deben seguir los paquetes de información para alcanzar su destino. Conforma los paquetes IP que serán enviados por la capa inferior.
  - Desencapsula los paquetes recibidos pasando a la capa superior la información dirigida a una aplicación.

- **Capa de Interfaz de Red.**

  - Emite al medio físico los flujos de bit y recibe los que de él provienen. Consiste en los manejadores de los dispositivos que se conectan al medio de transmisión.

Protocolo Internet (IP)
Características

El Protocolo Internet proporciona un servicio de distribución de paquetes de información orientado a no conexión de manera no fiable. La orientación a no conexión significa que los paquetes de información, que será emitido a la red, son tratados independientemente, pudiendo viajar por diferentes trayectorias para llegar a su destino. El término no fiable significa más que nada que no se garantiza la recepción del paquete.

La unidad de información intercambiada por IP es denominada **datagrama**. Tomando como analogía los marcos intercambiados por una red física los datagramas contienen un encabezado y una área de datos. IP no especifica el contenido del área de datos, ésta será utilizada arbitrariamente por el protocolo de transporte.

[35] http://www.geocities.com/SiliconValley/Lab/7583/articulos/asartic2.htm

Direcciones IP

Para que en una red dos computadoras puedan comunicarse entre sí ellas deben estar identificadas con precisión. Este identificador puede estar definido en niveles bajos (identificador físico) o en niveles altos (identificador lógico) de pendiendo del protocolo utilizado. TCP/IP utiliza un identificador denominado dirección Internet o dirección IP, cuya longitud es de 32 bits. La dirección IP identifica tanto a la red a la que pertenece una computadora como a ella misma dentro de dicha red.

Clases de Direcciones IP

Clases	Número de Redes	Número de Nodos	Rango de Direcciones IP
A	127	16,777,215	1.0.0.0 a la 127.0.0.0
B	4095	65,535	128.0.0.0 a la 191.255.0.0
C	2,097,151	255	192.0.0.0 a la 223.255.255.0

Fig. 4.8

Unidad Máxima de Transferencia MTU (Maximum Transfer Unit)

La Unidad de Transferencia Máxima determina la longitud máxima, en bytes, que podrá tener un datagrama para ser transmitida por una red física. Obsérvese que este parámetro está determinado por la arquitectura de la red: para una red Ethernet el valor de la MTU es de 1500 bytes. Dependiendo de la tecnología de la red los valores de la MTU pueden ir desde 128 hasta unos cuantos miles de bytes.

Fragmentación

La arquitectura de interconexión de redes propuesta por TCP/IP indica que éstas deben ser conectadas mediante una compuerta. Sin obligar a que la tecnología de las redes físicas que se conecten sea homogénea. Por tal motivo si para interconectar dos redes se utilizan medios con diferente MTU, los datagramas deberán ser fragmentados para que puedan ser transmitidos. Una vez que los paquetes han alcanzado la red externa los datagramas deberán ser reensamblados.

Protocolo de Mensajes de Control de Internet ICMP (Internet Control Message Protocol)

Su función es la de notificar los eventos en los que los paquetes enviados no alcanzaron su destino. Proporciona un medio de transporte para que los equipos compuerta se envíen mensajes de control y error. ICMP no está orientado a la corrección de errores, sólo a su notificación.

Protocolo de Control de Transferencia

- Proporciona comunicación bidireccional completa mediante circuitos virtuales.
- Desde el punto de vista del usuario la información es transmitida por flujos de datos.
- Confiabilidad en la transmisión de datos por medio de:
- Asignación de números de secuencia a la información segmentada.
- Validaciones por suma.
- Reconocimiento de paquetes recibidos.
- Utiliza el principio de ventana deslizable para esperar reconocimientos y reenviar información.

Proporciona un mecanismo confiable para la transferencia de flujos de información. Aunque está íntimamente relacionado con IP, TCP es un protocolo independiente de propósito general. Al

ser un protocolo de alto nivel su función es que grandes volúmenes de información lleguen a su destino correctamente, pudiendo recobrar la pérdida esporádica de paquetes.

A cada paquete que es enviado se le asigna un número de identificador, el equipo que lo recibe deberá enviar un reconocimiento de dicho paquete, lo que indicará que fue recibido. Si después de un tiempo dado el reconocimiento no ha sido recibido el paquete se volverá a enviar. Obsérvese que puede darse el caso en el que el reconocimiento sea el que se pierda, en este caso se reenviará un paquete repetido.

4.1.3 Socket

Definición [36]

Un socket es un punto final de un enlace de comunicación de dos vías entre dos programas que se ejecutan a través de la red.

Los sockets no son más que puntos o mecanismos de comunicación entre procesos que permiten que un proceso hable (emita o reciba información) con otro proceso incluso estando estos procesos en distintas máquinas. Esta característica de inter conectividad entre máquinas hace que el concepto de socket nos sirva de gran utilidad. Esta interfaz de comunicaciones es una de las distribuciones de Berkeley al sistema UNIX, implementándose las utilidades de interconectividad de este Sistema Operativo (por ejemplo las aplicaciones rlogin, telnet, ftp, etc.) usando sockets.

Un socket es al sistema de comunicación entre computadoras lo que un buzón o un teléfono es al sistema de comunicación entre personas: un punto de comunicación entre dos agentes (procesos o personas respectivamente) por el cual se puede emitir o recibir información. La forma de referenciar un socket por los procesos implicados es mediante un descriptor del mismo tipo que el utilizado para referenciar archivos. Debido a esta característica, se podrá realizar redirecciones de los archivos de E/S estándar (descriptores 0,1 y 2) a los sockets y así combinar entre ellos aplicaciones de la red. Todo nuevo proceso creado heredará, por tanto, los descriptores de sockets de su padre.

La comunicación entre procesos a través de sockets se basa en la filosofía CLIENTE-SERVIDOR: un proceso en esta comunicación actuará de proceso servidor creando un socket cuyo nombre conocerá el proceso cliente, el cual podrá "hablar" con el proceso servidor a través de la conexión con dicho socket nombrado.

El proceso crea un socket sin nombre cuyo valor de vuelta es un descriptor sobre el que se leerá o escribirá, permitiéndose una comunicación bidireccional, característica propia de los sockets y que los diferencia de los pipes, o canales de comunicación unidireccional entre procesos de una misma máquina.

El mecanismo de comunicación vía sockets tiene los siguientes pasos:

1) El proceso servidor crea un socket con nombre y espera la conexión.
2) El proceso cliente crea un socket sin nombre.
3) El proceso cliente realiza una petición de conexión al socket servidor.
4) El cliente realiza la conexión a través de su socket mientras el proceso servidor mantiene el socket servidor original con nombre.

Es muy común en este tipo de comunicación lanzar un proceso hijo, una vez realizada la conexión, que se ocupe del intercambio de información con el proceso cliente mientras el proceso padre servidor sigue aceptando conexiones. Para eliminar esta característica se cerrará el

[36] http://www.geocities.com/SiliconValley/Lakes/2227/red/sockets.html

descriptor del socket servidor con nombre en cuanto realice una conexión con un proceso socket cliente.

Todo socket viene definido por dos características fundamentales:

- El **tipo del socket**, que indica la naturaleza del mismo, el tipo de comunicación que puede generarse entre los sockets.
- El **dominio del socket** especifica el conjunto de sockets que pueden establecer una comunicación con el mismo.

Vamos a tratar con más detalle estos dos aspectos:

Tipos de sockets. [37]

Define las propiedades de las comunicaciones en las que se ve envuelto un socket, esto es, el tipo de comunicación que se puede dar entre cliente y servidor. Estas pueden ser:

- Fiabilidad de transmisión.
- Mantenimiento del orden de los datos.
- No duplicación de los datos.
- El "Modo Conectado" en la comunicación.
- Envío de mensajes urgentes.

Los tipos disponibles son los siguientes:

- **Tipo SOCK_DGRAM**: sockets para comunicaciones en modo no conectado, con envío de datagramas de tamaño limitado (tipo telegrama). En dominios Internet como la que nos ocupa el protocolo del nivel de transporte sobre el que se basa es el UDP.
- **Tipo SOCK_STREAM**: para comunicaciones fiables en modo conectado, de dos vías y con tamaño variable de los mensajes de datos. Por debajo, en dominios Internet, utiliza el protocolo TCP.
- **Tipo SOCK_RAW**: permite el acceso a protocolos de más bajo nivel como el IP (nivel de red)
- **Tipo SOCK_SEQPACKET**: tiene las características del SOCK_STREAM pero además el tamaño de los mensajes es fijo.

Dominio de socket.

Indica el formato de las direcciones que podrán tomar los sockets y los protocolos que soportarán dichos sockets.

La estructura genérica es:

```
struct sockaddr {
        u_short   sa_family;      /* familia */
        char      sa_data[14];    /* dirección */
};
```

Los dominios pueden ser:

[37] Stevens Richard W. , UNIX Network Programming. Networking APIs: Sockets and XTI Volume 1. Chapter 4

- **Dominio AF_UNIX** (Address Family UNIX): El cliente y el servidor deben estar en la misma máquina. Debe incluirse el archivo de cabecera /usr/include/sys/un.h.

  La estructura de una dirección en este dominio es:

```
struct sockaddr__un {
        short       sun__family;  /* en este caso AF_UNIX */
char        sun__data[108]; /* dirección */
        };
```

- **Dominio AF_INET** (Address Family INET): El cliente y el servidor pueden estar en cualquier máquina de la red Internet. Deben incluirse los archivos de cabecera /usr/include/netinet/in.h, /usr/include/arpa/inet.h, /usr/include/netdb.h.

  La estructura de una dirección en este dominio es:

```
struct in__addr {
        u__long    s__addr;
};

struct sockaddr__in {
short       sin_family;  /* en este caso AF_INET */
        u__short    sin_port;   /* numero del puerto */
struct in__addr        sin__addr; /* direcc Internet */
        char        sin_zero[8];    /* campo de 8 ceros */
};
```

- **Dominio AF_NS**: Servidor y cliente deben estar en una red XEROX.

- **Dominio AF_CCITT**: Para protocolos CCITT, protocolos X25, etc.

Puertos [38]

Generalmente, una computadora tiene una sola conexión física con la Red. Todos los datos destinados a una computadora particular llegan a través de la conexión. Sin embargo, los datos podrían ser utilizados por diferentes aplicaciones ejecutándose en la computadora. ¿Entonces cómo sabe la computadora a qué aplicación enviarle los datos? A través del uso de los puertos.

Los datos transmitidos por Internet están acompañados por una información de dirección que identifica la computadora y el puerto al que están destinados. La computadora está identificada por su dirección IP de 32 bits, esta dirección se utiliza para enviar los datos a la computadora correcta en la red. Los puertos están identificados por un número de 16 bits, que TCP y **UDP** utilizan para enviar los datos a la aplicación correcta.

En aplicaciones basadas en la conexión, una aplicación establece una conexión con otra aplicación uniendo un socket a un número de puerto. Esto tiene el efecto de registrar la aplicación con el sistema para recibir todos los datos destinados a ese puerto. Dos aplicaciones no pueden utilizar el mismo puerto: intentar acceder a un puerto que ya está utilizado dará un error.

En comunicaciones basadas en datagramas, los paquetes de datagramas contienen el número de puerto del destinatario.

[38] http://www.comunnet.com.mx/Informacion/%2373419

De lo anterior deducimos que, una aplicación servidor normalmente escucha a un puerto específico esperando una petición de conexión de un cliente. Cuando llega una petición de conexión, el cliente y el servidor establecen una conexión dedicada sobre la cual pueden comunicarse. Durante el proceso de conexión, el cliente es asignado a un número de puerto, y enlaza un socket a ella. El cliente habla al servidor escribiendo sobre el socket y obtiene información del servidor cuando lee de él. Similarmente, el servidor obtiene un nuevo número de puerto local (necesita un nuevo puerto para poder continuar escuchando la petición de conexión del puerto original.) El servidor también enlaza un socket a este puerto local y se comunica con él mediante lectura y escritura.

El cliente y el servidor deben ponerse de acuerdo sobre el protocolo, esto es, deben de hablar el mismo lenguaje para transferir la información de vuelta a través del socket.

4.2 Análisis de la comunicación entre el cliente y el servidor de correo electrónico

Puesto que contamos con 2 servidores, uno de los cuales contiene la información de la base de datos y la interfaz gráfica y el otro contiene las cuentas de las claves de correo electrónico, esto con la finalidad de evitar saturar la carga de trabajo del equipo con el que contamos, es necesario que definamos de que manera la máquina cliente se va a comunicar con el servidor de correo electrónico.

Para esto tenemos que establecer claramente que es lo que el sistema tiene que hacer.

Algunas de las características que requerimos son:

- Que el servidor esté ejecutándose como demonio [39].
- Que el servidor reciba una petición del cliente y genere una respuesta a la petición solicitada.
- Que el cliente solicite una petición al servidor y sea capaz de interpretar la respuesta generada por el servidor.
- Que la comunicación entre el cliente y el servidor sea segura.
- Que sólo clientes autorizados puedan tener comunicación con el servidor.

Nos referimos a que el servidor este ejecutándose como demonio, ya que debe estar en espera de alguna petición desde un cliente y quedar libre para recibir otra petición inmediatamente, esto lo podemos lograr si nuestro servidor libera el puerto de comunicación por el cual está entrando la petición del cliente, que va a ser el mismo para todos los clientes, y abre un puerto desocupado en ese momento para la comunicación, de esta forma el puerto por el que escucha el servidor quedará listo para recibir una nueva petición.

Por otro lado, el servidor debe reconocer que petición le está haciendo el cliente, realizar la tarea solicitada y generar una respuesta adecuada los requerimientos hechos. Hay que considerar que el servidor puede realizar varias tareas.

Adicionalmente el cliente que realice una petición al servidor tiene que analizar la respuesta que éste genera y dependiendo del tipo de respuesta deberá mostrar la información al cliente de manera ordenada, de acuerdo con lo que solicitó.

Una de las cosas en las que debemos poner más empeño es en la seguridad de la comunicación, debemos garantizar que la información que viaja entre el cliente y el servidor y viceversa estará segura y que ningún medio o agente externo podrá modificar o hacer uso de ella.

---

[39] ***Demonio**. Es un programa que se encuentra residente en memoria y que se ejecuta cada vez que es requerido.*

Una consideración muy importante para un esquema de seguridad seria que el servidor sólo permitiera el acceso de clientes que nosotros definiéramos, así como tener un registro de que cliente se conectó y que tarea solicitó, de esta manera podemos tener un mayor control de lo que está haciendo cada cliente, y utilizar esta información para generar un listado de las actividades realizadas en determinado tiempo para llevar una estadística.

4.2.1 Análisis Estructurado

Hemos decidido trabajar con un esquema de análisis estructurado por lo que es necesario incluir herramientas como:

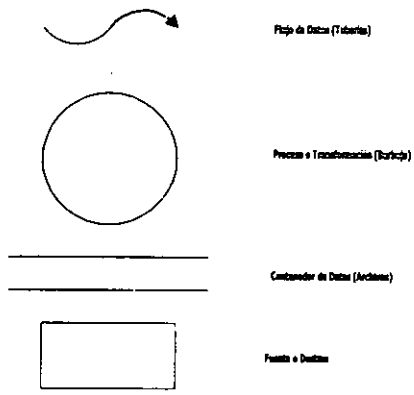
- Diagramas de Flujo de Datos (D.F.D.)
- Anotaciones en un Diccionario de Datos
- Mini especificaciones
- Definición de Archivos.
- Estructura Lógica de Archivos.
- Numeración de Censos Físicos.
- Censo de Accesos Físicos.
- Censo de Lecturas Lógicas.
- Censo de Escrituras Lógicas.

Diagramas de Flujo de Datos (D.F.D.)

En este tipo de diagramas, la estructura de la solución debe corresponder con la estructura del problema.

Para lograrlo tenemos que hacer derivar el flujo de datos a través de todo el problema.

Los elementos que componen un D.F.D. son los siguientes:



Caracteristicas de los D.F.D.

1) Es un modelo del sistema desde el punto de vista de los datos.
2) No representa temporalidad (como si todo sucediera al mismo tiempo).
3) No existen secuencias de control.
4) Ningún proceso genera datos (lo que entra debe salir, pero transformado).
5) No existen ciclos o loops.

Existen 3 tipos de D.F.D.

1. Híbrido: contiene flujos de datos tanto físicos como lógicos.
2. Físico: contiene solamente flujos de datos físicos.
3. Lógico: contiene solamente flujos de datos lógicos.

A continuación se incluyen los D.F.D. de nuestro programa cliente y servidor.

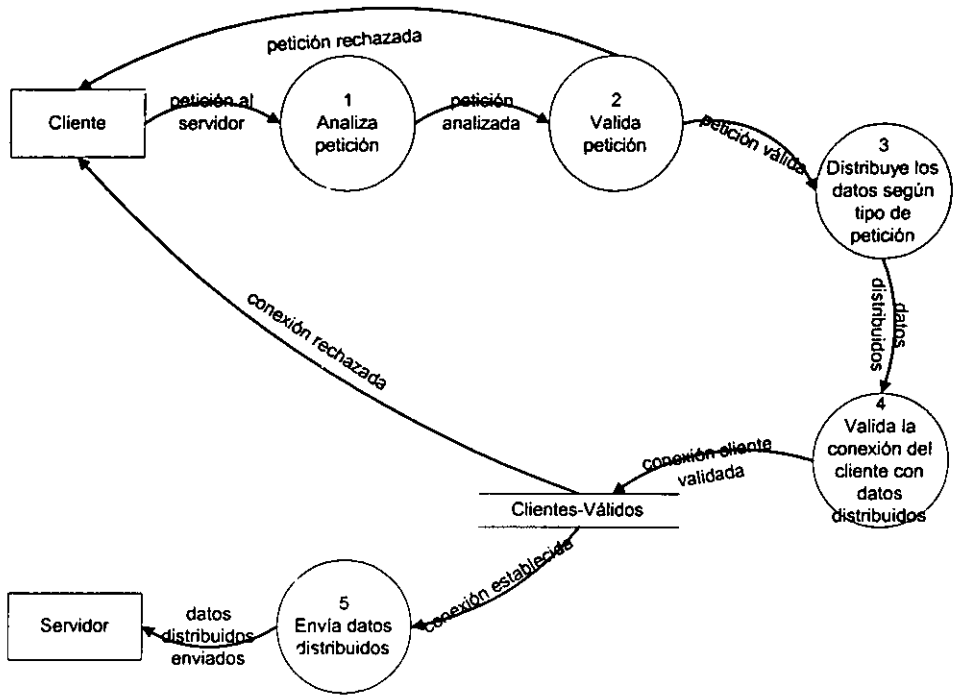


Fig. 4.9

D.F.D. del cliente

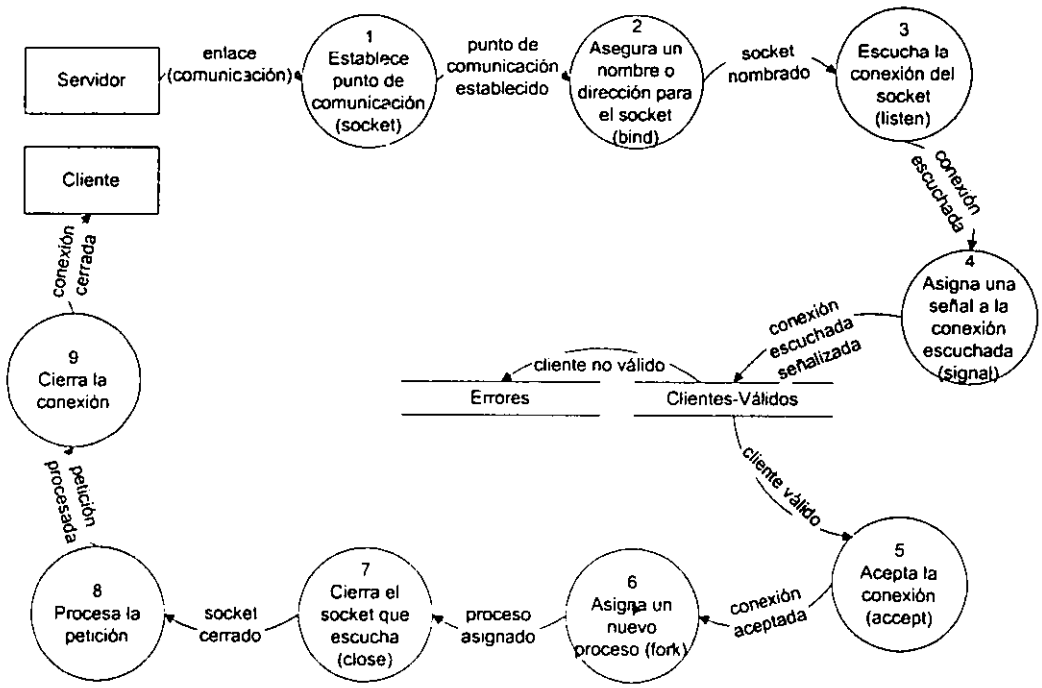


Fig. 4.10
D.F.D del servidor

4.2.2 Diccionario de Datos

Incluye todas las definiciones de los flujos de datos.

Notación:	Así se lee:
=	"se compone de"
+	"junto con"
[ ]	"seleccionar uno de"
{ }	"alteraciones o repeticiones de"
( )	"opcional"

Fig. 4.11

4.2.2.1 Diccionario de Datos del Cliente.

Petición al Servidor = Tipo-Petición
Petición analizada = Tipo-Petición + Identificación-Cliente
Identificación-Cliente = Nombre-Cliente + Dirección-IP-Cliente
Tipo-Petición = "alta" 1
"baja" 2
"cambio" 3
"renovar" 4
"passwd" 5
"reactiva" 6
"suspende" 7
Petición-Válida = Tipo-Petición + Identificación-Cliente + Indicador-Válido
Petición-Rechazada = Tipo-Petición + Identificación-Cliente + Indicador-No-Válido
Datos-Distribuidos = Tipo-Petición + Login + (Comentario) + (Fecha-Expiración) + (Password) + Actualización-Bitácora
Conexión-Cliente-Validada = Identificación-Cliente + Datos-Distribuidos
Conexión-Establecida = Conexión-Cliente-Validada + Autorización-Cliente
Conexión-Rechazada = Conexión-Cliente-Validada + Rechazo-Cliente
Autorización-Cliente = Identificación-Cliente + Indicador-Cliente-Válido
Rechazo-Cliente = Identificación-Cliente + Indicador-Cliente-No-Válido
Datos-Distribuidos-Enviados = Datos-Distribuidos

Miniespecificaciones

Consiste en la definición de los procesos del D.F.D.

Proceso 1. Analiza Petición.
Contenido: Por cada petición verifica si Tipo-Petición existe.

Proceso 2. Valida Petición.
Contenido: Por cada petición analizada verifica que Tipo-Petición sea válida y exista. De otro modo "Petición-Rechazada".

Proceso 3. Distribuye los Datos según Tipo de Petición.
Contenido: Por cada petición válida organiza los datos de acuerdo al tipo de petición, teniendo como salida los datos distribuidos.

Procesos 4. Valida la conexión del Cliente con Datos Distribuidos.
Contenido: Por cada paquete de datos distribuidos verifica que estos datos y el cliente sean válidos para que el servidor los pueda entender, de manera que se pueda establecer una conexión exitosa. De otro modo se rechaza la conexión.

Proceso 5. Envía datos distribuidos.

Contenido: Por cada conexión establecida envía los datos, ya organizados, hacia el servidor.

Definición de Archivos

Nombre del archivo: Clientes-Válidos
Alias: Clientes
Composición: {Dirección-IP-Cliente}
Organización: Secuencial, indexado por la Dirección IP del cliente.
Notas: Archivo de clientes autorizados

Nombre del archivo: Mail_Server
Alias: Servidor
Composición: Dirección-IP-Servidor
Organización: Secuencial
Notas: Archivo en el que se especifica a que máquina va a hacer las peticiones el cliente.

Estructura Lógica de Archivos

Archivo Clientes-Válidos: { Dirección-IP-Cliente }

Archivo Mail_Server: Dirección-IP-Servidor

Numeración de Censos Físicos

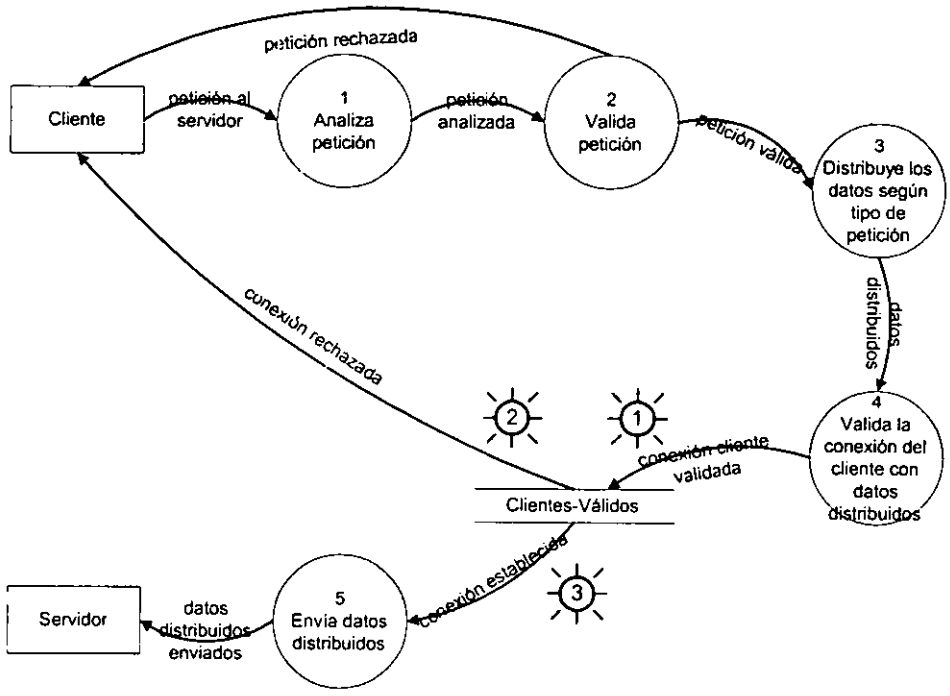


Fig. 4.12

Censo de Accesos Físicos

Acceso	Proceso	Lectura/Escritura	Propósito del flujo de datos.

1	4	Lectura	Consultar si el cliente es válido para procesar su petición.
2	4	Lectura	Consulta si la dirección IP del cliente existe en el archivo Clientes-Válidos. De no ser así, rechaza la conexión.
3	5	Lectura	Consulta si el cliente es válido para establecer la conexión con el servidor.

Fig. 4.13

Censo de Lecturas Lógicas

Acceso	Llave	Flujo de datos lógicos
1	Identificación-Cliente	Dirección-IP-Cliente
2	Identificación-Cliente	Dirección-IP-Cliente + Indicador-Cliente-no-válido
3	Identificación-Cliente	Dirección-IP-Cliente + Indicador-Cliente-válido

Fig. 4.14

4.2.2.2 Diccionario de Datos del Servidor

enlace (comunicación) = Puerto-Principal + Dirección-IP-Servidor

punto de comunicación establecido = enlace + Nombre-Socket

socket nombrado = Nombre-Socket + Tamaño-Socket + Dirección-IP-Servidor

conexión escuchada = socket nombrado + Tipo-conexión
Nombre-Socket = Familia-Socket + Tipo-Socket + Protocolo

conexión escuchada señalizada = conexión escuchada + señal

cliente-válido = Identificación-Cliente + Indicador-cliente-válido

cliente-no-válido = Identificación-Cliente + Indicador-cliente-no-válido

conexión aceptada = socket nombrado + Dirección-IP-Cliente + Tamaño-Socket

proceso asignado = Puerto-Auxiliar + conexión aceptada

socket cerrado = Nombre-Socket + Puerto-Principal + Indicador-Fin-Transmisión

petición procesada = conexión aceptada + Tipo-petición

Tipo-petición = "alta" 1
"baja" 2
"cambio" 3
"renovar" 4
"passwd" 5
"reactiva" 6
"suspende" 7

conexión cerrada = proceso asignado + Indicador-Fin-Transmisión

Miniespecificaciones

Proceso 1 Establece punto de comunicación (socket)

Contenido: Para cada enlace verifica si el puerto principal esta libre y proporciona en punto para la comunicación entre el servidor y el cliente.

Proceso 2. Asegura un nombre o dirección para el socket (bind)
Contenido: Para cada punto de comunicación establecido le liga o enlaza un nombre o dirección al socket.

Proceso 3. Escucha la conexión del socket (listen)
Contenido: Para cada socket nombrado verifica el tipo de conexión solicitada.

Proceso 4. Asigna una señal a la conexión escuchada (signal)
Contenido: Para cada conexión escuchada asigna un señal para ese proceso.

Proceso 5. Acepta la conexión (accept)
Contenido: Para cada cliente válido acepta una conexión, de manera que puede tener varias conexiones al mismo tiempo.

Proceso 6. Asigna un nuevo proceso (fork)
Contenido: Para cada conexión aceptada crea un nuevo proceso y lo levanta en un puerto distinto.

Proceso 7. Cierra el socket que escucha (close)
Contenido: Para cada proceso asignado libera el puerto principal de modo que pueda recibir otra petición por ese puerto. Asigna un nuevo puerto (puerto secundario) a cada proceso asignado.

Proceso 8. Procesa petición
Contenido: Para cada socket cerrado procesa la petición en otro puerto, según el tipo de petición.

Proceso 9. Cierra conexión (close)
Contenido: Para cada petición procesada libera o cierra el puerto secundario de conexión, de manera que este puede ser utilizado por otro proceso.

Definición de Archivos

Nombre del archivo: Clientes-Válidos
Alias: Clientes
Composición: { Dirección-IP-Cliente }
Organización: Secuencial, indexado por la dirección IP del cliente.
Notas: Archivo en el que se especifican las direcciones IP de los clientes que pueden acceder al servidor utilizando este servicio.

Nombre del archivo: Errores
Alias: Error
Composición: { Mensaje + Tipo-Error }
Organización: Secuencial
Notas: Archivo en el que se guardan los tipos de error que manda el sistema al momento de procesar una petición inválida.

Nombre del archivo: server.log
Alias: Bitácora
Composición: { Fecha + Hora + Dirección-IP-Cliente + Puerto + Nombre-Cliente + Nombre-Host + Tipo-petición + login + estado-proceso }
Organización: Secuencial, indexado por fecha y hora.
Notas: Archivo en el que se registran todas las peticiones que se le han hecho al sistema

Estructura Lógica de Archivos

Aquí se hace la declaración de los datos completos de los archivos del sistema.

Archivo Clientes-Válidos: { Dirección-IP-Cliente }

Archivo Errores: { Mensaje-Error + Tipo-Error }

Mensaje-Error: Cadena de mensaje + (causa-error)

Tipo-Error = (Número-Error) + Identificador-Error

Archivo server.log: { Fecha + Hora+ Dirección-IP-Cliente + Puerto + Nombre-Cliente + Nombre-Host + Tipo-petición + login + Estado-proceso }

Fecha = Mes + Día + Año

Estado-proceso = "done" 1
"failed" 2

Hora = Horas + Minutos + Segundos

Numeración de Censos Físicos

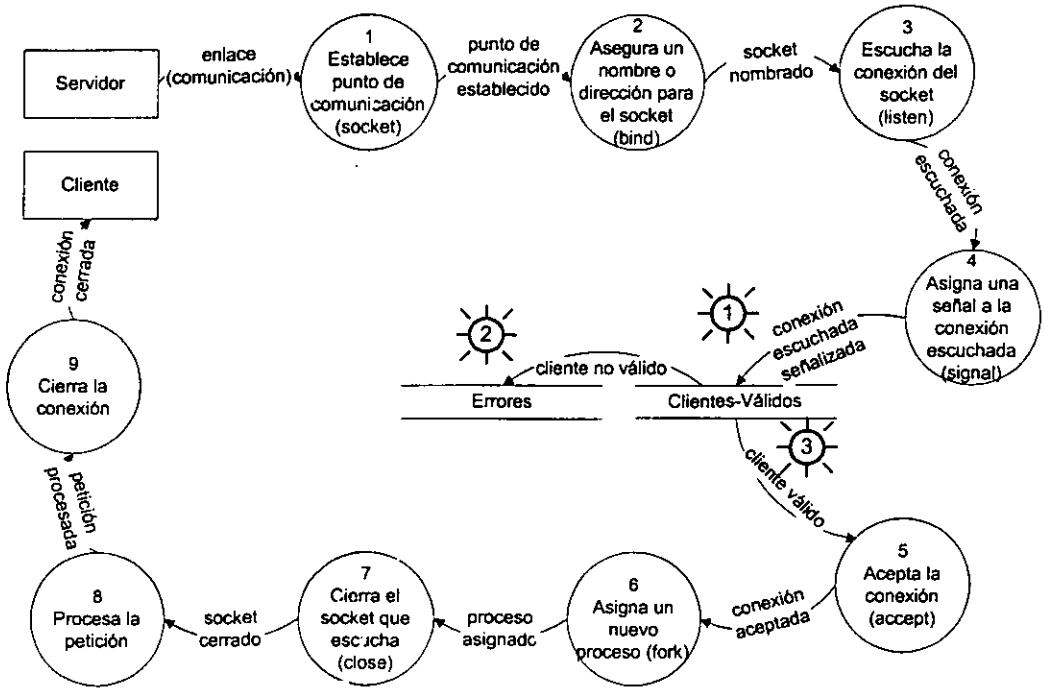


Fig. 4.15

Censo de Accesos Físicos

Acceso	Proceso	Lectura/Escritura	Propósito del flujo de datos.
1	4	Lectura	Consulta si el cliente que hace la petición esta autorizado para solicitarla.
2	4	Escritura	Rechaza la conexión si no se trata de un cliente válido para el uso del sistema.
3	5	Lectura	Permite la conexión con el cliente, ya que este ha sido validado.

Fig. 4.16

Censo de Lecturas Lógicas

Acceso	Llave	Flujo de datos lógicos
1	Identificación-Cliente	Dirección-IP-Cliente
3	Identificación-Cliente	Dirección-IP-Cliente + Indicador-Cliente-válido

Fig. 4.17

Censo de Escrituras Lógicas

Acceso	Llave	Flujo de datos lógicos
2	Identificación-Cliente	Dirección-IP-Cliente + Indicador-Cliente-no-válido + Tipo-Error

Fig. 4.18

4.3 Diseño del socket

Para esta sección vamos a utilizar el método del diseño estructurado.

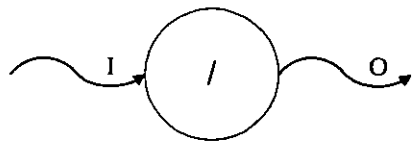
4.3.1 Diseño Estructurado.

El diagrama que representa al diseño estructurado es el diagrama de estructura de datos (D.E.D.), también conocido con el nombre de carta de estructura.

Diagrama de Estructura de Datos

El recorrido general de un D.E.D. es de la siguiente manera:

La técnica I/O es de forma triangular.



Debe convertirse en :

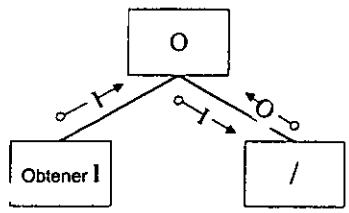


Fig. 4.19

Sistema de Empaquetamiento

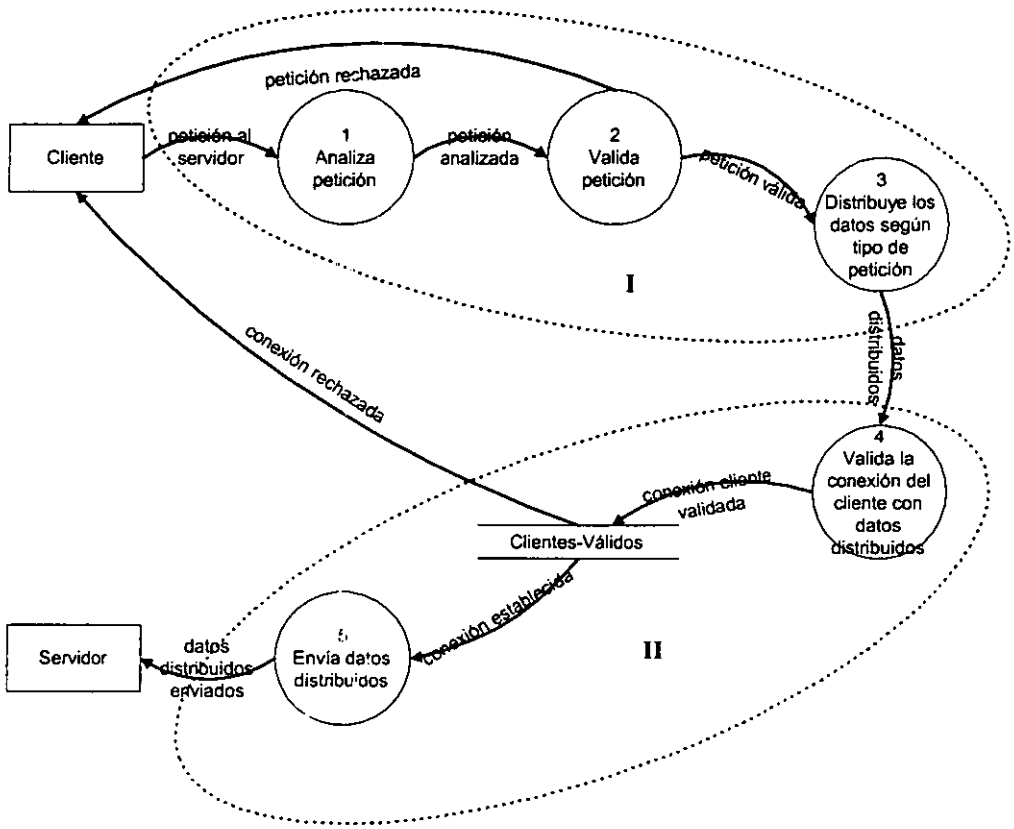


Fig. 4.20

Análisis de Transformación

PAQUETE I

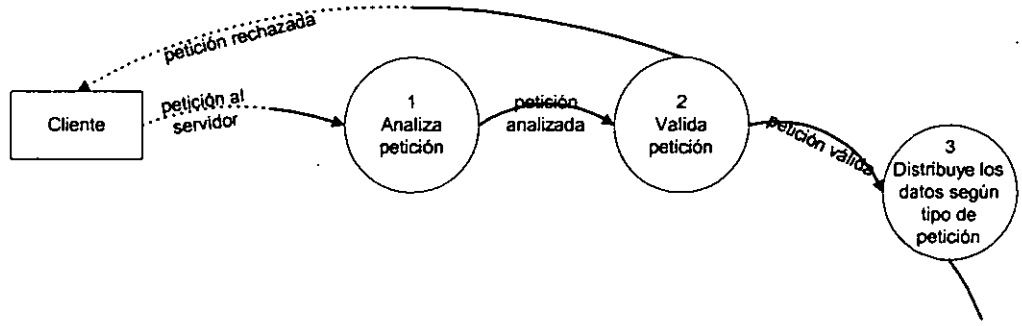


Fig. 4.21

PAQUETE II

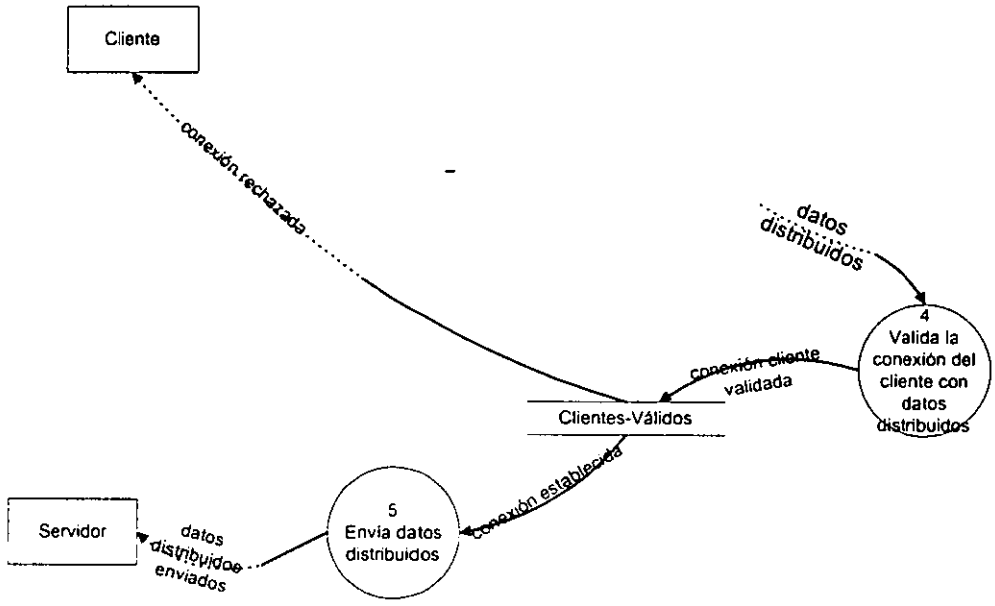


Fig. 4.22

Diagrama de Estructura de Datos del Cliente

TRABAJO I

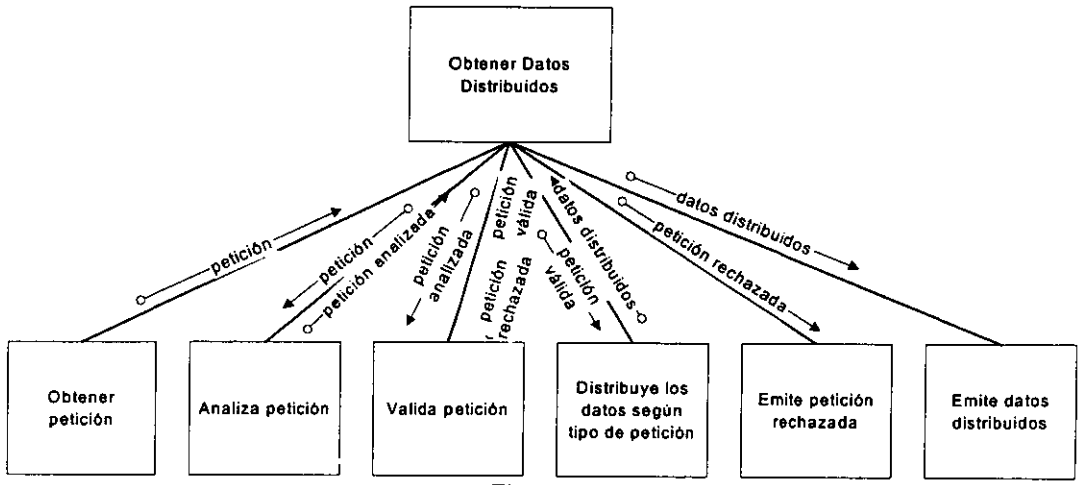


Fig. 4.23

TRABAJO II

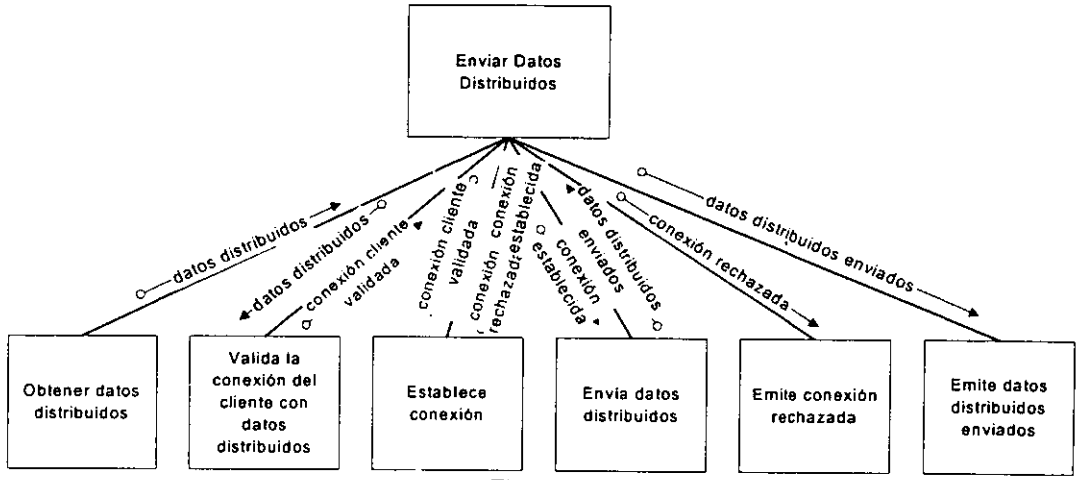


Fig. 4.24

Sistema de Empaquetamiento

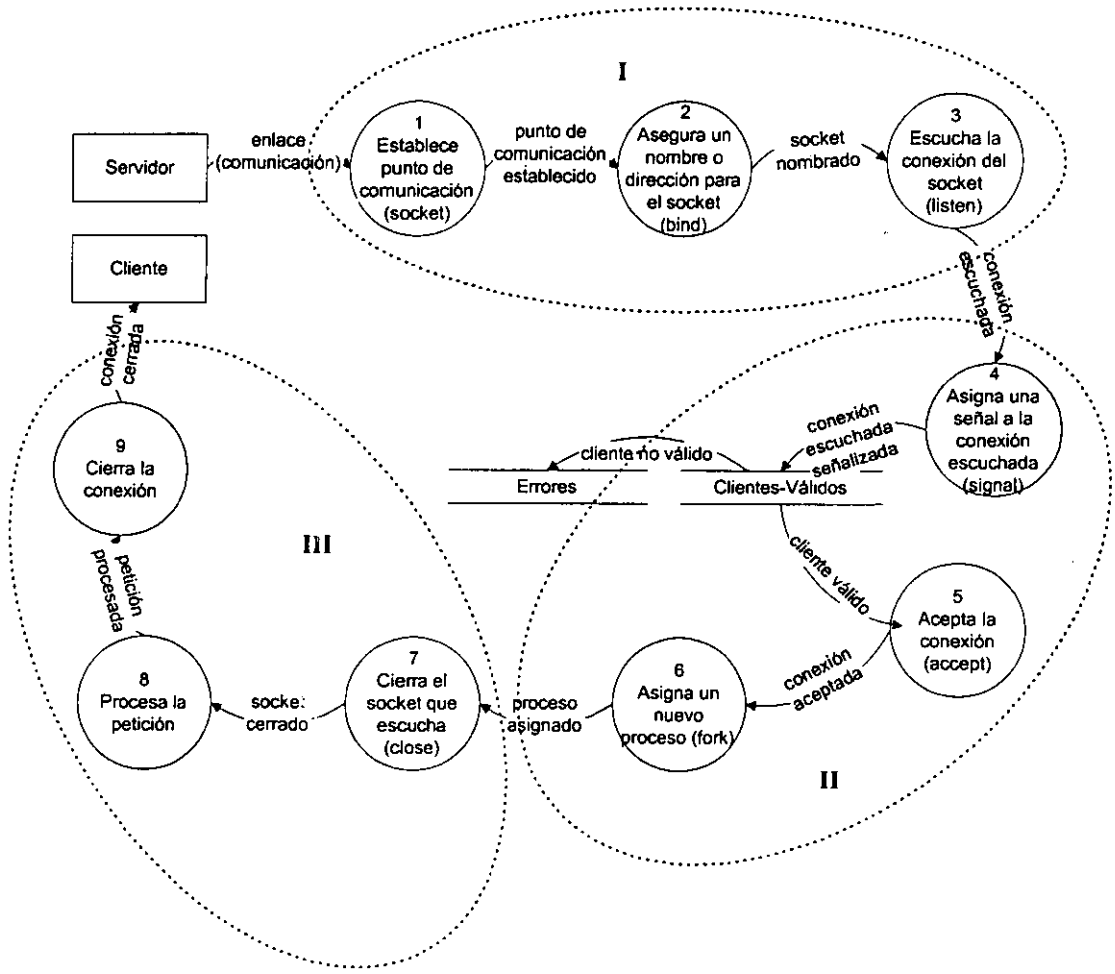


Fig. 4.25

Análisis de Transformación

PAQUETE I

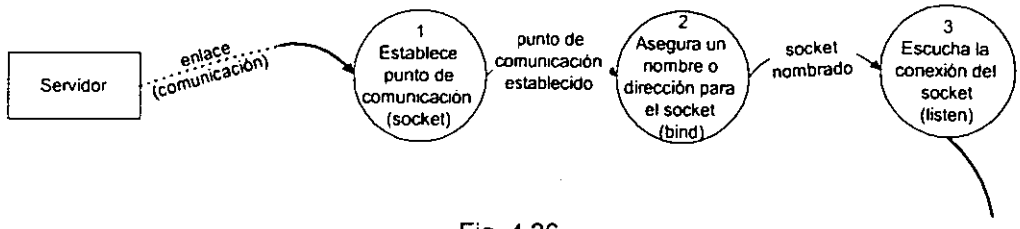


Fig. 4.26

PAQUETE II

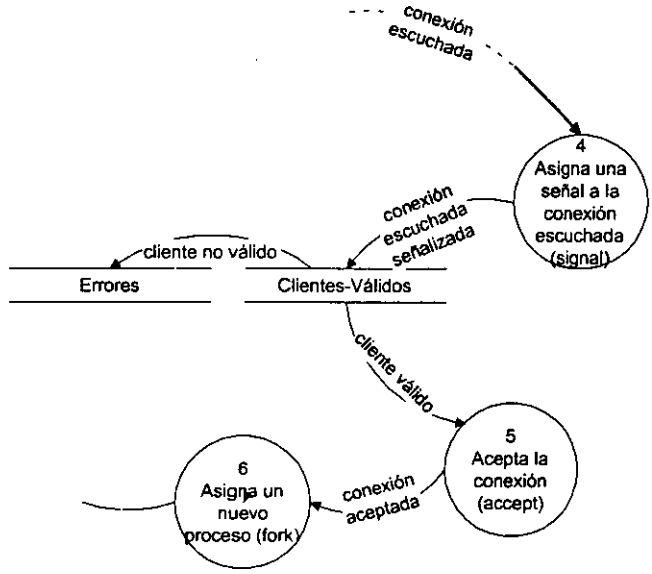


Fig. 4.27

PAQUETE III

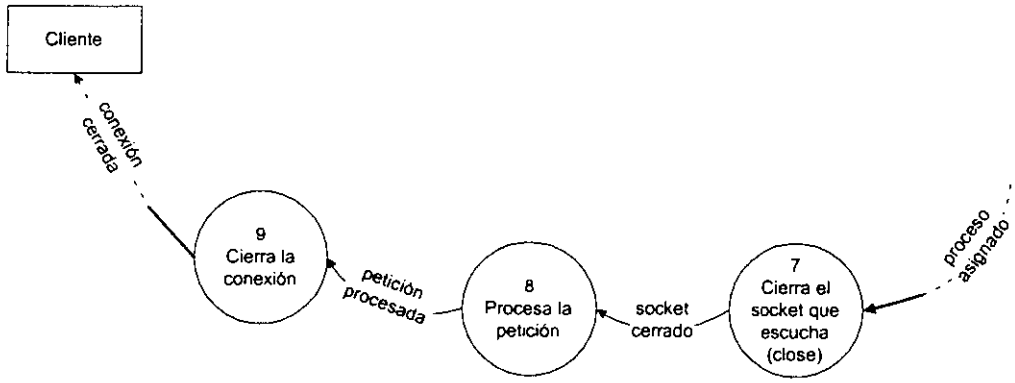


Fig. 4.28

Diagrama de Estructura de Datos del Servidor

TRABAJO 1

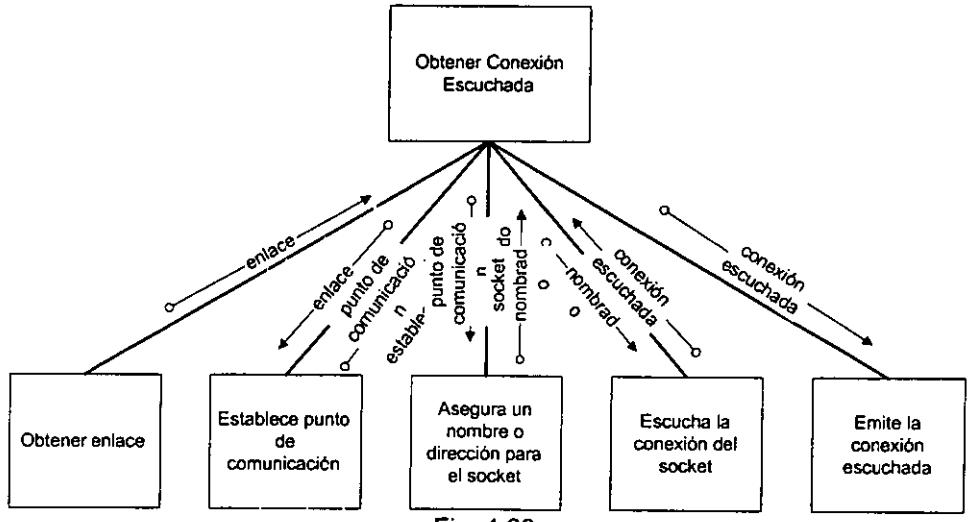


Fig. 4.29

TRABAJO 2

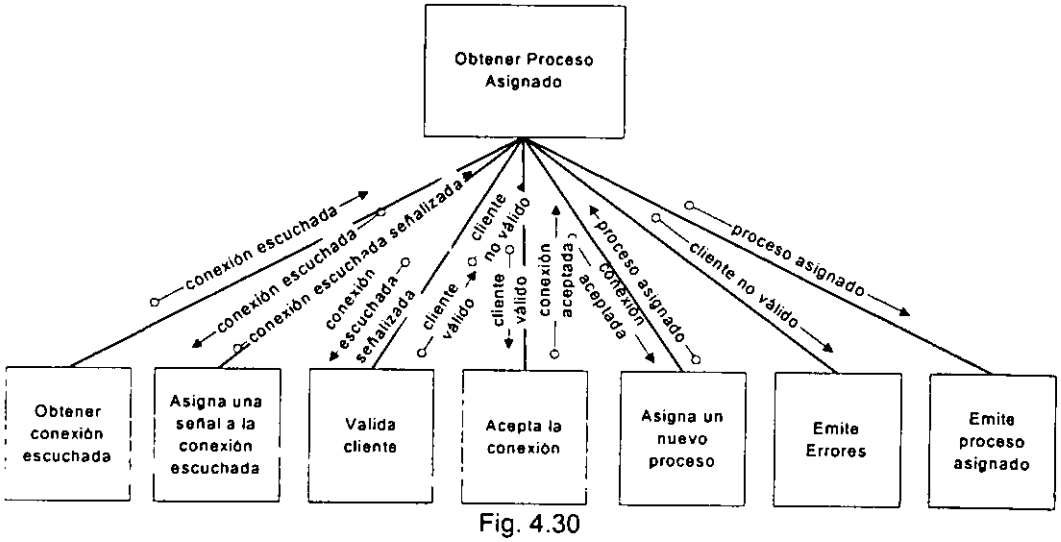


Fig. 4.30

TRABAJO 3

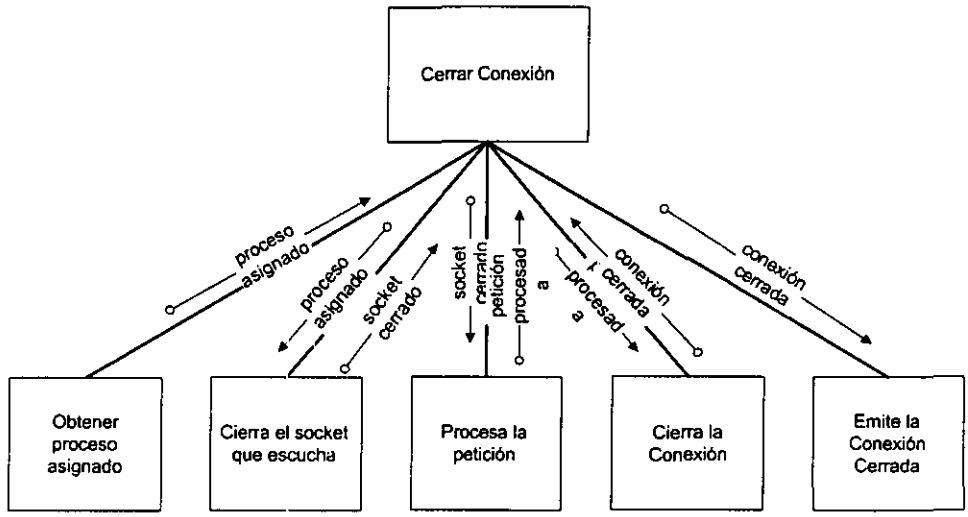


Fig. 4.31

4.4 Programación del socket [40]

Antes que nada, analizamos el lenguaje de programación que deberíamos utilizar para la codificación del socket. En dicho análisis tomamos en cuenta los conocimientos y experiencia que teníamos en los lenguajes de programación que habíamos utilizado en nuestras tareas escolares y nuestro trabajo, por lo que decidimos programarlo en lenguaje C, ya que es conocido y manejado ampliamente por nosotros, además de que nos permite trabajar en un ambiente Unix, que a final de cuentas es en donde va a correr el servidor.

A continuación presentamos una breve historia del lenguaje escogido y algunas de sus características.

EL lenguaje C fue obtenido de un proceso de desarrollo que inició con un lenguaje denominado BCPL. De aquí surgió otro llamado B, que fue inventado por Ken Thompson. Ya, en los años 70; éste lenguaje llevó a la aparición del lenguaje C.

Con la popularidad de las microcomputadoras muchas compañías comenzaron a implementar su propio C por lo cual surgieron discrepancias entre sí.

Por esta razón **ANSI** (American National Standars Institute), estableció un comité en 1983 para crear una definición no ambigua del lenguaje C e independiente de la máquina que pudiera utilizarse en todos los tipos de C.

C es un lenguaje de programación de nivel medio ya que combina los elementos del lenguaje de alto nivel con la funcionalidad del ensamblador.

Su característica principal es que es portable, es decir, es posible adaptar los programas escritos para un tipo de computadora en otra.

Otra de sus características principales es el ser estructurado, es decir, el programa se divide en módulos (funciones) independientes entre sí.

[40] Stevens Richard W. , UNIX Network Programming. Networking APIs: Sockets and XTI Volume 1. Chapter 4

A continuación presentamos la descripción de algunas de las funciones que vamos a utilizar en la programación del socket.

En la siguiente figura se muestran las funciones elementales para un socket Cliente-Servidor de TCP.

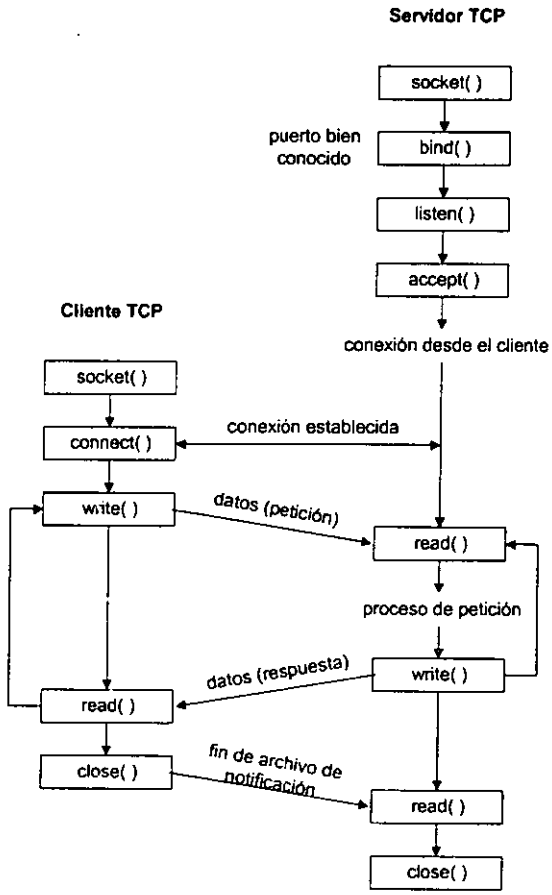


Fig. 4.32

Función socket

La primera cosa que un proceso debe hacer es llamar a la función socket, especificando el tipo de protocolo de comunicación deseado (TCP usando IPv4, UDP usando IPv6, etc).

#include <sys/socket.h> // Librería proporcionada por el compilador de C para UNIX.

int socket (int *family*, int *type*, int *protocol);*

Regresa : un valor positivo si es exitoso y un -1 si ocurre un error

El valor *family* especifica la familia del protocolo y es una de las constantes que se muestran en la siguiente figura :

family	Descripción
AF_INET	Protocolos IPv4

AF_INET6	Protocolos IPv6
AF_LOCAL	Protocolos de dominio Unix
AF_ROUTE	Routing sockets
AF_KEY	Key sockets

Fig. 4.33

El tipo *(type)* de socket es una de las constantes que se muestran en la siguiente figura. Normalmente el argumento de protocolo en la función **socket** es puesto en cero (0) excepto para *raw sockets (sockets para la comunicación entre IPv4 o IPv6 en forma directa).*

Type	Descripción
SOCK_STREAM	stream socket
SOCK_DGRAM	datagram socket
SOCK_RAW	raw socket

Fig. 4.34

No todas las combinaciones de los sockets family y tipos (types) son validas. En la siguiente figura se muestran las combinaciones validas , a través del protocolo actual este es seleccionado a la par. Las campos marcados como "SI", son válidos pero no tienen siglas a manejar.

	AF_INET	AF_INET6	AF_LOCAL	AF_ROUTE	AF_KEY
SOCK_STREAM	TCP	TCP	SI		
SOCK_DGRAM	UDP	UDP	SI		
SOCK_RAW	IPv4	IPv6		SI	SI

Fig. 4.35

Función connect

La función connect es usada por el cliente de TCP para establecer una conexión con el servidor TCP .

```
#include <sys/socket.h>

int connect (int sockfd, const struct sockaddr *servaddr, socklen_t addrlen);
```

Regresa : cero (0) si es exitoso y un -1 si ocurre un error .

sockfd es un descriptor de un socket que es regresado por la función socket. El segundo y tercer argumento son punteros a una estructura de dirección del socket . La estructura de dirección del socket debe contener una dirección IP y un número de puerto de el servidor.

El cliente no tiene que llamar a **bind** antes que se llama a connect : el kernel escogerá para ambos un puerto efímero y la dirección IP fuente si es necesario .

En el caso del socket TCP , la función regresa sólo cuando la conexión es establecida o en el caso de que ocurra un error.

Función bind

La función **bind** asigna una dirección del protocolo local a un socket . Con los protocolos de Internet , la dirección del protocolo es la combinación de cualquiera de los 32 bits de una dirección IPv 4 o de los 128 bits de una dirección IPv6, además de un número de puerto de los 16 bits de TCP o UDP.

```
# include <sys/socket.h>

int bind(int sockfd, const struct sockaddr *myaddr, socklen_t addrlen);
```

Regresa : cero (0) si es exitoso y un -1 si ocurre un error .

El segundo argumento es un puntero a una dirección del protocolo especificado y el tercer argumento es el tamaño de la estructura de la dirección . Con TCP , la llamada bind nos deja especificar un número de puerto , una dirección IP , ambos , o ninguno .

Función listen

La función **listen** es llamada sólo por un servidor TCP y hace dos acciones.

Cuando un socket es creado por la función **socket**, este es asumido a ser un socket activo, esto es, un socket cliente que emitirá la función **connect**. La función **listen** convierte un socket que no está conectado en un socket pasivo, indicándole al kernel que debe aceptar peticiones dirigidas a este socket.

El segundo argumento de esta función especifica el número máximo de conexiones que el cliente debe encolar para este socket.

```
# include <sys/socket.h>

int listen(int sockfd, int backlog);
```

Regresa : cero (0) si es exitoso y un -1 si ocurre un error .

Esta función normalmente es llamada después de las funciones **socket** y **bind** y debe ser llamada antes de la función **accept**.

Función accept

La función **accept** es llamada por un servidor TCP para regresar la siguiente conexión completada desde la cola de conexiones.

```
# include <sys/socket.h>

int accept(int sockfd, struct sockaddr *cliaddr, socklen_t *addrlen);
```

Regresa : un valor positivo si es exitoso y un -1 si ocurre un error .

Los argumentos *cliaddr* y *addrlen* son usados para regresar la dirección del protocolo del proceso conectado (cliente).

Si **accept** es exitoso, regresa un valor que es automáticamente creado por el kernel. Este valor se refiere a la conexión TCP con el cliente.

Las funciones fork y exec

La función **fork** es el único camino en Unix para crear un proceso nuevo.

```
# include <unistd.h>

pid_t fork(void);
```

Regresa : cero (0) en el proceso hijo, el identificador del proceso del hijo en el proceso padre y un -1 si ocurre un error .

La razón por la cual **fork** regresa un cero (0) en el proceso hijo en lugar del identificador del proceso padre es porque el hijo sólo tiene un padre y siempre va a obtener el identificador de proceso del padre mediante **getppid**.

Hay dos usos típicos de **fork**.

Un proceso hace una copia de sí mismo para que una copia pueda manejar una operación mientras la otra hace otra tarea. Esto es típico en los servidores de red.

Un proceso quiere ejecutar otro programa. Desde que el único camino para crear un proceso nuevo es llamando a **fork**, el proceso primero llama a **fork** para hacer una copia de sí mismo y luego una de las copias llama a la función **exec** para reemplazarse con el programa nuevo. Esto es típico para los programas como los shells.

La única forma en la que un archivo ejecutable en el disco es ejecutado por Unix es mediante el uso de cualquiera de la seis modalidades de la función **exec**.

```
# include <unistd.h>

int execl(const char *pathname, const char *arg0, ... /* (char *) 0 */);

int execv(const char *pathname, char *const argv[ ]);

int execle(const char *pathname, const char *arg0, ... /* (char *) 0, char *const envp[ ] */);

int execve(const char *pathname, char *const arg[ ], char *const envp[ ]);

int execlp(const char *pathname, const char *arg0, ... /* (char *) 0 */);

int execvp(const char *pathname, char *const argv[ ]);
```

Regresan : -1 si ocurre un error .

Estas funciones regresan -1 solo en el caso de que exista un error. En cualquier otro caso el control pasa al inicio del nuevo programa, normalmente la función main.

Función close

La función normal **close** en Unix es también usada para cerrar un socket y terminar una conexión de TCP.

```
# include <unistd.h>

int close(int sockfd);
```

La acción por defecto de la función **close** con un socket de TCP es marcarlo como cerrado y regresar al proceso inmediatamente.

Las funciones getsockname y getpeername

Estas dos funciones regresan la dirección del protocolo local asociada con un socket (**getsockname**) o la dirección del protocolo remoto asociado con un socket (**getpeername**).

```
# include <sys/socket.h>

int getsockname(int sockfd, struct sockaddr *localaddr, socklen_t *addrlen);

int getpeername(int sockfd, struct sockaddr *peeraddr, socklen_t *addrlen);
```

Ambas regresan : cero (0) si es exitoso y un -1 si ocurre un error .

Estas dos funciones son requeridas por las siguientes razones:

- Después de que **connect** regresa un valor exitoso en un cliente TCP que no llama a **bind**, **getsockname** regresa la dirección IP y número de puerto locales asignados a la conexión por el kernel.
- Después de haber llamado a **bind** con el número de puerto 0, **getsockname** regresa el número de puerto local que fue asignado.
- **getsockname** puede ser llamado para obtener la dirección de la familia de un socket.
- En un servidor TCP, una vez que hay establecida una conexión con un cliente, el servidor puede utilizar **getsockname** para obtener la dirección IP local asignada a la conexión.
- Cuando un servidor es ejecutado por el proceso que llama la función **accept**, el único camino por el cual el servidor puede obtener la identidad del cliente es utilizando la función **getpeername**.

Al tener ya nuestra base de datos y la interfaz para el usuario definidas, en éste capítulo se explicaron las herramientas y la forma de cómo se enlazarán con el servidor de cuentas de correo electrónico a través de la red (medio de enlace) y los protocolos que se utilizarán así como sus características. Se mencionó también las condiciones que deben cumplir tanto de parte del cliente como de el servidor para que ésta comunicación se realice correctamente.

CAPITULO 5

INTEGRACIÓN DE LOS COMPONENTES DEL SISTEMA

CAPITULO 5

Integración de los componentes del sistema

En este capítulo se explica la instalación y configuración de cada una de las herramientas que se utilizaron, así como los procesos desarrollados en la integración de este sistema.

Las herramientas utilizadas son:

- **Sybase** 11.0.3
- **Apache** 1.3.9

5.1 Instalación del servidor de Base de Datos (Sybase)

Este procedimiento consiste en lo siguiente:

Crear una cuenta de usuario para Sybase

 # useradd -u 1012 -g 10 -c "Administrador de Sybase" -s /bin/csh -d /usr/sybase -m sybase
6 blocks

Asignarle un password a esa cuenta de usuario

 # passwd sybase
New password:
Re-enter new password:
passwd (SYSTEM): passwd successfully changed for sybase

Agregar el usuario sybase al grupo sys en el archivo /etc/group

 # vi /etc/group

root::0:root
other::1:
bin::2:root,bin,daemon
sys::3:root,bin,sys,adm,sybase
adm::4:root,adm,daemon
uucp::5:root,uucp
mail::6:root
tty::7:root,tty,adm
lp::8:root,lp,adm
nuucp::9:root,nuucp
staff::10:
daemon::12:root,daemon
sysadmin::14:
nobody::60001:
noaccess::60002:
nogroup::65534:

Convertirse en el usuario sybase

 # su - sybase
Sun Microsystems Inc. SunOS 5.6 Generic August 1997

Ejecutar el comando /cdrom/sybasecd/sybsetup

% /cdrom/sybasecd/sybsetup

Aparece la siguiente pantalla en la que se indica en que directorio va a ser instalado sybase

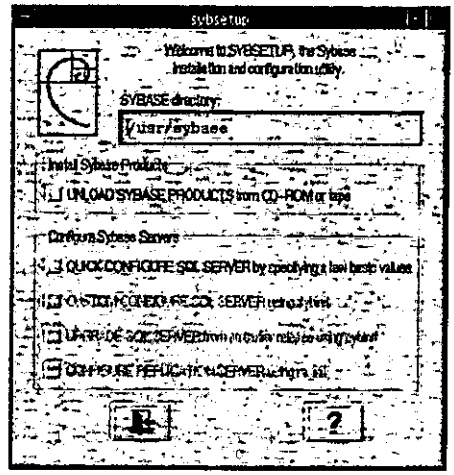


Fig. 5.1

Posteriormente aparece una pantalla en la que se confirma el directorio de instalación

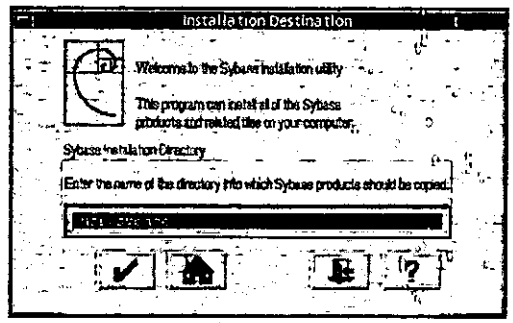


Fig. 5.2

Si el directorio de instalación no existe, pregunta para crearlo.

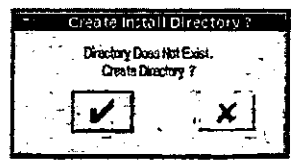


Fig. 5.3

En caso de tener la variable de ambiente SYBASE declarada aparece lo siguiente:

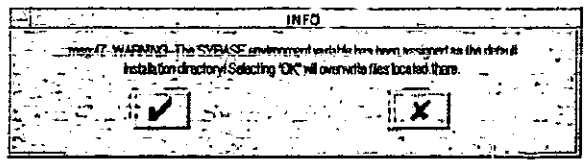


Fig. 5.4

Posteriormente pregunta cual es el dispositivo de instalación del software y donde se encuentra el archivo imagen desde donde sybase será copiado.

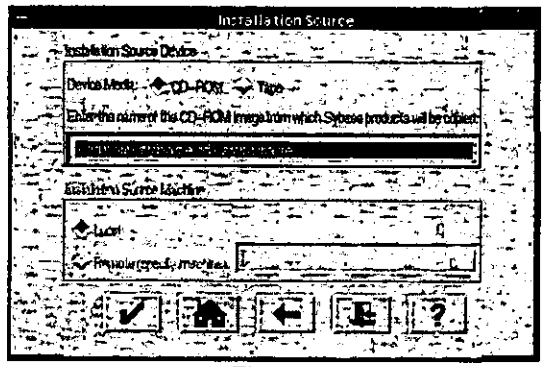


Fig. 5.5

Después solicita la Cadena de Autorización del cliente que viene con la distribución del software.

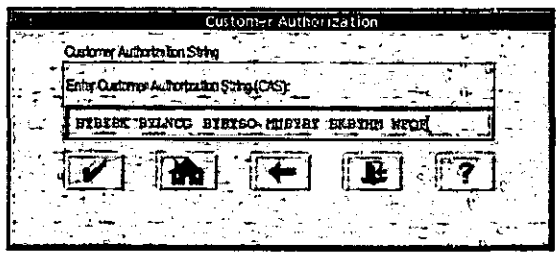


Fig. 5.6

La siguiente sección es la selección de la distribución de software del producto, aquí se pueden escoger los productos deseados para instalar.

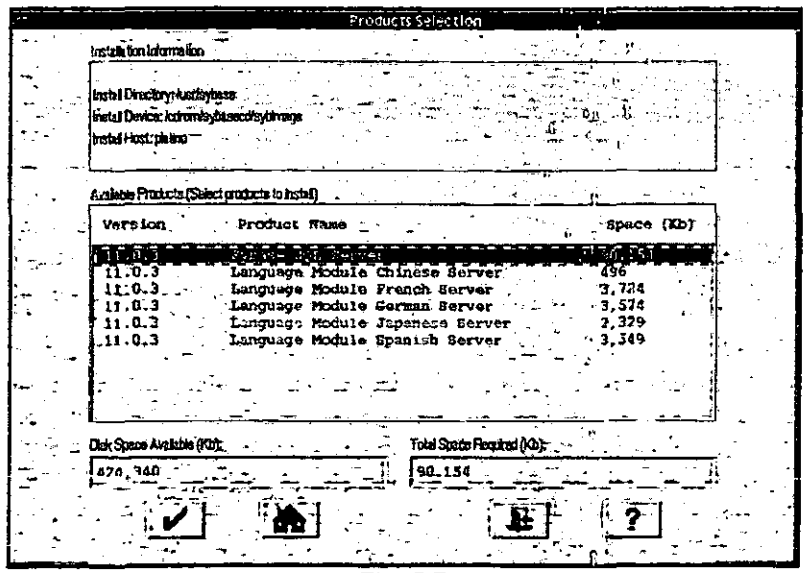


Fig. 5.7

Pide confirmación para iniciar el proceso de instalación.

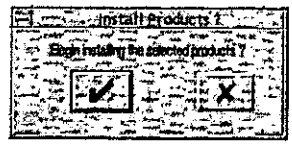


Fig. 5.8

En la siguiente pantalla se muestra la barra de estado de progreso en el proceso de instalación.

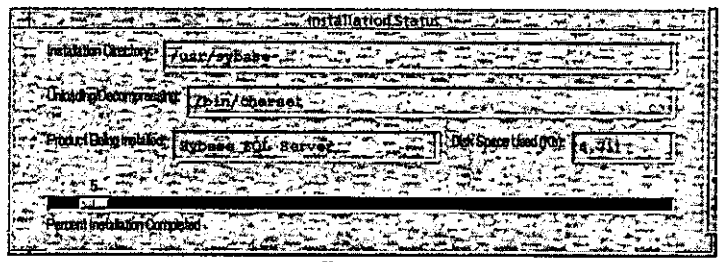


Fig. 5.9

Indica que el proceso de instalación ha finalizado.

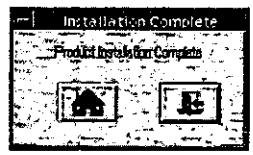


Fig. 5.10

En el paso siguiente hay que indicarle que se quiere configurar el servidor de Sybase

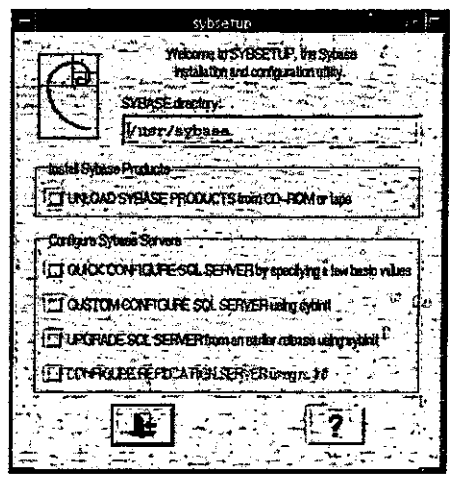


Fig. 5.11

Se utiliza este tipo de configuración para indicarle el nombre del servidor de SQL, el número de puerto por el que va a estar respondiendo, así como la ubicación y el tamaño de la bitácora de errores, la ubicación y el tamaño de los dispositivos Master y Sybsystemprocs. Lo mismo para el **BACKUP**.

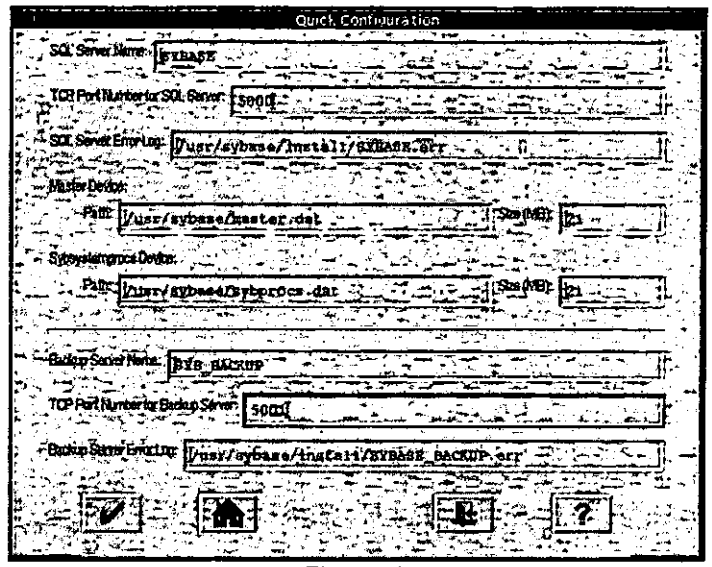


Fig. 5.12

Finalmente aparece la siguiente pantalla para indicar el estado de inicio de sybase llamado Sybinit

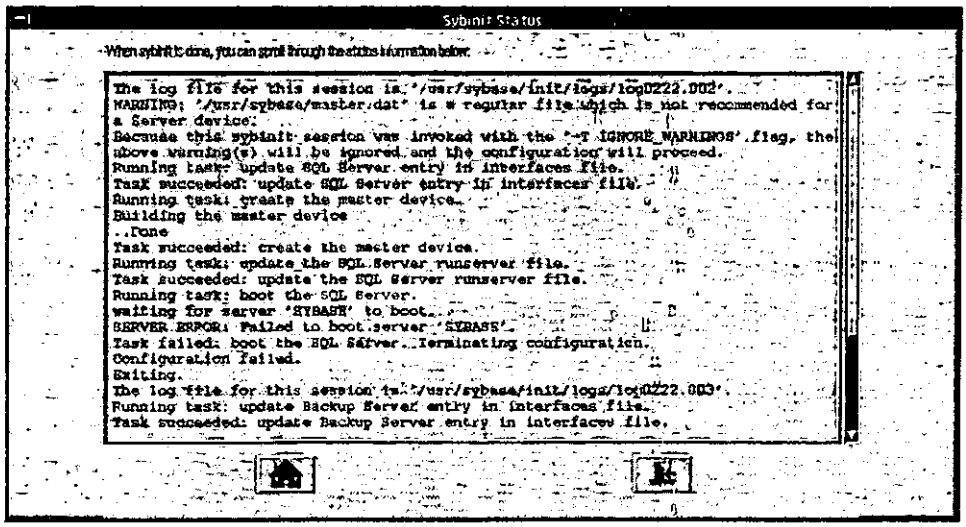


Fig. 5.13

Cuando se instala el Servidor Sybase se tienen las siguientes Bases de Datos:

- B.D. **master**
- B:D. **model**
- B.D. de procedimientos de Sistema: **sybsystemprocs**
- B.D. temporal: **tempdb**

Las Bases de datos **master**, **model** y **temporal** residen sobre el dispositivo nombrado durante la instalación, el cual es conocido como **d_master**. La Base de datos **master** esta contenida completamente sobre el dispositivo **master** y no puede ser expandida sobre cualquier otro dispositivo. Todas las demás Bases de Datos y objetos del usuario pueden ser creados sobre otros dispositivos.

La Base de Datos **model** contiene las tablas de sistema requeridas para cada Base de Datos de usuario. Esta puede ser modificada al ajustar la estructura de la Base de datos creada recientemente, cada vez que se cambia **model** será reflejado en cada nueva Base de Datos.

NOTA: Una nueva Base de Datos nunca puede ser más pequeña que la base de Datos **model**.

Los procedimientos del sistema de SYBASE son almacenados en la Base de Datos **sybsystemprocs**. Cuando un usuario en cualquier Base de Datos ejecuta cualquier procedimiento almacenado cuyo nombre empieza con los caracteres "sp_", el servidor SQL primero busca que el procedimiento este en la Base de Datos actual del usuario. Si no localiza el procedimiento con el nombre dado, el servidor SQL lo busca en **sybsystemprocs**, y en caso de tampoco estar, el servidor SQL buscará el procedimiento en **master**.

El servidor SQL tiene una base de Datos temporal, **tempdb**. Esta proporciona un área de almacenamiento para tablas temporales y otras necesidades temporales de almacenamiento. El espacio en **tempdb** es compartido entre todos los usuarios de todas las Bases de Datos sobre el servidor[41].

Activación del servidor de Bases de Datos de Sybase

Una vez que Sybase ha sido instalado es necesario iniciar el servidor, de manera que este listo para recibir peticiones, esto se logra mediante el siguiente comando, que debe ser ejecutado por el usuario sybase, ya que este usuario es el dueño de la Base de Datos en este servidor:

% /usr/sybase/install/startserver -f /usr/sybase/install/RUN_SYBASE

Por otro lado hay que ejecutar también el servidor de backup:

% /usr/sybase/install/startserver -f /usr/sybase/install/RUN_SYB_BACKUP

Declaración de las variables de ambiente o de entorno de trabajo para trabajar con un servidor de Bases de Datos de Sybase

Es importante señalar que para trabajar con un servidor de Bases de Datos de Sybase, es necesario establecer dos variables de ambiente en cada cliente, que en este caso hará uso de un programa llamado **isql** para conectarse al servidor. Las variables de ambiente son:

DSQUERY
SYBASE

La forma de declararlas es la siguiente:

en C shell

% setenv DSQUERY <nombre_del_servidor_de_Base_de_Datos>
% setenv SYBASE <directorio_en_donde_esta_instalado_Sybase>

Por lo cual para nuestro caso es:

% setenv DSQUERY SYBASE
% setenv SYBASE /usr/sybase

[41] Notas del Curso Administración de Sybase. Departamento de Innovación. DCAA. UNAM. 1999.

en Bourne o korn shell

$ DSQUERY=SYBASE; export DSQUERY
$ SYBASE=/usr/sybase; export SYBASE

Configuración de la clave del super usuario sa

Después de haber instalado sybase y de haber declarado las variables de ambiente, se recomienda asignarle un password (contaseña) a la clave de administrador de Sybase **sa** (sybase administrator), que por default no tiene. Esto se obtiene mediante lo siguiente:

Hay que hacer una conexión a Sybase mediante el comando isql, cuya sintaxis es:

% isql -U*nombre_de_usuario_de_sybase*

% isql -Usa
Password:

Existe un procedimiento almacenado que viene incluido con la instalación de Sybase para cambiar el password a **sa**, la sintaxis es:

> sp_password *password_viejo, password_nuevo*
> go

Aplicando esto al servidor se hace este procedimiento, asignando el password $ac@adm a la cuenta **sa**:

1> sp_password null,"$ac@dM14"
2> go
Password correctly set.
(return status = 0)

Como se observa, ya no es necesario especificar el usuario **sa** en la sintaxis del comando anterior, ya que cambia el password al usuario que inició la sesión, en este caso **sa**; por lo que se recomienda iniciar la sesión con el usuario al que se le quiere cambiar el password.

Creación de los dispositivos que almacenarán las tablas

Como regla general se crean 2 dispositivos para las Bases de datos, uno que es para almacenar los datos y otro para almacenar los logs, esto es, accesos, operaciones que se han ejecutado en esa Base de Datos, errores que han surgido en algún procedimiento, etc.

La sintaxis para crear los dispositivos es:

> disk init
> name=*nombre_del_dispositivo*,
> physname=*nombre_fisico_del_dispositivo*,
> vdevno=*número_del_dispositivo*,
> size=*tamaño_del_dispositivo_en_bloques*,
> go

La asignación de **size** se obtiene multiplicando el número de Mb por 512.

Los dispositivos que serán creados son de 40 Mb para datos y 10 Mb para log

Para crear el dispositivo para datos se hace lo siguiente:

```
1> disk init
2> name="sac",
3> physname="/usr/sybase/dev/sac",
4> vdevno=3,
5> size=20480
6> go
```

Para crear el dispositivo de log:

```
1> disk init
2> name="saclog",
3> physname="/usr/sybase/dev/saclog",
4> vdevno=4,
5> size=5120
6> go
```

Para visualizar los dispositivos existentes, hay un dispositivo almacenado llamado **sp_helpdevice**, que se ejecuta como se indica a continuación:

```
1> sp_helpdevice
2> go
```

device_name	physical_name	description	status	cntrltype	device_number	low	high	
master	d_master	special, default disk, physical disk, 21.00 MB	3	0	0	0	10751	
sac	/usr/sybase/dev/sac	special, physical disk, 40.00 MB	2	0	3	0331648	50352127	
saclog	/usr/sybase/dev/saclog	special, physical disk, 10.00 MB	2	0	4	67108864	67113983	
sybsecurity	/usr/sybase/sybsecur.dat	special, physical disk, 5.00 MB	2	0	2	33554432	33558991	
sysprocsdev	/usr/sybase/sybprocs.dat	special, physical disk, 16.00 MB	2	0	1	16777216	16785407	
tapedump1	/dev/rmt4	tape, 625 MB, dump device		16	3	0	0	20000
tapedump2	/dev/rst0	disk, dump device		16	2	0	0	20000

(7 rows affected, return status = 0)

Creación de la Base de Datos

Para crear la Base de datos se utiliza la siguiente sintaxis:

> create database *nombre_base_de_datos* on *dispositivo_de_datos=tamaño_en_Mb* log on *dispositivo_de_log=tamaño_en_Mb*
> go

Para crear una Base de Datos de 40 Mb para datos y 10 Mb para log se ejecuta lo siguiente:

```
1> create database sac on sac=40 log on saclog=10
2> go
CREATE DATABASE: allocating 20480 pages on disk 'sac'
CREATE DATABASE: allocating 5120 pages on disk 'saclog'
```

Existe un procedimiento almacenado para visualizar el estado de la Base de Datos que acaba de ser creada llamado **sp_helpdb**, que tiene la siguiente sintaxis:

> sp_helpdb *nombre_base_de_datos*
> go

Para ver el estado de la Base de Datos sac:

```
1> sp_helpdb sac
2> go
name            db_size         owner           dbid            created         status
```

```
        sac             50.0 MB         sa              6               Feb 28, 2000    no options
set

        device_fragments                size            usage           free kbytes
        sac                             40.0 MB         data only       40192
        saclog                          10.0 MB         log only        10224
        (return status = 0)
```

Creación de una clave de login

Para crear o agregar una clave de login, Sybase cuenta con un procedimiento almacenado que se llama **sp_addlogin**. Su sintaxis es:

> sp_addlogin *nombre_del_login, password, Base_de_datos_por_default, lenguaje_por_default, comentario_del_login*
> go

Para la aplicación del Sistema de Administración de Cuentas de correo electrónico es necesario agregar el siguiente login:

```
1> sp_addlogin sacadm,"secret",sac,null,"Administrador de SAC"
2> go
Password correctly set.
Account unlocked.
New login created.
(return status = 0)
```

Una vez creado el login, hay que cambiarle el dueño a la base de datos, esto se logra con el procedimiento almacenado **sp_changedbowner**, que se usa de la siguiente manera:

> sp_changedbowner *nombre_del_nuevo_dueño*
> go

Ahora es necesario cambiarle el dueño a la Base de Datos sac:

```
1> use sac
2> go
1> sp_changedbowner sacadm
2> go
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
Database owner changed.
(return status = 0)
```

Para verificar que realmente se le cambió el dueño a esa Base de Datos, hay que hacer una conexión con el nuevo usuario propietario de la misma y ejecutar sp_helpdb:

```
% isql -Usacadm
Password:
1> sp_helpdb
2> go
```

name	db_size	owner	dbid	created	status
master	5.0 MB	sa	1	Jan 01, 1900	no options set
model	2.0 MB	sa	3	Jan 01, 1900	no options set
sac	50.0 MB	sacadm	6	Feb 28, 2000	no options set
sybsecurity	5.0 MB	sa	5	Feb 28, 2000	trunc log on chkpt
sybsystemprocs	16.0 MB	sa	4	Feb 28, 2000	trunc log on chkpt
tempdb	2.0 MB	sa	2	Feb 28, 2000	select into/bulkcopy

(return status = 0)

Modificación del Archivo /etc/system

Para que Sybase tenga un rendimiento óptimo y solo utilice los recursos que el administrador le proporcione, es necesario modificar el archivo /etc/system, agregándole al final las siguientes líneas:

 # vi /etc/system

```
set shmsys:shminfo_shmmax=268435456
set semsys:seminfo_semmap=250
set semsys:seminfo_semmni=500
set semsys:seminfo_semmns=500
set semsys:seminfo_semmsl=500
set semsys:seminfo_semmnu=500
set semsys:seminfo_semume=100
set shmsys:shminfo_shmmni=100
set shmsys:shminfo_shmseg=10
```

Donde cada línea es un semáforo de la base de datos que indica lo siguiente[42]:

shmsys:shminfo_shmmax	es el tamaño máximo del segmento de memoria compartida.
semsys:seminfo_semmap	es el número de entradas en el mapa de semáforos.
semsys:seminfo_semmni	es el número de identificadores para los semáforos.
semsys:seminfo_semmns	es el número de semáforos en el sistema.
semsys:seminfo_semmsl	es el número máximo de semáforos por número de identificación.
semsys:seminfo_semmnu	es el número de estructuras que se pueden deshacer en el sistema.
semsys:seminfo_semume	es el número de entradas que se pueden deshacer por proceso.
shmsys:shminfo_shmmni	es el número de identificadores de memoria compartida.
shmsys:shminfo_shmseg	es el número máximo de segmentos de memoria compartida por proceso.

Sólo el super usuario root puede modificar este archivo.

Después de haber efectuado estos cambios es necesario reiniciar la máquina para que la modificación tenga efecto. Para verificar que el sistema ha actualizado estos parámetros se puede utilizar el siguiente comando:

 # sysdef

[42] http://www.sybase.com

Creación de las tablas del Sistema de Administración de Cuentas

Para la creación de las tablas de SAC, se elaboró un script que crea una a una las tablas que se van a utilizar, incluye la creación de índices, llaves primarias y llaves secundarias, además de la definición de catálogos en las tablas que los requieren. Este script se muestra a continuación:

```
#!/bin/sh
#
# Script de recuperacion de la base de datos sac
# Sistema de Administracion de Cuentas
#
# Para uso exclusivamente en caso de perdida total de los datos
# Desarrollado en el Departamento de Administracion de Servidores
# DGSCA - UNAM. 3 de noviembre de 1998.
#

clear
printf "\n        ! IMPORANTE !\n"
printf "\nEste programa realiza la creación de\n"
printf "la base de datos sac para el Sistema de\n"
printf "Administracion de Cuentas.\n\n"
printf "Para continuar con la recuperacion,\n"
printf "escribir el password de la cuenta sacadm\n"
printf "de lo contrario teclear Ctrl-C.\n\n"

if [ `ps -fea | grep dataserver | grep -v grep | wc -l` -eq 0 ]; then
  printf "El SQL Server no se encuentra en ejecucion...\n\n"
  exit 1
fi

if [ ! -x /usr/sybase/bin/isql ]; then
  printf "No fue posible encontrar ISQL...\n\n"
  exit 1
fi

printf "Password: "
stty -echo
read PASS
stty echo
printf "\n"
/usr/sybase/bin/isql -Usacadm -P${PASS} << EOF
print 'Creando tabla claves...'
create table claves(
clv_id     int     NOT NULL,
us_id      int     NOT NULL,
tip_id     tinyint NULL,
dir_id     int     NULL,
est_id     tinyint NULL,
ser_id     tinyint NOT NULL,
nombre     char(8) NOT NULL,
password   char(15) NOT NULL,
inicio     datetime NOT NULL,
modifica   datetime NOT NULL,
horas      int     NULL,
vence      datetime NULL,
forward    char(32) NULL,
descripcion char(48) NULL)
go

print 'Creando tabla dependencias...'
create table dependencias(
dep_id     int     NOT NULL,
tip_id     int     NOT NULL,
nombre     char(80) NOT NULL)
go

print 'Creando tabla directorios...'
create table directorios(
dir_id     int     NOT NULL,
directorio char(25) NOT NULL)
go

print 'Creando tabla estados...'
create table estados(
est_id     tinyint NOT NULL,
estado     char(20) NOT NULL)
go
```

```
print 'Creando tabla pagos...'
create table pagos(
pag_id     int      NOT NULL,
clv_id     int      NOT NULL,
fecha      datetime NOT NULL,
monto      money    NOT NULL,
recibo     char(32) NOT NULL,
tipo_pago  char(32) NOT NULL)
go

print 'Creando tabla servicios...'
create table servicios(
ser_id     tinyint  NOT NULL,
servicio   char(32) NOT NULL)
go

print 'Creando tabla tipo_clv...'
create table tipo_clv(
tip_id     tinyint  NOT NULL,
tipo       char(32) NOT NULL)
go

print 'Creando tabla tipo_dep...'
create table tipo_dep(
tip_id     tinyint  NOT NULL,
tipo       char(50) NOT NULL)
go

print 'Creando tabla usuarios...'
create table usuarios(
us_id      int      NOT NULL,
dep_id     int      NOT NULL,
nombre     char(16) NOT NULL,
paterno    char(16) NOT NULL,
materno    char(16) NOT NULL,
rfc        char(13) NOT NULL,
direccion1 char(32) NOT NULL,
direccion2 char(32) NULL,
cp         char(5)  NULL,
ciudad     char(32) NULL,
tel_num    char(25) NULL,
tel_desc   char(25) NULL,
titulo     char(12) NULL,
puesto     char(32) NULL,
e_mail     char(64) NULL)
go

print 'Creando indice claves_indx en claves...'
create unique clustered index claves_indx on claves (clv_id)
go

print 'Creando indice dependencias_indx en dependencias...'
create unique clustered index dependencias_indx on dependencias (dep_id)
go

print 'Creando indice directorios_indx1 en directorios...'
create unique clustered index directorios_indx1
on directorios (dir_id)
go

print 'Creando indice directorios_indx2 en directorios . '
create unique nonclustered index directorios_indx2
on directorios (directorio)
go

print 'Creando indice estados_indx1 en estados. .'
create unique clustered index estados_indx1 on estados (est_id)
go

print 'Creando indice estados_indx2 en estados  '
create unique nonclustered index estados_indx2 on estados (estado)
go

print 'Creando indice pagos_indx en pagos...'
create unique clustered index pagos_indx on pagos (pag_id)
go

print 'Creando indice servicios_indx1 en servicios...'
```

```
create unique clustered index servicios_indx1 on servicios (ser_id)
go

print 'Creando indice servicios_indx2 en servicios...'
create unique nonclustered index servicios_indx2 on servicios (servicio)
go

print 'Creando indice tipo_clv_indx1 en tipo_clv...'
create unique clustered index tipo_clv_indx1 on tipo_clv (tip_id)
go

print 'Creando indice tipo_clv_indx2 en tipo_clv...'
create unique nonclustered index tipo_clv_indx2 on tipo_clv (tipo)
go

print 'Creando indice tipo_dep_indx1 en tipo_dep...'
create unique clustered index tipo_dep_indx1 on tipo_dep (tip_id)
go

print 'Creando indice tipo_dep_indx2 en tipo_dep...'
create unique nonclustered index tipo_dep_indx2 on tipo_dep (tipo)
go

print 'Creando indice usuarios_indx en usuarios...'
create unique clustered index usuarios_indx on usuarios (us_id)
go

print 'Creando llaves primarias...'
execute sp_primarykey claves, clv_id
execute sp_primarykey dependencias, dep_id
execute sp_primarykey directorios, dir_id
execute sp_primarykey estados, est_id
execute sp_primarykey pagos, pag_id
execute sp_primarykey servicios, ser_id
execute sp_primarykey tipo_clv, tip_id
execute sp_primarykey tipo_dep, tip_id
execute sp_primarykey usuarios, us_id
go

print 'Creando llaves foraneas en claves...'
execute sp_foreignkey claves, directorios, dir_id
execute sp_foreignkey claves, estados, est_id
execute sp_foreignkey claves, servicios, ser_id
execute sp_foreignkey claves, tipo_clv, tip_id
execute sp_foreignkey claves, usuarios, us_id
go

print 'Creando llaves foraneas en dependencias...'
execute sp_foreignkey dependencias, tipo_dep, tip_id
go

print 'Creando llaves foraneas en pagos...'
execute sp_foreignkey pagos, claves, clv_id
go

print 'Creando llaves foraneas en usuarios '
execute sp_foreignkey usuarios, dependencias, dep_id
go

print 'Insertando valores en la tabla directorios '
insert directorios values(0,'/usr/users00')
go
insert directorios values(1,'/usr/users01')
go
insert directorios values(2,'/usr/users02')
go
insert directorios values(3,'/usr/users03')
go
insert directorios values(4,'/usr/users04')
go
insert directorios values(5,'/usr/users05')
go
insert directorios values(6,'/usr/users06')
go
insert directorios values(7,'/usr/users07')
go
insert directorios values(8,'/usr/users08')
go
insert directorios values(0,'/usr/users00')
```

```
go
insert directorios values(10,'/usr/users10')
go
insert directorios values(11,'/usr/users11')
go
insert directorios values(12,'/usr/users12')
go
insert directorios values(13,'/usr/users13')
go

print 'Insertando valores en la tabla estados...'
insert estados values(0, 'ACTIVA')
go
insert estados values(1, 'SUSPENDIDA')
go

print 'Insertando valores en la tabla servicios...'
insert servicios values(1,'CORREO ELECTRONICO')
go

print 'Insertando valores en la tabla tipo_clv...'
insert tipo_clv values(0,'CORTESIA')
go
insert tipo_clv values(1,'BECARIO O SERVICIO SOCIAL')
go
insert tipo_clv values(2,'ESTUDIANTE')
go
insert tipo_clv values(3,'INVESTIGADOR')
go
insert tipo_clv values(4,'ACADEMICO TIEMPO COMPLETO')
go
insert tipo_clv values(5,'ACADEMICO ASIGNATURA')
go
insert tipo_clv values(6,'FUNCIONARIO')
go
insert tipo_clv values(7,'PARTICULAR')
go
insert tipo_clv values(8,'NO LUCRATIVA')
go
insert tipo_clv values(9,'ADMINISTRATIVO')
go


print 'Insertando valores en la tabla tipo_dep...'
insert tipo_dep values(0,'INTERNA')
go
insert tipo_dep values(1,'EXTERNA')
go
insert tipo_dep values(2,'LUCRATIVA')
go
insert tipo_dep values(3,'NO LUCRATIVA')
go
insert tipo_dep values(4,'PARTICULAR')
go

quit
EOF
```

5.2 Instalación del servidor de WWW (Apache)

Apache requiere lo siguiente para su instalación[43]:

- 12 Mb de espacio en Disco Duro
- Compilador C de GNU (GCC) o cualquier compilador compatible con ANSI.
- Perl (opcional), para soporte de scripts en ese interprete.

1. Obtener la distribución actual del Servidor de WWW Apache para el sistema operativo que se este manejando.

2. Copiar esa distribución a un directorio[44]

[43] http://www.apache.org

% cp apache-1.3.9.tar.gz /dir

3. Descomprimir y expandir la distribución

% cd /dir
% gunzip apache-1.3.9.tar.gz
% tar xvf apache-1.3.9.tar

4. Iniciar proceso de instalación

% cd apache-1.3.9

5. Ejecutar el siguiente script para obtener parámetros de configuración del equipo

% ./configure --prefix=PREFIX
.....Configuring for Apache, Version 1.3.9
Creating Makefile
Creating Configuration.apaci in src
Creating Makefile in src
+ configured for Solaris 260 platform
+ setting C compiler to gcc
+ setting C pre-processor to gcc -E
+ checking for system header files
+ adding selected modules
+ doing sanity check on compiler and options
Creating Makefile in src/support
Creating Makefile in src/main
Creating Makefile in src/ap
Creating Makefile in src/regex
Creating Makefile in src/os/unix
Creating Makefile in src/modules/standard

6. Ejecutar el siguiente comando para compilar los programas de apache que van a ser instalados

% make
.....===> src
===> src/os/unix
gcc -c -I../../os/unix -I../../include -DSOLARIS2=260 `../../apaci` os.c
gcc -c -I../../os/unix -I../../include -DSOLARIS2=260 `../../apaci` os-inline.c
rm -f libos.a
ar cr libos.a os.o os-inline.o
ranlib libos.a
<=== src/os/unix
===> src/ap
gcc -c -I../os/unix -I../include -DSOLARIS2=260 `../apaci` ap_execve.c
gcc -c -I../os/unix -I../include -DSOLARIS2=260 `../apaci` ap_cpystrn.c
gcc -c -I../os/unix -I../include -DSOLARIS2=260 `../apaci` ap_signal.c
gcc -c -I../os/unix -I../include -DSOLARIS2=260 `../apaci` ap_slack.c
gcc -c -I../os/unix -I../include -DSOLARIS2=260 `../apaci` ap_snprintf.c
gcc -c -I../os/unix -I../include -DSOLARIS2=260 `../apaci` ap_fnmatch.c
rm -f libap.a
ar cr libap.a ap_execve.o ap_cpystrn.o ap_signal.o ap_slack.o ap_snprintf.

[44] http://www.apache.org

```
o ap_
fnmatch.o
ranlib libap.a
<=== src/ap
===> src/main
.....
.....
.....
```

7. Hacer la instalación como se muestra a continuación

```
# make install
.....===> [mktree: Creating Apache installation tree]
./src/helpers/mkdir.sh /usr/local/apache/bin
mkdir /usr/local/apache
mkdir /usr/local/apache/bin
./src/helpers/mkdir.sh /usr/local/apache/sbin
mkdir /usr/local/apache/sbin
./src/helpers/mkdir.sh /usr/local/apache/libexec
mkdir /usr/local/apache/libexec
./src/helpers/mkdir.sh /usr/local/apache/man/man1
mkdir /usr/local/apache/man
mkdir /usr/local/apache/man/man1
./src/helpers/mkdir.sh /usr/local/apache/man/man8
mkdir /usr/local/apache/man/man8
./src/helpers/mkdir.sh /usr/local/apache/etc
mkdir /usr/local/apache/etc
./src/helpers/mkdir.sh /usr/local/apache/share/htdocs
mkdir /usr/local/apache/share
mkdir /usr/local/apache/share/htdocs
./src/helpers/mkdir.sh /usr/local/apache/share/icons
mkdir /usr/local/apache/share/icons
.....
.....
```

 # PREFIX/bin/apachectl start

NOTA: PREFIX debe ser sustituido por el directorio o ruta en donde Apache va a ser instalado. Por ejemplo /usr/local/apache.

Para verificar que el servidor de WWW esta funcionando correctamente con Apache, hay que utilizar un web browser. La siguiente pantalla aparece cuando esto es satisfactorio:

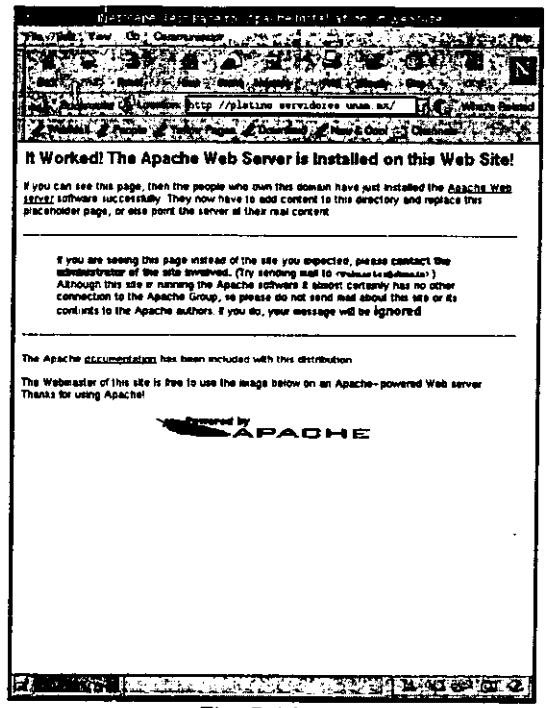


Fig. 5.14

Configuración del servidor de WWW con Apache

Editar el archivo httpd.conf que se encuentra en PREFIX/conf y hacer las siguientes modificaciones[45]:

- Definir el tipo de servidor para que este tenga una configuración **standalone** (toma todos los recursos del servidor)

ServerType standalone

- Especificar el puerto por el que el servidor va a recibir las peticiones.

Port 80

- Establecer los nombres de usuario y grupo que van a ser los encargados de ejecutar el proceso httpd.

User nobody
Group nogroup

- Determinar la dirección electrónica del Administrador del servidor.

ServerAdmin root@maquina.dominio

- Indicar el directorio de configuración del servidor.

ServerRoot /usr/local/apache

[45] http://www.apache.org

- Especificar el nombre del archivo donde se almacenarán todos los errores que pueda haber en el servidor.

ErrorLog /usr/local/apache/var/log/error_log

- Determinar el nombre del archivo donde se van a registrar todos los accesos al servidor.

CustomLog /usr/local/apache/var/log/access_log common

- Indicar el nombre del servidor.

ServerName maquina.dominio

- Establecer el nombre del directorio donde esta alojada la página principal del servidor.

DocumentRoot "/usr/local/apache/share/htdocs"

- Determinar los alias (pseudónimo) que se requieran, por ejemplo:

Alias /icons/ "/usr/local/apache/share/icons/"

ScriptAlias /cgi-bin/ "/usr/local/apache/share/cgi-bin/"

- Para usar cgi's es necesario habilitar lo siguiente:

AddHandler cgi-script .cgi

- Para utilizar imagenes mapeadas se requiere:

AddHandler imap-file map

- Definir la siguiente variable (Directory) con el nombre del directorio donde esta alojada la página principal del servidor.

<Directory "/usr/local/apache/share/htdocs">

- Especificar las opciones que se pueden manejar en las páginas, por ejemplo índices, ligas, etc.

Options Indexes FollowSymLinks

- Establecer las variables para la ejecución de cgi's.

<Directory "/usr/local/apache/share/cgi-bin">
AllowOverride None
Options FollowSymLinks ExecCGI
</Directory>

Configuración del Host Virtual

A continuación se muestran las directivas y su descripción que pueden ser utilizadas para la configuración de un Host Virtual[46]:

Directiva	Descripción
NameVirtualHost *Dir_IP_Servidor*	Utilizada para especificar la dirección IP del servidor
<VirtualHost *Dir_IP_Servidor*>	Utilizada para crear un Host Virtual. Se complementa con </VirtualHost>
ServerAdmin *Administrador*	Utilizada para definir el administrador del servidor de WWW. Generalmente se especifica la dirección de correo electrónico del mismo
DocumentRoot *Directorio*	Utilizada para especificar el directorio donde se aloja la página principal de este Host Virtual
ScriptAlias /cgi-bin/ *Directorio*	Utilizada para especificar el directorio cgi-bin de este Host Virtual
ServerName *Nombre_o_Dir_IP_Servidor*	Utilizada para definir el nombre del Host Virtual en caso de haberlo definido en un DNS o en su defecto se define la dirección IP del Host Virtual. En este último caso sólo se podría definir un Host Virtual en el servidor
ErrorLog *Archivo*	Utilizada para definir la ubicación y el nombre del archivo de errores del sitio de WWW de este Host Virtual
CustomLog *Archivo* common	Utilizada para definir la ubicación y el nombre del archivo de accesos al sitio de WWW de este Host Virtual
</VirtualHost>	Utilizada para indicar que se ha terminado la configuración de un Host Virtual.
<Directory *Directorio*>	Utilizada para definir características adicionales a un directorio. Se complementa con </Directory>
Options Indexes FollowSymLinks Includes ExecCGI	Utilizada para definir lo que se puede hacer en este directorio con las páginas HTML, como por ejemplo hacer índices, ligas, incluir archivos, ejecutar CGI´s.
AllowOverride *Archivo*	Utilizada para definir los archivos que deben ser ignorados en ese directorio. Generalmente se usa **None**.
</Directory>	Utilizada para indicar que se ha terminado la configuración de un directorio.
AuthType *Tipo*	Especifica el tipo de acceso que se tendrá en la página principal del servidor.
AuthName *Titulo_de_Identificación_de_Acceso*	Indica el nombre de Identificación de la página principal del servidor.
AuthUserFile *Archivo_de_Usuarios_de_Acceso*	Determina los usuarios y contraseñas que tienen permiso de acceder a la página principal del servidor.

Para configurar el Host Virtual es necesario editar el archivo PREFIX/conf/httpd.conf. A continuación se muestra la configuración utilizada en este proyecto:

[46] http://www.apache.org

```
NameVirtualHost 132.248.115.21

<VirtualHost 132.248.115.21>
   ServerAdmin sac@dominio
   DocumentRoot /home/sac/htdocs
   ServerName maquina.dominio
   ErrorLog logs/maquina.dominio-error_log
   CustomLog logs/maquina.dominio-access_log common
</VirtualHost>

<Directory /home/sac/htdocs>
order allow,deny
allow from 132.248.115. 132.248.111. 132.248.237.
Options Indexes FollowSymLinks ExecCGI
AuthType Basic
AuthName "Sistema de Administración de Cuentas"
AuthUserFile /usr/local/apache/conf/passwd.sac
<Limit GET POST>
require valid-user
</Limit>
</Directory>
```

Para crear el archivo que es referenciado por la Directiva AuthUserFile hay que ejecutar el siguiente comando:

```
# PREFIX/bin/htpasswd -c passwd.sac usuario
New password:
Re-type new password:
Creating password for user usuario
```

Después de haber editado el archivo de configuración de Apache y creado el archivo de autorización de usuarios, hay que reiniciar el servidor de Apache para que tenga efecto la nueva configuración:

```
# /usr/local/apache/bin/apachectl stop
/usr/local/apache/bin/apachectl stop: httpd stopped

# /usr/local/apache/bin/apachectl start
/usr/local/apache/bin/apachectl start: httpd started
```

5.3 Instalación del socket

Para instalar el socket es necesario hacer configuraciones tanto del lado del cliente como del lado del servidor. Para esto es necesario seguir lo siguiente:

En el servidor hay que crear la clave donde se va a alojar el programa.

 # useradd -u 1018 -g 10 -c "Sistema de Administración de Cuentas" -s /bin/csh -d /home/sac -m sac

Asignarle un password a la clave de usuario sac

 # passwd sac
New password:
Re-enter new password:
passwd (SYSTEM): passwd successfully changed for sac

Ejecutar el Script de Instalación llamado instala_servidor.

 # instala_servidor

Este Script realiza las tareas que se indican a continuación:

- Copia el archivo ejecutable **server** al directorio de la clave de sac que se creo anteriormente.
- Crea un directorio llamado **tmp** dentro del directorio de la clave sac, para crear archivos temporales que el servidor pueda generar.
- Crea un directorio llamado **conf** dentro del directorio de la clave sac, para los archivos de configuración. Dentro de conf, se crea un archivo vacío llamado **Ip_Allow**. En este archivo se debe especificar la(s) dirección(es) IP de la(s) máquina(s) que van a poder accesar vía telnet a este servidor por el puerto en el que va a recibir las peticiones. Esto sólo debe ser con fines de administración. Se recomienda que sólo se especifique la(s) dirección(es) IP de la(s) máquina(s) de los administradores.
- Crea un directorio llamado **logs** dentro del directorio de la clave sac, para almacenar un archivo llamado **server.log** donde se va a registrar cada acceso que se tenga al servidor. Este archivo presenta el formato:

  Fecha y hora de conexión.
  Usuario válido que se conectó. De donde se conectó. Que operación realizó. A que clave afectó. Resultado.
  Fecha y hora en que cerró la conexión.

Con lo anterior, sólo es necesario ejecutar el programa llamado server de la siguiente manera:

 # server &

Este programa empieza a atender las peticiones que son hechas por el cliente hacia puerto **9877** del servidor.

En el cliente hay que crear la clave donde se van a colocar todas las páginas HTML, CGI's y archivos de configuración.

 # useradd -u 1019 -g 10 -c "Sistema de Administración de Cuentas" -s /bin/csh -d /home/sac -m sac
6 blocks

Asignarle un password a la clave de usuario sac

 # passwd sac
New password:
Re-enter new password:
passwd (SYSTEM): passwd successfully changed for sac

Ejecutar el Script de Instalación llamado instala_cliente.

 # instala_cliente

Este Script realiza las tareas que se indican a continuación:

- Crear un directorio llamado **Claves** dentro del directorio htdocs de la clave sac, en el cual van a ser colocadas las páginas HTML para administar las claves del sistema.

- Crear un directorio llamado **usuarios** dentro del directorio htdocs de la clave sac, en el cual van a ser colocadas las páginas HTML para administar los usuarios del sistema.

- Crear un directorio llamado **pagos** dentro del directorio htdocs de la clave sac, en el cual van a ser colocadas las páginas HTML para llevar un registro de los pagos efectuados por los usuarios.

- Crear un directorio llamado **reportes** dentro del directorio htdocs de la clave sac, en el cual van a ser colocadas las páginas HTML para elaborar reportes de las claves del sistema.

- Crear un directorio llamado **CGI** dentro del directorio htdocs de la clave sac, para copiar todos los archivos de CGI's que van a ser utilizados por los usuarios para el manejo de las claves del servidor de correo electrónico.

- Crear un directorio llamado **images** dentro del directorio htdocs de la clave sac, para copiar todos los archivos de imagenes que se requieren en la interfaz gráfica.

- Crear un directorio llamado **conf** dentro del directorio htdocs de la clave sac, para los archivos de configuración. Dentro de conf, se crea un archivo llamado **Server_Host**, en este archivo se debe especificar la dirección IP del servidor en el cual se van a administrar las claves. Además se crea también un archivo llamado **Max_Users**, en el cual se especifica el número de usuarios que van a ser alojados en cada sistema de archivos de usuarios.

- Crea un directorio llamado **tmp** dentro del directorio htdocs de la clave sac, para generar archivos temporales que el sistema requiera.

5.4 Integración del Sistema

De esta manera quedan configurados todos los componentes del Sistema de Administración de Cuentas (SAC) de correo electrónico de RedUNAM.

Una vez que se terminó con la configuración de cada componente del sistema se muestra a continuación como se integran.

Para esto se requiere que la base de datos Sybase y la interfaz gráfica de usuario de la que se habló en el Capitulo 3, se encuentren instaladas en el mismo servidor, así como el servidor de Web Apache debe estar debidamente configurado en el mismo.

De esta manera podemos asegurar que todas las consultas del sistema que se hagan desde la interfaz gráfica de usuario van a estar de forma independiente al servidor que tiene alojadas todas las cuentas de correo electrónico, es decir, que no afectarán para nada el rendimiento del servidor de correo electrónico, ya que las consultas se harán desde el servidor de Web que hará peticiones a la base de datos que esta en el mismo servidor, por tanto no habrá carga adicional sobre el servidor de correo electrónico.

En el caso de las altas, bajas o cambios es necesario que el servidor de Web y base de datos interactúen con el servidor de correo electrónico, es aquí donde se necesita uno de los componentes más importantes, que precisamente logra establecer una comunicación entre estos servidores. Estamos hablando del socket, por el cual va a viajar toda la información del servidor de Web y base de datos al servidor de correo electrónico.

De esta forma el servidor de correo electrónico va tener una carga adicional normal de estas tres operaciones (altas, bajas y cambios), previo a esto, se hará una validación dentro de la base de datos para evitar cargas innecesarias sobre el servidor de correo y sobre la red por la cual se envían las peticiones y respuestas entre un servidor y otro.

Con la validación que se efectúa dentro de la base de datos antes de realizar una operación se gana tiempo y se ahorran recursos, que le dan respuesta más rápida al usuario que opera desde el Web, en el caso de que por ejemplo se quiera dar de alta una cuenta de correo electrónico que ya existe, primero se hace la revisión en la base de datos antes de enviar una petición al servidor de correo electrónico y si ya se encuentra un registro que se quiere agregar nuevamente, le arrojará al usuario un mensaje de que el registro ya existe, por lo que no se requiere que se valide el registro en el servidor de correo electrónico, de esta manera no habrá tráfico en la red ya que no se hará uso del socket y no habrá carga sobre el servidor de correo electrónico.

Cabe recalcar que el socket no funciona si en el servidor de correo electrónico no se esta ejecutando el programa **server** que es el servidor, visto desde una arquitectura cliente/servidor, que estará distribuyendo las solicitudes al servidor de correo electrónico. A su vez envía respuestas a cada petición, si esta es satisfactoria, envía un mensaje de exitoso, de lo contrario presenta un mensaje de error y motivo que lo origina, o bien el mensaje que regresa el sistema operativo.

Con la funcionalidad de este sistema quedan cubiertas las necesidades que se presentaron dentro del Departamento de Administración de Servidores de la D.G.S.C.A para la administración de las cuentas de correo electrónico de RedUNAM.

Ya con todos los elementos que necesita el sistema, en el presente capitulo se explicó de que forma quedan integrados, mencionando las herramientas y software utilizado para cumplir con las funciones requeridas.

Creemos que está solución puede tomarse como pauta para que en futuros proyectos se pueda utilizar este documento como base. Al momento de realizar este trabajo, estas eran las herramientas con las que contaba la institución. Seguramente habrá herramientas mas poderosas que faciliten el desarrollo de este tipo de aplicaciones.

CONCLUSIONES

Por un lado durante el trabajo de investigación tuvimos la oportunidad de conocer cuales fueron los inicios, en primera instancia de lo que es llamado Red de Redes o Internet, además de conocer como se fue formando y como ha ido evolucionando la infraestructura de Red de la UNAM hasta nuestros días, que afortunadamente va de la mano con el avance de la tecnología.

Además nos dio la oportunidad de conocer como se tienen implementados los servicios de Internet en nuestra institución y hacer una comparación de como se llevan en otras instituciones, lo que nos deja ver y nos da gusto que hasta la fecha nuestra institución sigue siendo una de las mejores.

Así también nos permitió aplicar los conocimientos teóricos que habíamos aprendido durante nuestra formación universitaria y tuvimos la oportunidad de integrarlos a una aplicación real y comercial como Sybase.

Por otro lado desarrollamos la habilidad para programar CGI's y a su vez integrarlos a una base de datos comercial de una manera muy sencilla y plasmarlos gráficamente de manera amigable a un usuario final. Muchas veces uno, como usuario final, no se imagina la complejidad que trae un proyecto por delante. Así también nos involucramos en el mundo de programación del lenguaje C bajo el sistema operativo UNIX, que nos ha dado mucha experiencia para desarrollar cosas que parecían muy difíciles, incluso nos ha llevado a conocer mucho de administración de sistema operativo UNIX, orientado hacia Solaris de Sun, que es en el ambiente en el que desarrollamos nuestro proyecto.

También se pudo llevar a la práctica el conocimiento que hemos adquirido durante nuestro tiempo en que hemos estado en distintas áreas de los sistemas y la computación y así aplicar nuevas técnicas que durante la carrera no fueron de nuestro conocimiento y así entender de una manera mas clara y eficiente el análisis y el desarrollo de un sistema. De la misma forma, se aprendió a dar una mejor explicación al proceso de la vida de un sistema, esto es, aprender a explicar los motivos y las razones por las cuales se necesitaban cada uno de los módulos del sistema y así entender nosotros mismos mejor el funcionamiento del mismo y las mejoras que se le pueden hacer para futuras versiones.

Fue un camino difícil, pero con las bases que teníamos, paso a paso, logramos construir y concluir un trabajo de tesis con los objetivos y alcances que nos habíamos planteado.

Para finalizar, el desarrollo de un proyecto como este, nos ha abierto caminos e inquietudes para desenvolvernos mejor en las actividades que actualmente realizamos, ya que nos ha enseñado que siempre hay algo más allá y que no lo ultimo que se ve o se encuentra es todo. Siempre habrá cosas mejores.

GLOSARIO

Alias: Sobre nombre para identificar un Host o un comando en UNIX.
ANSI: *American National Standars Institute.* Grupo que define estándares para muchas industrias en EUA, es el miembro del ISO de los EUA.
Apache Web Server : Servidor de WWW
Archie: Es un servicio de Internet que busca en los indices de los servidores FTP anónimo los nombres de archivos y directorios. Este servicio se proporciona comunmente a través de una sesión Telnet y correo electrónico ademas de los clientes de Archie.
ARPA : Primera etapa de Internet , producto de un proyecto del Gobierno de los Estados Unidos que data de 1970.
ARPAnet: Red nacida a principios de la decada de los setenta que enlazó en sus primeros años varias decenas de sitios en una red nacional dedicada a la comunidad de investigación en computación.
ASCII : *American Standard Code for Information Interchange* . Código americano estandard para intercambio de información. Código binario de datos que se usa en comunicaciones en la mayor parte de las mini computadores y computadoras personales.
Backup: Término empleado para respaldo de información comúnmente utilizado en la administración de base de datos.
BDI: Base de Datos Institucional
BITNET: *Because Its Time Network* . Red desarrollada en la Universidad de Nueva York . Permite a los usuarios intercambiar correo electrónico y archivos , pero no proporciona otros servicios de Internet.
***Browser* (Navegador) :** Para poder hacer uso de la WEB, se utiliza la tecnología del hipertexto para enlazar una gran cantidad de documentos que pueden incluir texto, imágenes, sonido, video y otras formatos multimedia. Para poder hacer uso y aprovechar estos formatos es neceario poder "navegar" a través de estas ligas, para esto se crearon los Browsers o Navegadores . Ej. Mosaic, Netscape Comunicator, Internet Explorer. A través de un navegador también se pueden utilizar otros servicios de Internet, tales como el FTP, Correo Electrónico o Buscadores de Información.
CGI: *Common Gateway Interface (Compuerta de Interfaz Común).* Mecanismo que utiliza el HTTP y que especifica como se entrega la información del usuario al servidor y desde él a otros programas externos. En otras palabras, es una especificación para permitir a un servidor de WWW ejecutar un programa externo y devolver los resultados al navegador
Cliente: Componente que se encarga de hacer peticiones a un servidor dentro de la arquitectura cliente-servidor .
Conmutación de Paquetes: La técnica de conmutación de paquetes consiste en organizar el flujo de datos entre un gran número de computadoras terminales, puestos de trabajo y otros elementos interconectados, de manera que sea posible compartir las rutas de transmisión; aprovechando el hecho de que, en la mayoría de sesiones de comunicaciones bidireccionales, los datos fluyen en realidad durante una parte relativamente pequeña de tiempo.
DARPA: Agencia de Proyectos de Investigación Avanzada para la Defensa
Datagrama: Es el formato de paquete definido por IP .
DBA: *Data Base Administrator* . Administrador de Base de Datos.
DBMS: *Data Base Management System.* Sistema Manejador de Base de Datos. Programa que nos va a facilitar la manipulación de una o varias bases de datos de manera transparente y simple
DCL: *Data Control Language :* Lenguaje de Control de Datos.
DDL: *Data Definition Language. Lenguaje de Definición de Datos*
DML: *Data Manipulation Language* . Lenguaje de Manipulación de Datos
DNS: *Domain Name Service* (Servidor de Nombre de Dominios) , DNS permitirá traducir los nombres de anfitrión que utilizan las personas a las direcciones IP numéricas que utlizan las máquinas, asi es como cada sitio tendrá información sobre sus propios anfitriones y podrá encontrar la información para otros sitios.
DTD: *Data Definition Type.* (Definición de Tipo de Documento). Estructura orientada a describir los elementos estructurales que aparecen en el hipertexto.

E-mail : Es uno de los servicios de redes más populares y básicos. Trabaja bajo el protocolo simple de transferencia de correo (SMTP).
ETD: Equipo Terminal de Datos.
Ethernet : Protocolo de red que en sus frames contiene la dirección destino, dirección fuente, tipo de campo y la información . Utiliza CSMA/CD (Carrier Sense and Multiple Access / Collision Detection)
FDDI : *Fiber Distribuited Data Interface* .Interfaz de datos distribuida por fibra . Es un tipo particular de red **LAN** que utiliza conexiones de fibra óptica.
Finger: Este servicio búsca información sobre un usuario que tiene una cuenta en la máquina que está conultando, sin importar el cliente inició una sesión o no una sesión en ese momento en esa máquina.
FTP: *File Transfer Protocol* (Protocólo de Transferencia de Archivos). Es el protocolo estándar de Internet para este propósito;proporciona el servicio para la transerferenica de archivos a, o desde una computadora remota.
Gcc :Compilador de Lenguaje C para UNIX distribuido por GNU
Gopher: Este servicio es un rastreador interactivo que se emplea por medio de menus jerárquicos que permiten buscar información en Internet a través de conexiones transparentes al usuario. Hoy en dia casi desaparecido e incorporado a servidores WWW.
Hipervínculo: Propiedad del lenguaje HTML que sirve para poder enlazar documentos dentro del WWW (textos, imágenes, videos, etc).
Host: Es la computadora en la cual residen los recursos.
HTML: *Hyper Text Markup Lenguage* (Lenguaje para marcar hipertexto), el cual es el lenguaje estándar para crear documentos Web y pueden contener : texto, imágenes,audio, video, etc.
HTTP: *HyperText Transfer Protocol* (Protocolo de Transferencia de HiperTexto) : Es el principal protocolo de aplicación que utiliza el World Wide Web, proporciona acceso de usuario a los archivos de usuario a los archivos que conforman el servicio WEB.
IMAP : *Internet Mail Agent Protocol.* Protocolo de Cliente servidor que sirve para manejar buzones electrónicos de usuarios.
Internet : Red mundial que enlaza a millones de máquinas y decenas de millones de usuarios en todo el mundo y que trabaja en base de varios protocolos de comunicación dentro de los cuales el principal y mas importante es el protocólo TCP/IP, mismo que se mantiene como estándar en la actualidad.
IP (Direcciones IP): Dirección numérica de cuatro bytes de longitud separadas por puntos (.) que identifica a una máquina conectada a Internet por lo tanto no pueden existir direcciones IP repetidas. Ejemplo de dirección IP 132.247.0.0.
IP : *Internet Protocol* (Protocolo de Internet): Es el protocolo de internet que da el servicio básico de envio de paquetes sobre el cual se construyen las redes TCP/IP. Toda la información TCP/IP fluye a través de IP, hacia dentro y hacia fuera sin importar su destino final. IP es un protocolo sin conexión, esto es, que no intercambia información de control para establecer una conexión extremo a extremo antes de transmitir los datos. IP confia en los protocolos de otros niveles para establecer la conexión si se necesita un servicio orientado a conexión.
ISO : *International Standard Organitation* . Organización Internacional de Normas. Emite Normas en una gama muy amplias de temas.
LAN: *Local Area Network.* Red de Area Local.
Majordomo: Aplicación para listas de correo electrónico.
Meta-lenguaje: Un lenguaje para definir lenguajes.Sistema de signos empleados para describir la estructura del propio lenguaje.
MS Exchange : Agente de Correo Electrónico de Microsoft.
Netscape Messaging Server : Servidor de Correo Electrónico de Netscape.
News: Llamados grupos de Noticias están diseñados para comunicación de muchos a muchos, es la contraparte de Internet a los tableros de foros de discusión .
NIC : *Network Information Center (*Centro de Información de la Red) : Es el Centro de Información que se encarga de distribuir la información de los servicios de red.
NOC: *Network Operation Center (*Centro de Operación de la Red): Centro de Operación de la red que se encarga de monitorear el comportamiento de la red así como la operación de ésta y darle el

mantenimiento necesario. El objetivo primordial del NOC es el mantener en óptimas condiciones la operación de la red y de los dispositivos conectados a ella.
NSF : *National Science Foundation (*Fundación Nacional de Ciencia).
ODBC: Object Data Base Control.
OSI: Open Systems Interconection. (Modelo de) Sistemas de Interconexión Abierta. Desarrollado por la *ISO*, se usa para describir la estructura y función de los protocolos de comunicación de datos.
Perl: Interprete utilizado para elaborar CGI's y scripts.
***POP** : Post Office Protocol.* Protocolo de Cliente servidor que sirve para manejar buzones electrónicos de usuarios.
PPP: *Point to Point Protocol* .- Protocolo utilizado para enviar tráfico TCP/IP a través de una línea de transmisión serial.
Procesos batch: Proceso por lotes ejecutado secuencialmente.
Protocolo: Establecen una descripción formal de los formatos que deberán presentar los mensajes para poder ser intercambiados por equipos de cómputo; además definen las reglas que ellos deben seguir para lograrlo.
RIT : Red Integral de Telecomunicaciones
Script: Conjunto de ordenes e instrucciones para realizar uno o varios procesos.
Sendmail : Servidor SMTP más común en UNIX.
Servidor: Componente que se encarga de dar servicios a las peticiones hechas por los clientes dentro de la arquitectura cliente-servidor.
SGML: *Standard Generalized Mark-Up Language(*Lenguaje Normal Generalizado de Marcado).
Shell: Intérprete de órdenes de UNIX. El cual indica que el usuario está presente desde el momento que se conecta y estará esperando sus ordenes o comandos, mostrando una marca o prompt.
SMTP: *Simple Mail Transfer Protocol (Protocolo Simple de Transferencia de Correo) :* Es el protocolo estándar de Internet para enviar y recibir correo electrónico.
Socket: Punto final de un enlace de comunicación de dos vías entre dos programas que se ejecutan a través de la red.
SQL: *Strucured Query Language. Lenguaje estructurado de consultas* .
SQL Server: Servidor de Base de Datos .
SSL: *Secure Socket Layer.* Herramienta para manejar transacciones y que hace a un servidor seguro, el cual se encarga de encriptar la información hecha por esta transacción.
Standalone: Configuración en la cual un servidor utiliza sus propios recursos .
Sybase : Manejador de Base de Datos.
TCP : *Transfer Control Protocol* . Protocolo de Control de Transferencia. El protocolo usado más comúnmente para los servicios de Internet. Proporciona una dirección bidireccional confiable entre dos extremos.
TCP/IP : Protocolo estándar de Internet para la comunicación entre todas las computadoras y servidores conectados a Internet.
Teleproceso: Técnica de intercambio de datos informáticos consistentes en una computadora central conectada a terminales alejadas.
Telnet: Es un servicio remoto permite que el usuario interactue con programas de aplicación que corran en una computadora ubicada en otro sitio.
Token Ring : Protocolo de red que trabaja por medio de tokens (mensajes) y en topologías tipo estrellas, utilizando Token Passing.
Topología: Forma de configurar a un red .
Triggers: Método de consulta o modificación de bases de datos que se ejecuta en el momento de un suceso definido.
TTVN : *Trans Texas Video Network*
UDP: *User Datagram Protocol* .Protocolo de Datagrama de Usuario. Proporciona un servicio de envio de datagramas con menos información de control que TCP, y trabaja sin conexión no confiable ya que no hay técnicas en el protocolo para verificar que la información llego correctamente al otro extremo de la red.
URL: *Uniform Resource Locator (Localizadores Uniformes de Recursos).* La forma general de los URL , es : servicio://host[:port]/path) . Hay varios valores posibles para el campo *servicio* : "ftp", "http",

"gopher" y "telnet" . Y en la parte de Host por lo general especifica el nombre del servidor o su dirección IP. El puerto (port) estádnar es el 80 el cual no es necesario ponerlo a menos que se trate de otro puerto que no sea el estándar, por ejemplo :8880 . Path se refiere a la ruta a la que se quiera acceder, generalmente se refiere a un documento o a un archivo que nos dará el servicio.
WWW: *World Wide* ***Web*** . Es la colección de servidores HTTP en Internet. El WEB es responsable, en su mayoría,de la reciente explosición de actividad dentro de Internet. (Ver *Browser*). El principal formato del WEB es el HTML (*Hyper Text Markup Lenguage).*(Ver HTML).

BIBLIOGRAFÍA

- Internet . Douglas E. Comer. Ed. Prentice Hall. México 1995.
- Diseño de Bases de Datos. Wiederhold Gio. Stanford University Ed. McGrawHill
- Introducción a las Bases de Datos . Gillenson Mark L. Mc Ed. Graw Hill . 1988
- Notas del Curso de SQL. Departamento de Innovación. . DCAA. UNAM. 1999
- Notas Sobre la Historia de Red UNAM. Departamento de Redes . DGSCA UNAM 1997
- Programming JavaScript for Netscape 2.0. Tim Ritchey / New Ed. Riders Publishing, Indianapolis, Indiana, USA 1996
- HTML and CGI Unleashed. John December and Mark Ginsburg. Sams.net Ed. Publishing. 1995.
- Black, Uyless; Redes de Computadoras; Madrid , España, Ed. Ra-Ma, 1989; pag. 1
- Stevens Richard W. , UNIX Network Programming. Networking APIs: Sockets and XTI. Volume 1 Second Edition. Ed. Prentice Hall PTR 1998.
- URL : http://www.nic.unam.mx/redunam/historia.html
- URL: http://ttvn.tamu.edu
- URL: http://webdia.cem.itesm.mx/ac/rtrejo/Interfaz/www.html
- URL: http://www.itlp.edu.mx/publica/tutoriales/basedat1/tema1_4.htm
- URL: http://www.usa.ethek.com/basedatos/intro.htm
- URL: http://www.kiyen.face.ubiobio.cl/~cvidal/base_datos...nal/tsld040.htm
- URL: http://www.face.ubiobio.cl/index.html
- URL: http://businessglobal.com/Formacion%20empresarial/Marketing%20en%20WWW/conc
eptos_basicos_www/internet_que_es.html
- URL : http://burn.ucsd.edu/~oli/html-2/htmldef.html
- URL: http://webdia.cem.itesm.mx/ac/rtrejo/Interfaz/browsers.html
- URL: http://emision.uson.mx/tips/Cgi/cgi.htm
- URL: http://www.cicei.ulpgc.es/gsi/curso_cgi2/cgi.htm
- URL: http://a01-unix.gsyc.inf.uc3m.es/~esther/lro9798/faq.html#11
- URL: http://kal-el.ugr.es/~victor/curso_javascript/js_intro.html

- URL: http://kal-el.ugr.es/NEW/www/index.html
- URL: http://www.mty.itesm.mx/dcic/centros/ciete/tutor_js/requisito.htm
- URL: http://www.ipn.mx/
- URL: http://www.colmex.mx/areas_apoyo/computo/computo.htm
- URL : http://www.uam.mx/
- URL . http://www.uvmnet.edu/ servicios/servicios.htm
- URL: http://www.sybase.com
- URL: http://www.apache.org
- URL : http://www.lpis.com/ayudas/tcp.html
- URL : http://www.geocities.com/SiliconValley/Lab/7583/articulos/asartic2.htm
- URL : http://www.comunnet.com.mx/Informacion/%2373419
- URL : http://www.geocities.com/SiliconValley/Lakes/2227/red/sockets.html
- URL : http://www.comunnet.com.mx/Informacion/%2373419
- FTP: ftp://ftp.nic.unam.mx/Politicas/DirIP-Politicas