12
24.

**UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO**

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
"CAMPUS ARAGON"

# "USO DEL HTML Y CGI PARA EL DESARROLLO DE APLICACIONES EN INTERNET"

## T E S I S

**QUE PARA OBTENER EL GRADO DE**
**INGENIERO EN COMPUTACION**
P R E S E N T A :


**CESAR CRUZ CORTES**


DIRECTOR DE TESIS:
ING. MARTIN ORDOÑEZ ROSALES

MÉXICO 1997.

A mis abuelos:

Agustín Cruz Morales†
Trinidad Angeles Angeles†
Rosa Cortés Martínez

Gracias por iniciar esta apasionante aventura. Los amo.

A mis padres:

Mateo Cruz Angeles
Fausta Sofía Cortés

Son un gran ejemplo de tenacidad y lucha por la vida. Gracias por aguantarme.

A mi hermano:

Ulises Cruz Cortés

Gracias por tus sabios consejos, eres mi segundo padre.

A la Universidad Nacional Autónoma de México,
mi alma mater.

A **México**:

# "Uso de HTML y CGI para el desarrollo de aplicaciones en Internet"

*César Cruz Cortés*

# Objetivos

- Identificar cuáles son los principios y elementos básicos que intervienen en el proceso de desarrollo de aplicaciones en Internet, a través del lenguaje HTML y el empleo adecuado de la tecnología CGI.

- Proporcionar mediante la información contenida en el presente trabajo de tesis una guía que sirva como base para el desarrollo de futuras aplicaciones empleando la tecnología HTML-CGI.

- Emplear las técnicas y procedimientos identificados durante el desarrollo de esta publicación para la implantación de una aplicación real.

# Índice

# Introducción

Desde hace algunos años hemos tenido que convivir más frecuentemente con los términos "Internet", "supercarretera de la información", "red global", etc.; los cuales intentan describir un conjunto inmenso de computadoras dispersas en todo el mundo conviviendo armónicamente para compartir recursos e información. Internet representa un nuevo medio de expresión que –aunque actualmente está restringido a un pequeño grupo de la población mundial– en unos cuantos años será el más importante medio de comunicación y de compartición de información de la humanidad.

Internet ha tenido gran aceptación a nivel mundial gracias a la serie de servicios que proporciona: e-mail, ftp, telnet, gopher, veronica, etc; sin embrago, de todos ellos existe uno que ha contribuido decisivamente en la penetración y aceptación de la "red global" en nuestras vidas: WWW ó World Wide Web. Este servicio no sólo ofrece las grandiosas facilidades de uso, comunicación y compartición de información presentes en los demás servicios, sino que además posee una interfaz gráfica que lo hace ameno, divertido y una gran experiencia para toda aquella persona que hace uso de él.

WWW representa un medio de comunicación y expresión de ideas ilimitado, pues no sólo permite la exposición de texto (como TODOS los demás servicios de la "red global"), sino que además posibilita la inclusión de gráficos, audio, video, etc.; ocasionando con esto que los documentos que presenta al usuario tengan mucho más impacto en éste que sus equivalentes de algún otro servicio. Los documentos existentes en WWW (comúnmente llamados páginas) se encuentran desarrollados en lenguaje HTML (Hipertext Markup Language); su uso implica el empleo de metodologías bien definidas que facilitan en gran medida el desarrollo de aplicaciones para WWW y, en consecuencia, para Internet.

La gran demanda en el uso de WWW provocó que sus usuarios –no conformes con las facilidades que recibían– tuvieran la necesidad de incluir en sus documentos HTML elementos que no pertenecieran a WWW: bases de datos; interacción con otros servidores; creación de páginas interactivas que presentaran diferencias en base a las selecciones del usuario; etc. Para lograrlo, fue desarrollada una tecnología denominada CGI (Common Gateway Interface), la que –a grosso modo– se compone por un programa (compuerta) que representa una interfase entre el documento HTML y la aplicación ajena a WWW.

Actualmente, el uso combinado de HTML y CGI ha permitido el desarrollo de aplicaciones diversas, todas ellas aprovechando las grandes facilidades que en materia de comunicación visual proporciona WWW; su proceso de desarrollo no es sencillo, pues exige al desarrollador un conocimiento mediano de HTML e Internet, conocimiento

de un lenguaje de programación y dominio de la metodología empleada para el desarrollo de la aplicación ajena a WWW.

Al momento de escribir estas líneas (enero de 1997), existían sólo unas cuantas publicaciones referentes a CGI, pero ninguna de ellas exponía en su totalidad los pasos a seguir para desarrollar una aplicación completa que integrara a WWW un documento HTML y una aplicación externa. Esta falta de información fue el detonante que me impulsó a investigar las bases teóricas de Internet, HTML y CGI, plasmándolas en el presente escrito, cuya intención primordial es proporcionar a la comunidad de ENEP Aragón (y de México en general) una guía que sirva como fundamento teórico y práctico para el correcto desarrollo de aplicaciones en Internet (concretamente en WWW), empleando la tecnología que tanta popularidad le ha dado a la "red global"y que está tan de moda en nuestros días: HTML y CGI.

La presente publicación está dividida en cinco secciones:

En el **Capítulo 1 (Internet)** expongo la historia y características de la "red global": composición, fundamentos, estructura, servicios; además, conoceremos a detalle las características de WWW.

El **Capítulo 2 (Metodología para el desarrollo de aplicaciones)** nos presenta una técnica a seguir para el desarrollo en WWW; ésta presenta grandes semejanzas con las metodologías convencionales, sin embargo posee características interesantes que hay que considerar para el correcto desarrollo de aplicaciones útiles y bien estructuradas en WWW.

En el **Capítulo 3 (Implementación: HTML)** conoceremos a detalle el lenguaje en que se basa WWW; entenderemos la estructura de los documentos HTML, así como gran parte de los elementos que lo conforman.

El **Capítulo 4 (Programación de CGI)** nos brinda la oportunidad de abrir la caja negra que hasta este momento hemos conocido como CGI, explorando la estructura de los programas que le dan vida a esta tecnología que ha revolucionado por completo la forma en que son producidos los documentos HTML que conforman a WWW.

Finalmente, el **Capítulo 5 (Caso de estudio)** representa la fusión de todos los conceptos teóricos expresados en las primeras cuatro secciones de esta obra, materializándolas en la planeación, análisis, diseño, e implantación de una aplicación que hace ***"uso de HTML y CGI para el desarrollo de aplicaciones en Internet"***.

Espero que esta publicación sea útil a quienes la consulten, pues su estructura y contenido le permite ser tomada en cuenta como una guía para el desarrollo de más y mejores aplicaciones que la que aquí expongo. Sin más preámbulo, comencemos pues el estudio de esta tecnología que ha –y seguirá– revolucionado los métodos de intercambio y compartición de información de la humanidad.

## Convenciones Notacionales.

Con la intención de facilitar la lectura de esta publicación, han sido adoptadas ciertas convenciones en el texto, las cuales son:

El texto con `tipo de letra Courier`, indica al lector una palabra incluida en el glosario de términos al final de la obra.

El texto con tipo de letra Rino, indica al lector una sección de código de cualquier tipo, es decir el contenido de un programa.

El texto con tipo de letra Times New Roman, señala un "comando", el cual deberá ser escrito tal cual aparece en la obra, para poder ser ejecutado.

Las palabras en *itálicas*, destacan términos que pudieran ser desconocidos y NO son incluidos en el glosario de términos, sino expuestos en una sección completa de la presente publicación.

En caso de adoptarse otras disposiciones para la notación del texto, ésto será indicado en su momento; sin embargo, las reglas anteriores generalizan en gran medida el contenido de esta obra.

# 1

# Internet

## I.1 Historia.

### Internet Ayer.

La aparición de Internet fue motivada –al igual que la mayoría de los grandes avances tecnológicos contemporáneos– no por un espíritu puramente científico, sino por intereses militares.

A finales de la década de los 60's, el Departamento de Defensa de los Estados Unidos decidió llevar a cabo una investigación para crear una red de computadoras (geográficamente distantes) tolerante a fallas, que pudiera mantener el control y flujo de información en caso de presentarse una guerra nuclear. Se designó para ello a **ARPA (Advanced Research Projects Agency**; Agencia de Proyectos de Investigación Avanzada), dependiente del Departamento de Defensa de los Estados Unidos; esta agencia realizó su trabajo apoyándose en las investigaciones de el **Laboratorio Nacional de Física del Reino Unido** y la **Sociedad Internacional de Telecomunicaciones Aeronáuticas de Francia**; ambas instituciones estudiaban un método para comunicar computadoras entre sí, mediante *conmutación de paquetes*; este método proveía un proceso de gran flexibilidad y rentabilidad al "mover datos" eficientemente entre computadoras remotas.

ARPA financió una investigación a través de la firma **Bolt Beranek and Newman (BBN)**, y en 1969, ésta presentó un protocolo de red basado en *conmutación de paquetes* llamado **Network Control Protocol (NCP)**, así como una computadora central llamada **Information Message Processor** (**IMP**; Procesador de Mensajes de Información). El primer IMP fue instalado en la Universidad de California en Los Ángeles (UCLA) en 1969; para 1970 existían otros tres IMPs: uno en la Universidad de California en Santa Bárbara, otro en la Universidad de Utah en Salt Lake y el tercero en la Universidad de Stanford, todos ellos conformando la **primera red de computadoras basada en un protocolo de conmutación de paquetes: ARPAnet**; si alguna de las conexiones de esta red se interrumpía, los mensajes podían ser transmitidos aún a través de las conexiones restantes, proveyendo así la tolerancia a fallas tan buscada por ARPA. Este hecho significó el nacimiento de Internet

Para 1972 había ya 40 computadoras (IMPs) conectadas a ARPAnet, las cuales podían entre otras cosas: intercambiar información a través de *correos electrónicos ó e-mails;* accesar IMPs geográficamente distantes; transferirse grandes archivos de texto conteniendo información mediante *FTP* (*file transfer protocol*; protocolo de transferencia de archivos), etc. La tecnología base para la evolución de ARPAnet a lo que hoy conocemos como Internet, estaba ya establecida.

A pesar de las grandes ventajas de interoperabilidad entre equipos distantes que ofrecía ARPAnet, existía una seria desventaja en ella, pues sólo podían participar en su funcionamiento los equipos IMP diseñados por BBN; era necesario desarrollar una tecnología capaz de ser usada en cualquier equipo de cómputo del mundo para comunicarse con otros equipos en redes de computadoras distantes. Para 1974 se desarrolló, bajo la dirección de **Vinton Cerf** y **Robert Kahn** (financiados por el gobierno de los E.U. a través de su recién bautizada Agencia de Proyectos de Investigación Avanzada del Departamento de Defensa; **Defense Advanced Research Project Agency** ó **DARPA**), varios protocolos de red que permitían la comunicación confiable entre redes de computadoras distintas, basadas en conmutación de paquetes: *TCP/IP*, que serían a la postre los **dos protocolos principales de Internet.**

Los trabajos desarrollados por Kahn y Cerf permitieron comunicar en 1974 a ARPAnet con dos nodos de **EUROnet** (una red de computadoras europea basada en los viejos IMPs de BBN): uno en el **Colegio Universitario de Londres** y el segundo en las **Reales Instalaciones de Radar de Noruega**. Gracias a esta conexión, ARPAnet proveía grandes facilidades de comunicación y compartición de información entre los investigadores con acceso a ella en E.U. y Europa, aunque su uso era exclusivo del personal del gobierno de cada nación.

En 1977, la Universidad de Wisconsin (E.U.A.) creó una red de uso exclusivo para investigadores: **Theorynet**, debido a la "marginación tecnológica" en la que se encontraban los institutos y universidades que no estaban conectados a ARPAnet. Theorynet fue el resultado de la necesidad de otra red como ARPAnet, pero destinada enteramente a investigación científica no gubernamental. En 1979, se sostuvo una reunión entre varias universidades, DARPA y la Fundación Nacional de Ciencia de los Estados Unidos (**NSF**; **National Science Foundation**), donde se acordó la creación de **CSnet** (**Computer Science Research Network**), una red apoyada en gran parte por NSF, de uso menos restringido y que proveía a sus usuarios la mayoría de las facilidades de ARPAnet.

Para 1980, CSnet estaba conformada por una colección de redes de computadoras independientes, unidas gracias a una *compuerta* ó *gateway*; partiendo de dicha configuración, Vinton Cerf propuso la unión de CSnet y ARPAnet empleando la misma técnica: una compuerta, y el protocolo que él y Robert Kahn crearan años atrás (*TCP/IP*). La idea de Cerf se materializó y permitió que las dos redes de computadoras más grandes hasta ese momento se unieran y trabajaran armónicamente. Este hecho marca el nacimiento de la implementación física actual de Internet (figura I.1).

En 1986, el gobierno norteamericano crea **NSFnet (National Science Foundation Network)**, que era una red que ligaba varias supercomputadoras a lo largo de la unión americana. El equipo de supercómputo mencionado, servía a manera de

nodos principales de dicha red, de tal manera que cada uno de ellos constituía un punto central para otras redes. El propósito básico de NSFnet era proveer servicios de cómputo de la más alta calidad a investigadores y alumnos de universidades en los Estados Unidos. NSFnet tenía contacto con la entonces Internet (red ARPAnet-CSnet), enlazada con ésta empleando una compuerta (gateway) y el protocolo TCP/IP.

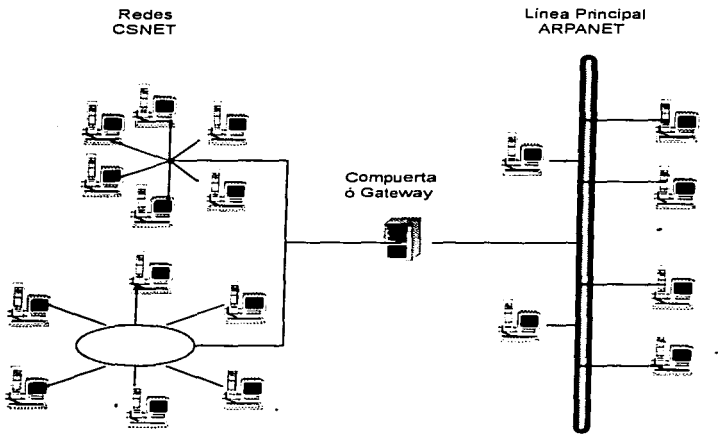



**FIGURA I.1**. La red CSnet se unía con ARPAnet mediante una compuerta ó gateway.

El gran apoyo brindado a NSFnet por parte del gobierno de los E.U. provocó que esta superara en importancia a la red ARPAnet-CSnet, la cual, a pesar de su apertura de 1979, no dejaba de ser elitista, empleada por un grupo reducido de personas. Es debido al incremento de popularidad de NSFnet, que se pierde el momento en que dicha red dejó de ser llamada como tal y comenzó a ser llamada INTERNET.

Durante la segunda mitad de la década de los 80's, apareció una tendencia de diversas organizaciones, institutos y universidades de crear sus propias redes y buscar su conexión a NSFnet, originando una verdadera explosión en el número de **hosts** ó **máquinas centrales** conectadas a la red (cada una de ellas capaces de proveer conexión a otras máquinas, multiplicando así el número de computadoras con acceso a

la red): para 1987, eran de más de 10,000 los nodos conectados, y para 1989 superaban ya los 100,000.

Entre 1990 y 1991, desaparecen ARPAnet (que había perdido todo vínculo con el ejército norteamericano desde 1983) y Csnet (a pesar de haberse fusionado con algunas otras redes); las funciones de éstas fueron absorbidas y posteriormente superadas por **NSFnet**. La red de NSF resultó ser la red de computadoras dominante en E.U. y el mundo. Una vez desaparecidos sus más fuertes rivales (desde el punto de vista de utilidad y uso) no hubo quien pudiera evitar que NSFnet fuera el eje físico en torno al cual giraría TODO el funcionamiento de Internet, es decir su columna vertebral.

Durante los primeros años de la década de los 90's infinidad de países y organizaciones de todo el mundo consiguieron su conexión a Internet (México lo hizo en 1990), como que el número de hosts llegó a un millón para 1992. Ese mismo año, en Suiza fue liberado *WWW* (*World Wide Web*), lo que permitió en gran medida la aceptación e incremento de usuarios y máquinas en Internet (Véase WWW en las secciones posteriores de este capítulo). El uso de Internet ya no se limitaba exclusivamente a investigadores o universidades, sino que se amplió a toda clase de personas; el tipo de información que viajaba por la red se diversificó enormemente.

Internet crece día a día en la conformación internacional que justifica en parte su nombre: **International Network** (red internacional). Véase tabla I.1

| Fecha | Hosts | Redes | Fecha | Hosts | Redes |
|---|---|---|---|---|---|
| 1969 | 4 | ? | Diciembre,1987 | 28,174 | ? |
| Abril,1971 | 23 | ? | Octubre,1988 | 56,000 | ? |
| Junio,1974 | 62 | ? | Octubre,1989 | 159,000 | 837 |
| Marzo,1977 | 111 | ? | Octubre,1990 | 313,000 | 2,063 |
| Agosto,1981 | 213 | ? | Octubre,1991 | 617,000 | 3,556 |
| Mayo,1982 | 235 | ? | Octubre,1992 | 1,136,000 | 7,505 |
| Agosto,1983 | 562 | ? | Octubre,1993 | 2,056,000 | 16,533 |
| Octubre,1984 | 1,024 | ? | Octubre,1994 | 3,864,000 | 37,022 |
| Octubre,1985 | 1,961 | ? | Julio,1995 | 6,642,000 | 61,538 |
| Noviembre,1986 | 5,089 | ? | Julio,1996 | 12,881,000 | 134,365 |

**Tabla I.1** Crecimiento de Internet.

## Internet Hoy.

Internet se compone actualmente por más de 200,000 redes de computadoras repartidas por todo el mundo. **Para julio de 1996, existían ya 12,881,000**[1] computadoras centrales ó hosts conectadas a la red global; cada una de ellas, sirviendo de punto central para la conexión de otras computadoras, resultando en millones y millones de nodos con acceso a la red de redes.

Ahora bien, habiendo conocido ya el nacimiento y evolución de la red global hasta su estado actual, cabe la pregunta ¿Quién gobierna o regula las acciones de Internet?

[1] Pueden obtenerse datos actualizados en el ftp público de **nic.merit.edu** en el directorio /nsfnet/statistics.

La respuesta es la Sociedad Internet (**ISOC**; **Internet Society**). Internet no tiene presidente, director ejecutivo o mandatario; ISOC es una sociedad de participación voluntaria, cuyo propósito es promover el **intercambio de información a nivel global** mediante el uso de la tecnología Internet: "La Sociedad Internet provee asistencia y soporte a grupos y organizaciones envueltos en el uso, operación y evolución de Internet. Apoya foros en los que pueden ser discutidas preguntas técnicas y operacionales, y provee mecanismos a través de los cuales las partes interesadas pueden ser informadas y educadas acerca de Internet, su función, uso y operación".

La Sociedad Internet actúa como un asesor, y sus miembros[2] pertenecen a varios grupos o foros dentro de la sociedad. Uno de esos grupos es el consejo de Arquitectura Internet (**IAB**; **Internet Architecture Board**), formado en 1983 para impulsar la investigación dentro de la red global. Actualmente, el IAB se concentra en los miembros del ISOC que producen estándares para interconectar redes y asignan recursos, creando grupos de trabajo como el de Ingeniería de Internet (**IETF**; **Internet Enginnering Task Force**); que es también un grupo de voluntarios cuya función es la de discutir problemas operacionales y desarrollos tecnológicos que pudieran (o debieran) tener impacto en Internet[3].

Internet ha sido por muchos años una red internacional, pero sólo se había extendido hacia países con buenas relaciones diplomáticas con los Estados Unidos. En la actualidad, con una situación política internacional menos tensa, Internet se ha difundido por todos lados, se han integrado incluso los países del antiguo bloque comunista, los cuales se encontraban aislados debido a las regulaciones del gobierno norteamericano. En los países del tercer mundo como México, vemos a Internet como un medio eficiente para elevar nuestros niveles educativos y tecnológicos.

Así pues, la red de redes se ha convertido en un rico banco de información al alcance de todo aquel que pueda "conectarse" a la misma, se calcula que comunica a cerca de 40 millones de personas repartidas por todo el mundo y la cifra sigue en aumento. Los *servicios* disponibles, proporcionados por Internet son expuestos más adelante en este capítulo.

## I.2 Funcionamiento.

### Antecedentes.

Para comprender el funcionamiento de Internet, es necesario entender el protocolo en el que la red de redes se basa: TCP/IP; su estudio implica una serie de conceptos que expongo a continuación.

---

[2] Para unirse a la Sociedad Internet, es posible solicitar información mediante correo electrónico a **isoc@isoc.org**. Las cuotas oscilan entre 70 y 25 dólares por año.

[3] Para poderse enterar de los avances o novedades en Internet, es posible consultarlas en el ftp público **nic.merit.edu** en el directorio **/newsletter**, o bien, mediante *e-mail* a **newsletter-request@is.internic.net.**

**Conmutación de datos.**

La **conmutación** es una importante técnica que determina cómo se realizarán las conexiones y cómo se manipulará el movimiento de los datos entre dos redes intercomunicadas. Para esto, existen tres métodos principales: **conmutación de circuitos, conmutación de mensajes** y **conmutación de paquetes.** Ver figura I.2

En la ***conmutación de circuitos*** se conecta al emisor y receptor mediante un único camino físico (una conexión física dedicada), mantenido durante la totalidad de la conversación entre nodos. Este método provee una velocidad garantizada en la transmisión; sin embargo, es muy caro, pues el canal entero es desperdiciado en el ancho de banda que no es utilizado al momento de la comunicación.

En la ***conmutación de mensajes***, no se establece una conexión dedicada entre dos estaciones; cada mensaje es tratado como una unidad independiente e incluye su propia *dirección* de origen y destino. Cada **mensaje completo** es transmitido y almacenado de dispositivo a dispositivo a lo largo de la red, por lo regular, a través de diferentes caminos. Provee un manejo eficiente del tráfico en la red; sin embargo, existe un retraso significativo en el almacenaje-envío de los mensajes, y resulta costoso el dotar a cada equipo intermedio con la capacidad suficiente para almacenar mensajes de gran tamaño.

La ***conmutación de paquetes*** reúne lo mejor de los dos métodos anteriores. Internet debe su éxito al empleo de un protocolo de conmutación de paquetes. Aquí, los mensajes son divididos en piezas más pequeñas llamadas **paquetes** (de bits); cada uno de los cuales incluye una cabecera con su origen, destino e información de la dirección de nodos intermedios. Los paquetes individuales no siempre siguen la misma ruta; esto es llamado *ruteo independiente*. La diferencia entre éste método y el de conmutación de mensajes, radica en que la conmutación de paquetes restringe a los mismos a una longitud máxima, la cual es lo suficientemente corta como para ser alojada en la memoria de los conmutadores, sin que éstos tengan que hacer operaciones de escritura en algún dispositivo de almacenamiento secundario, ocasionando que la operación sea más rápida. Existen dos métodos de conmutación de paquetes:

- **Conmutación de paquetes** datagrama. Este método trata a cada paquete como una unidad independiente, como si cada uno de ellos fuera un mensaje completo en lugar de ser parte de algo más grande. Los conmutadores "*rutean*" cada paquete independientemente a través de las redes que están interactuando; cada nodo intermedio determina el siguiente segmento de red en el camino del paquete. Los conmutadores pueden evitar segmentos de red con un gran tráfico, asegurando la entrega de los paquetes sin retardos innecesarios. Como los paquetes datagrama siguen diferentes rutas en su camino de una red a la otra: es posible que paquetes de un mismo mensaje lleguen en desorden, por lo que cada paquete incluye en su cabecera (bits iniciales) un número secuencial, que es empleado por el dispositivo receptor para reordenar los paquetes y reconstruir el mensaje original.

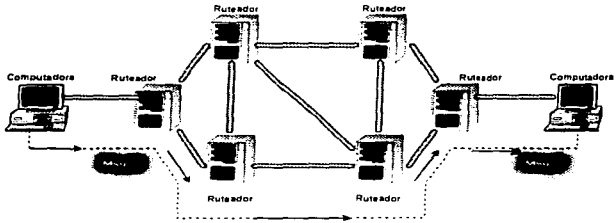



**Figura I.2a.** Conmutación de circuitos

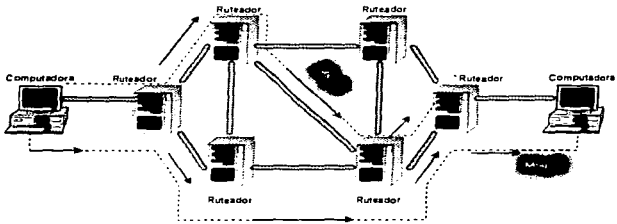



**Figura I.2b.** Conmutación de mensajes

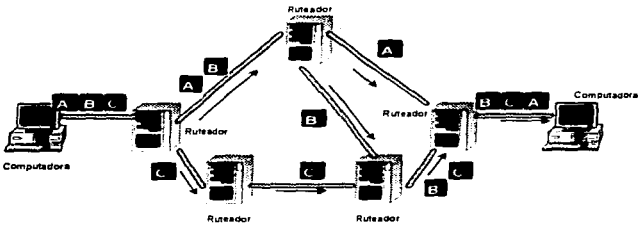



**Figura 1.2c.** Conmutación de paquetes

- **Conmutación de paquetes por circuitos virtuales.** Esta tecnología establece una conexión lógica entre los dispositivos origen y destino, llamada **circuito virtual**, pues no existe ningún circuito físico dedicado entre las máquinas, sino que es determinado al momento de requerirse la comunicación. Una vez que el circuito virtual es establecido, los dos dispositivos lo emplean por el resto de la conversación, o bien, mientras los dos dispositivos sean operacionales (no los apaguen o interrumpan su función). La diferencia principal entre éste método y el de *conmutación de paquetes datagrama* es que aquí todos los **paquetes** viajan a través de la conexión lógica establecida entre los dispositivos receptor y emisor; mientras que en el segundo, los paquetes podrían viajar a través de rutas diferentes.

**Direccionamiento de redes.**

Hemos visto que en los diferentes métodos de conmutación se habla continuamente de *paquetes de datos*, es importante saber que además de los datos de dichos paquetes, existen en ellos una serie de especificaciones técnicas entre las que se encuentran la designación de las máquinas origen y destino que están interactuando; ésta designación consiste en dos direcciones: una **dirección física** y una **dirección lógica**.

La **dirección física** (comúnmente llamada de hardware ó **dirección MAC**[3]), es empleada para transmitir paquetes desde y hacia una estación de trabajo especifica **dentro de una red local**. Cuando los mensajes a transmitir cruzan los límites de la red, y viajan por una red mayor (compuesta de varias redes `interoperando`), es necesario identificar la red lógica así como el dispositivo físico. La **dirección lógica** es necesaria para identificar cada red individualmente.

Mediante el empleo de la **dirección lógica**, un *ruteador* direcciona el paquete al segmento de red correcto, una vez que el paquete ha llegado a la red deseada, el ruteador emplea la **dirección física** para identificar un dispositivo específico dentro de dicha red. Sin direccionamiento lógico, el ruteo no sería posible.

La dirección lógica es asignada por el administrador de la red al momento de la configuración de la misma, en cambio, la dirección física es asignada por el fabricante de la tarjeta de red correspondiente.

**Ruteo.**

**Rutear** es el proceso de "encaminar" o "guiar" mensajes –generalmente a través de caminos diferentes– entre redes distintas, permitiéndoles a éstas interoperar. Este concepto implica la determinación de los caminos posibles entre las redes que interactúan (*descubrimiento de rutas*), así como la elección de los caminos disponibles (*selección de rutas*). Los dispositivos que realizan esta tarea son denominados **ruteadores**.

[3] **Media Access Control** ó Control de Acceso al Medio físico de la red

El **descubrimiento de rutas** es el proceso de encontrar los caminos disponibles entre las redes que interoperan; para ésto se construyen **tablas de ruteo,** donde se almacena dicha información. Como las condiciones de la red cambian con el tiempo, los ruteadores necesitan ejecutar el descubrimiento de rutas regularmente (típicamente una vez por minuto) asegurando con ésto que sus tablas de ruteo sean exactas y se encuentren actualizadas. Además de guardar información de los diferentes "caminos" de la red, las tablas de ruteo almacenan también **costos estimados** del envío de un mensaje a través de una ruta dada. El costo de una ruta en particular puede ser definido en base a varios aspectos: **tiempo** y/o **distancia estimados**, o **estimaciones monetarias** (como si el uso de una ruta "costara" más que el de otra).

Para la **selección de rutas**, un ruteador hace uso de la información de costos contenida en sus tablas de ruteo para elegir el mejor camino entre las redes interoperantes; dicho camino se determina en base a el menor número de ruteadores por el que un mensaje debe pasar para llegar a su destino (**menor número de hops**); la cantidad de tiempo requerida para que un mensaje llegue a su destino (**número de ticks; 1 tick = 1/18 segundo**); o costo relativo.

## TCP/IP.

### Modelo de redes DOD.

El protocolo TCP/IP se basa en un modelo conceptual denominado *modelo de redes DOD*[5]. El modelo DOD direcciona todos los procesos que son requeridos para una comunicación eficiente entre computadoras; estos procesos están divididos en grupos lógicos o capas. El uso de un modelo estratificado o dividido en capas presenta varias ventajas:

- Promueve la especialización. Los desarrolladores pueden enfocarse en la función de sólo una capa, teniendo la certeza de que las otras funciones necesarias para la comunicación serán atendidas por algun otro estrato.
- Modularidad. En caso de ser necesario un cambio en alguna capa, no significa que ese cambio se necesite en algún otro estrato.
- Compatibilidad. Si los desarrolladores de software se apegan a las especificaciones del modelo, todos los protocolos que conformen ese modelo referencial, en cualquier capa, trabajarán juntos.
- Un modelo en capas permite a los servicios de red ser definidos en base a funciones, no de sus implementaciones particulares.

[5] Llamado así por las siglas del Departamento de Defensa de los Estados Unidos ó Departament Of Defense, que patrocinó las investigaciones que le dieron origen

En un modelo de capas, los datos de una máquina origen (emisora) comienzan en la capa más alta y deberán seguir un camino descendente entre capas hasta la más baja, por su parte, la máquina destino (receptora) recibirá éstos datos en su capa más baja y recorrerá un camino ascendente entre las diferentes capas hasta la más alta. Cada capa destino ve y actúa únicamente sobre los datos que fueron "procesados" por su contraparte en el lado emisor. Véase figura I.3.

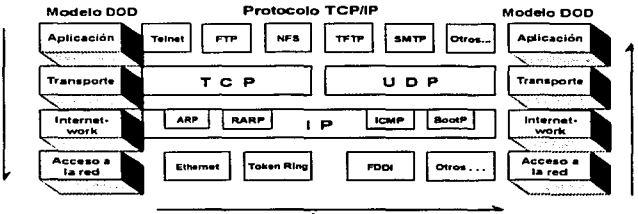



**Figura I.3.** Modelo de capas DOD.

A partir del modelo DOD, se deriva un conjunto de protocolos denominado TCP/IP; a continuacion expongo los protocolos implementados en TCP/IP, recorriendo cada una de las cuatro capas del modelo DOD, de la más superior (que es de donde se originan los datos) a la inferior:

**Capa de Aplicación.**

Uno de los objetivos en el diseño de los creadores originales de Internet, fue tener aplicaciones que pudieran correr en diferentes plataformas de cómputo, y aún así poder comunicarse entre sí. Casi todas las **aplicaciones** escritas con estos protocolos TCP/IP pueden ser catalogadas como **aplicaciones cliente-servidor**; estas aplicaciones distinguen tres entidades diferentes:

- **El servidor.** El software del servidor[6] se ejecuta en la máquina donde los datos residen (se encuentran físicamente); es aquí donde se realiza gran parte del procesamiento de datos así como su almacenamiento. El software del servidor es comúnmente llamado "proceso demonio" (daemon), que se está ejecutando en **modo de espera**, es decir, se encuentra en memoria "durmiendo", esperando alguna petición realizada por el software del cliente; en el momento en que ésta se presenta, el demonio "despierta" y atiende la petición que se le ha solicitado.

[6] A pesar de que el servidor resida en una computadora poderosa, lo que hace a una máquina ser servidor, no es su poder de cómputo, sino el ó los programas que en ésta se encuentren ejecutándose.

- **El cliente.** El *software* del cliente manda **peticiones** al software del servidor, quien —como ya vimos— las atiende al momento de recibirlas. Otra función del software del cliente, es proveer una interfaz al usuario; permitiendo al mismo manipular los datos que han sido recibidos desde el servidor.

- **La red.** Se refiere a la infraestructura de red donde corre la aplicación cliente/servidor.

Algunos de los protocolos TCP/IP que se encuentran en la capa de Aplicación del modelo DOD son:

**Telnet.** El propósito de este protocolo es la emulación de terminal. Permite al usuario en una máquina cliente (llamada **cliente Telnet**), accesar recursos de otra máquina denominada **servidor Telnet**, como si la máquina cliente se encontrara físicamente conectada al servidor, a pesar de encontrarse geográficamente distantes.

**FTP (File Transfer Protocol**; Protocolo de transferencia de archivos). Este protocolo permite transferir archivos entre dos máquinas que empleen este programa (pueden o no estar distantes geográficamente).

**TFTP (Trivial File Transfer Protocol)**. Esta es una versión más sencilla de FTP; no realiza todas las funciones de aquél, y manda bloques de datos mucho más pequeños.

**NFS (Network File System**; Sistema de Archivos en Red). Este protocolo está diseñado para compartir archivos. NFS permite a dos sistemas de archivos diferentes incorporarse en uno solo.

**SMTP (Simple Mail Transfer Protocol**; Protocolo Simple de Transferencia de Correo). Este protocolo esta diseñado para el manejo de correo electrónico (e-mail).

**LPD (Line Printer Daemon**; Demonio de Impresora en línea). Este protocolo está desarrollado para compartir impresoras.

**X Windows**. Diseñado para operaciones cliente-servidor, X Windows define un protocolo para la escritura de interfaces gráficas del usuario (GUI) basadas en aplicaciones cliente-servidor.

**SNMP (Simple Network Management Protocol**; Protocolo de Administración de Redes Común). Este protocolo es empleado en la administración de redes basadas en TCP/IP.

**Capa de transporte.**

El principal objetivo de la capa de transporte es aislar a la capa superior (de aplicación) de las complicaciones de la red. Esta capa recibe datos de la capa de

aplicación, con cualquier número de instrucciones, y comienza el proceso de preparar la información para ser enviada por la red a su destino. Los dos protocolos principales de esta capa son:

**TCP (Transmission Control Protocol**; Protocolo de Control de Transmisiones) Este protocolo toma grandes bloques de información de la capa superior de aplicación, y los rompe en segmentos; numera y secuencia cada uno de ellos, con el propósito de que el protocolo TCP destino pueda colocar los segmentos nuevamente en los largos bloques de información originales, que serán pasados a su vez a la capa superior. Antes del envío de los segmentos a las capas inferiores del modelo, el TCP origen contacta al TCP destino con el fin de establecer una conexión, formando lo que se conoce como ***circuito virtual***, este tipo de comunicación es denominado **orientado a la conexión** (véase al inicio de esta sección *conmutación de paquetes por circuito virtual*); durante esta fase inicial de la comunicación, las dos capas TCP (la emisora y la receptora) acuerdan la cantidad de información a transmitir antes de que el TCP receptor emita una señal de "acuse de recibo" (garantizando el arribo de los datos al destino). Esto establece una comunicación confiable para la capa superior del modelo.

**UDP (User Datagram Protocol**; Protocolo de Datagrama del usuario). Este protocolo puede ser usado en lugar de TCP, **nunca** los dos al mismo tiempo. UDP es considerado como un protocolo "delgado", pues no realiza todas las acciones que TCP sí hace. UDP recibe bloques de información de la capa superior y los rompe en segmentos. A cada segmento le es asignado un número para ser reensamblado en el bloque original por el UDP en la máquina destino. Este protocolo NO crea un *circuito virtual*; NO contacta al UDP destino antes del inicio de la transmisión, por lo que se le considera un **protocolo NO orientado a la conexión**; NO realiza el "acuse de recibo" de segmentos entregados; NO secuencia los segmentos, cada paquete posee un número para propósitos de reensamblado, pero UDP no pone atención en el orden en el que los segmentos llegan a su destino.

## Capa Internetwork[7]

Existen dos funciones principales para la capa de `interoperabilidad` de redes ó Internetwork:

- **Ruteo.** El protocolo principal de esta capa es el **IP (Internetwork Protocol**; Protocolo de interoperabilidad de redes), que podemos decir que "ve" todas las redes que se encuentran interconectadas en una red de redes. IP puede hacer esto ya que todas las máquinas en una red TCP/IP tienen una *dirección de software* llamada **dirección IP**. El protocolo IP ve la dirección de cada paquete y, entonces, utilizando un protocolo de *ruteo*, decide a dónde será enviado el paquete a continuación.

[7] Muchos autores hacen referencia a esta capa como **capa Internet**, pero no por citar a la red global, sino como un calificativo a la interoperabilidad de redes distintas, llámense éstas Internet o algunas otras redes de menor tamaño.

- **Proveer una `interfase` de red única a las capas superiores.** Sin esta capa, los programadores de aplicaciones necesitarían escribir en sus desarrollos –por simples que estos fueran– las interfases para cada protocolo de acceso a la red, provocando diferentes versiones de cada aplicación: una para `Ethernet`, una para `Token Ring`, `FDDI` etc. Esta interfase única de red es proporcionada por el protocolo IP.

Los protocolos existentes en esta capa son:

**IP (Internetwork Protocol**; Protocolo de interoperabilidad de redes). Este protocolo toma segmentos de la capa superior (la de transporte) y los fragmenta en `datagramas` (paquetes). IP reensambla los datagramas en segmentos del lado receptor ó destino. A cada datagrama se le asigna una dirección IP del origen y una dirección IP del destino. Cada dispositivo que recibe un datagrama realiza decisiones de ruteo en base a la dirección IP del paquete (*conmutación de paquetes*). Todos los protocolos en esta capa, al igual que todas las capas superiores del modelo, usan IP: **todos los caminos que pudieran existir dentro del modelo DOD, pasan por este protocolo**.

**ARP (Address Resolution Protocol**; Protocolo de Resolución de Direcciones). Cuando el protocolo IP tiene un datagrama que enviar, las capas superiores del modelo ya le han informado a dicho protocolo acerca de la dirección IP destino; sin embargo, IP debe informar también a los **protocolos de acceso a la red** (como `Ethernet`) la dirección de hardware destino. Si IP no sabe la dirección de hardware, emplea el protocolo ARP para encontrar esta información. ARP "interroga" a la red destino acerca de la máquina con la dirección IP especificada, recibiendo la dirección de hardware correspondiente.

**RARP (Reverse Address Resolution Protocol**; Protocolo de Resolución de Direcciones Inverso). Cuando una máquina en una red que emplea TCP/IP no tiene ningún tipo de dispositivo de almacenamiento, no hay manera de que ésta sepa inicialmente su dirección IP, pero conoce su dirección MAC. El protocolo RARP manda un paquete que incluye la dirección MAC y **realiza una petición** a una máquina (previamente designada) llamada **servidor RARP**, para que se le informe qué dirección IP es la asignada a la dirección MAC correspondiente.

**ICMP (Internet Control Message Protocol**). ICMP es un protocolo de administración para IP. Algunos de los eventos y mensajes a los que se refiere ICMP son:

- Destino inalcanzable. Si un ruteador no puede mandar un datagrama IP más lejos, emplea ICMP para mandar este mensaje de regreso a la máquina origen.
- `Buffer` lleno. Si la memoria `buffer` de un ruteador para recibir datagramas se llena, éste empleará ICMP para regresar este mensaje.
- Tiempo de vida finalizado. A cada datagrama IP le es asignado un número máximo de ruteadores por los que puede pasar, llamado **hops**. Si el paquete

alcanza el límite de **hops**, el último ruteador en recibir el datagrama lo desecha y emplea ICMP para informar a la máquina origen este acontecimiento.

**Capa de acceso a la red.**

Esta capa no está definida por el conjunto de protocolos TCP/IP. El modelo DOD señala los protocolos para la transmisión física de datos; hace referencia a los protocolos de comunicación específicos empleados en **cada tecnología de red en particular**. Sus funciones principales son:

- Recibir un *datagrama IP* (de la capa superior) y ***enmarcarlo*** en un flujo de bits (unos y ceros) dispuestos para su transmisión física.
- Especificar la **dirección física** ó **dirección MAC**. A pesar de que la capa Internetwork determina la dirección física destino, los protocolos en esta capa colocan la dirección MAC del dispositivo en el **marco (frame) MAC**.
- Asegurarse de que el flujo de bits que conforma el **marco (frame)** ha sido recibido correctamente.
- Especificar el método de acceso a la red física: `CSMA/CD` para redes `Ethernet` ó `token passing` para redes `Token Ring`, etc.

## Direccionamiento IP.

Uno de los puntos más importantes en cualquier discusión de TCP/IP es el direccionamiento IP. Una **dirección IP** es un identificador numérico asignado a **cada máquina en las redes que emplean TCP/IP**; ésta dirección es una dirección de software y, por lo tanto, no está grabada en la máquina ó en la tarjeta de red.

Una dirección IP está compuesta de 32 bits, los cuales están divididos en cuatro bytes. Existen dos métodos para describir una dirección IP:

- Decimal, como en 132.248.44.130
- Binario: 10000100.11111000.00101100.10000010

Ambos ejemplos representan la misma dirección IP.

De una manera instintiva se podría pensar que las direcciones están asignadas una por cada máquina conectada a Internet, lo que daría posibilidad de asignar más de 4294 millones de direcciones ($2^{32}$); sin embargo, ésto no es así, ya que **el esquema de direccionamiento en Internet es jerárquico de dos niveles**: en lugar de que los 32 bits de una dirección sean tratados como un identificador único, una parte de la dirección es

designada como **dirección de red**, y la otra parte como **dirección del nodo**; facilitando enormemente el ruteo.

La **dirección de red** es única para cada red[8]. Cada máquina en la misma red comparte una dirección de red común dentro de su dirección IP. Por ejemplo en la dirección 132.248.145.3, la porción 132.248 es la dirección de red (correspondiente en este caso a todas las redes de la UNAM). La **dirección del nodo** es un número único asignado a cada máquina dentro de una red común, en nuestro ejemplo sería 145.3 (correspondiente a la HP-730 de ENEP Aragón); define una máquina en particular dentro de una red dada; algunos autores la denominan **dirección del host**.

La manera en que una dirección IP es subdividida en una dirección de red y una dirección de host, es determinada por la **clase de red** de que se trate, las cuales fueron creadas basándose en el tamaño de las mismas. La tabla I.2 proporciona un resumen de las tres clases de redes.

| Clase | Formato | Patrón de los primeros bits | Rango decimal del primer byte de la dirección de red | Número máximo de nodos por red |
|---|---|---|---|---|
| A | Red.Nodo. Nodo Nodo | 0 | 1-127 | 16,777,216 |
| B | Red.Red. Nodo.Nodo | 10 | 128-191 | 65,534 |
| C | Red.Red. Red.Nodo | 110 | 192-223 | 254 |

**Tabla I.2 Clases de redes.**

En busca de un ruteo eficiente, los diseñadores de Internet definieron reglas para los primeros bits de cada una de las direcciones de red (véase ésto en la tabla I.2). Por ejemplo, como un ruteador sabe que una red clase A siempre comienza con cero, podrá rutear el paquete después de leer solamente el primer bit de su dirección. La figura I.4 ilustra cómo están definidos los bits iniciales de una dirección de red.

Existen algunas direcciones IP que están reservadas para propósitos especiales. La tabla I.3 lista estas direcciones especiales.

| Dirección | Función |
|---|---|
| Dirección de red compuesta totalmente de ceros<br>0.0.?.? | Hace referencia a "esta red" |
| Dirección de red compuesta totalmente de unos<br>1.1.?.? | Hace referencia a "todas las redes" |
| 127.0.0.1 | Reservada para pruebas de `loopback`. Designa el nodo local y permite a ese nodo mandarse paquetes a sí mismo sin generar tráfico en la red |
| Dirección del nodo compuesta totalmente de ceros<br>?.?.0.0 | Hace referencia a "este nodo" |
| Dirección del nodo compuesta totalmente de unos<br>?.?.1.1 | Hace referencia a "todos los nodos" en la red especificada. Por ejemplo, 132.248.255.255 significa "todos los nodos" de la red 132.248 (red clase B) |
| Dirección IP completa compuesta por unos<br>(equivalente a 255.255.255.255) | Hace referencia a todos los nodos de la red actual. Es conocida como dirección de `broadcast`. |

**Tabla I.3 Direcciones IP reservadas.**

[8] Para obtener una dirección IP que no esté en conflicto con alguna otra en Internet, es necesario solicitarla al NIC ó Network Information Center en **hostmaster@internic.net**. Si la red no va a tener acceso a Internet, puede asignársele cualquier dirección IP que el administrador decida.

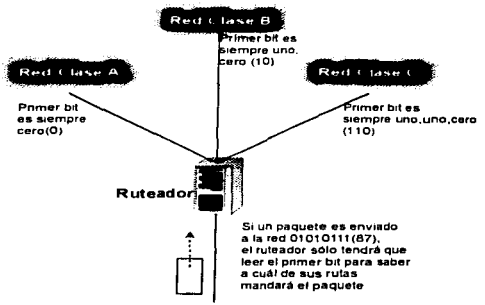



**Figura I.4.** Ruteo mediante los bits iniciales de una dirección de red.

**Redes Clase A.**

En las direcciones de redes clase A, el primer byte es asignado a la dirección de red, y los tres bytes restantes se emplean para la dirección del nodo. El formato es: ***Red.Nodo.Nodo.Nodo***. Por ejemplo, en la dirección IP 49.22.102.70, 49 es la dirección de red y 22.102.70 es la dirección del nodo; cada máquina en ésta red en particular tiene el número 49 como dirección de red.

El máximo número de redes para esta clase es 128 ($2^7$) y no 256 ($2^8$), pues el primer bit del octeto es empleado para la designación de la clase de red (véase tabla I.2 y figura I.4). Como las direcciones 0 y 127 están reservadas, queda un **número máximo de redes clase A de 126. El número máximo de nodos en esta clase es de 16,777,216** ($2^{24}$).

**Redes Clase B.**

En las direcciones de esta clase, los dos primeros bytes son empleados para la dirección de red y los dos restantes para la dirección del nodo, con el formato: **Red.Red.Nodo.Nodo**. En una red con dirección IP 132.248.145.5, 132.248 es la dirección de red y 145.5 es la dirección del nodo.

El **número máximo de redes clase B es 16,384** ($2^{14}$), recordemos que los 2 primeros bits son empleados para la designación de la clase de red (una dirección de red clase B siempre empieza con 1 y 0 binario), dejando únicamente 14 bits para manipular

la dirección de red. El **número máximo de nodos es de 65,534** ($2^{16}$), descartando ya las dos direcciones reservadas.

**Redes Clase C.**

En este tipo de direcciones, los tres primeros bytes son empleados por la dirección de red, y el restante por la dirección del nodo, con el formato: **Red.Red.Red.Nodo**. Por ejemplo, en la dirección IP 198.21.74.102, la dirección de red es 198.21.74 y la del nodo 102.

En una red clase C, los tres primeros bits de la dirección de red son: 1,1,0, por lo que el **número máximo de redes de esta clase es 2,097,152** ($2^{21}$). El **número máximo de nodos es de 254** ($2^8$ = 256, menos las dos direcciones reservadas).

**Clases adicionales de redes.**

Las direcciones **clase** D son consideradas adicionales debido a su uso especial y reservado, pues son empleadas para **paquetes multicasting**. Una **transmisión multicasting** es empleada cuando una máquina desea transmitir algo a múltiples destinos.

Es importante destacar que el tipo de direccionamiento expuesto en esta sección es esencial para las máquinas conectadas a Internet, pues gracias a él es posible rutear paquetes de información eficientemente de una red a otra, logrando que éstas interoperen. **Este tipo de direccionamiento, junto con el ruteo y el protocolo de conmutación de paquetes (TCP/IP) son el corazón de la tecnología Internet, responsables tecnológicos directos del éxito desmesurado de la red global.**

## Mapeo de nombres de hosts a direcciones IP.

Es posible asignar a las máquinas que conforman una red IP un nombre descriptivo llamado **nombre del host** ó **hostname**, los cuales son nombres simbólicos para una máquina, asignados por el administrador de la red. Los **nombres de hosts** facilitan el acceso de los usuarios a diversas máquinas, pues un usuario no deberá recordar (ni siquiera conocer) la dirección IP de una máquina dada, bastará con utilizar su **hostname**. Por ejemplo: en lugar de que el usuario escribiera `telnet 132.248.145.5`, bastaría con escribir `telnet indy` ó `telnet indy.aragon.unam.mx`, logrando el mismo resultado.

Los nombres del host pueden ser catalogados mediante **dominios**. Los nombres dominio han sido creados por las autoridades de Internet para describir tipos genéricos de organizaciones. Algunos ejemplos son:

| | |
|---|---|
| com | Comercio |
| edu | Educación |
| gov | Gobierno de los E.U. |
| mil | Milicia de los E.U. |
| net | Una organización administrativa para una red |
| org | Organizaciones, generalmente privadas, que no corresponden a los criterios anteriores |

Existen también dominios para nombre de países:

| | |
|---|---|
| de | Alemania |
| it | Italia |
| nz | Nueva Zelanda |
| mx | México |
| jp | Japón |
| uk | Reino Unido |

El formato de un nombre de host con un nombre dominio es: *Hostname.Dominio*. Por ejemplo: **internic.net**. Los dominios no son asignados arbitrariamente por los administradores, sino que deben de ser otorgados por el NIC (**Network Information Center**), con la finalidad de que –al igual que las direcciones IP– no existan nombres duplicados y –en consecuencia– conflictos en Internet.

Mediante el **mapeo de nombre de host a direcciones IP**, es posible simplificar comandos que implican las confusas direcciones numéricas IP. Existen tres maneras de realizar el mapeo citado: *tablas host; DNS, y NIS*.

Una **tabla host** es un archivo ASCII que asocia nombres de hosts con su dirección IP. Este archivo varía en nombre de un sistema operativo a otro, sin embargo, siempre es consultado cuando es necesario "resolver" un nombre dominio. En UNIX, este archivo se encuentra en **/etc/hosts**.

El **DNS** ó **Domain Name System** (Sistema de nombres dominio) es un mecanismo que ayuda a los usuarios localizar el nombre de una máquina y mapear su nombre a una dirección IP. Existen máquinas a lo largo de Internet llamadas **servidores de nombres**, que guardan una base de datos con un número considerable de nombres de hosts; esta base de datos está dispuesta de una manera jerárquica, comenzando en la raíz y descendiendo al dominio, subdominio y finalmente el nombre del host. Esta estructura es parecida a un árbol invertido (véase figura 1.5).

Los servidores de nombres manejan **zonas**. Una zona comienza en un nodo del árbol y abarca todas las ramas bajo éste. El servidor puede delegar autoridad sobre una subzona a otro servidor quien controlará la información bajo su dominio. DNS consiste en zonas anidadas en las cuales los servidores de nombres operan, reconociendo a sus vecinos arriba y abajo de ellos. Para garantizar la eficiencia, cada zona tiene al menos dos servidores activos (servidor primario y secundario) que proveen la misma información. El protocolo DNS provee mapeo de nombre host a direcciones IP.

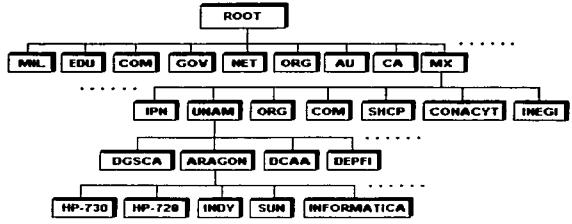



**Figura I.5.** Jerarquía DNS (Domain Name System)

Los servidores NIS ó **Network Information Services** (Servicios de información de red) pueden ser creados para un grupo de computadoras, denominado **dominio**; estos servidores NIS contienen bases de datos llamadas **mapas**, los cuales proveen traducción de nombres host a direcciones IP. La principal diferencia entre NIS y DNS es que los servidores NIS cubre un área más pequeña, pues éstos hacen referencia únicamente a grupos de computadoras, y no a todo Internet.

## I.3 Servicios de Internet.

Existe gran cantidad de "programas" que proporcionan servicios varios a los usuarios de Internet, sin embargo, los más ampliamente difundidos son los siguientes:

### Correo electrónico.

El correo electrónico es comúnmente referido como ***e-mail.*** Su función es la de permitir a las personas con acceso a Internet el **envío de mensajes** (correo) de una computadora a otra en cualquier parte del mundo; estos mensajes pueden ser simples textos, o bien complejos archivos que incluyan hojas de cálculo, gráficos, fotografías etc. Este servicio es el más empleado entre las personas con acceso a la red global. Presenta una interfaz funcional a través de una serie de comandos que no serán discutidos en este trabajo. Permite –al igual que la mayoría de los servicios de Internet– el uso de hostnames y/o direcciones IP.

## Veronica.

Veronica (*Very Easy Rodent-Oriented Net-wide Index to Computerized Archives; Índice fácil de archivos computarizados, orientado al ratón y para toda la red*) es un mecanismo de búsquedas, diseñado para realizarlas en todos los menús de los sitios Gopher del mundo. Este programa visita con regularidad todos los servidores en Internet, rastrea los menús y crea un índice de gran extensión (de cerca de 10 millones de elementos), el cual es empleado para buscar la información solicitada por el usuario al momento de hacer uso de esta herramienta.

## WAIS.

WAIS (*Wide Area Information Servers: servidores de Información de área amplia*) busca la información que reside en el interior de archivos o documentos; al igual que varios servicios de Internet, éste descansa sobre un índice creado por administradores en varios servidores WAIS, los cuales describen el contenido del archivo. Los archivos índices son coleccionados en bases de datos de temas relacionados; por ejemplo, un servidor WAIS puede incluir una colección de títulos acerca de temas de películas, a diferencia de otro que pudiera contener información de recetas de cocina. WAIS da prioridades al resultado de las búsquedas, pues los archivos que contienen más ocurrencias de la palabra o frase buscada aparecen al principio de la lista de resultados.

## World Wide Web.

Como hemos podido ver, Internet ha emergido como una fuente gigantesca de información, accesible sólo mediante una serie de interfaces no muy amigables. Los comandos básicos para Telnet, FTP, Archie, WAIS, e incluso e-mail son muy poderosos pero no intuitivos, y el rápido aumento en el uso de la red global resulta a su vez en un incremento en el número de usuarios que no tienen ni la paciencia ni el deseo de aprender a manejar estas interfaces. Entra en escena WWW, que fue diseñado con la idea de permitir investigación colaborativa para presentar trabajos completos con gráficas, ilustraciones, sonido, video y algunas otras características adicionales. **El proyecto WWW ha hecho posible el desarrollo de interfaces accesibles y atractivas, fáciles de usar en Internet.**

### Historia de WWW.

El origen de WWW tuvo lugar en el **Laboratorio Europeo de Física de Partículas en Suiza** (CERN por su traducción del francés), donde un científico llamado **Tim Berners-Lee** propuso en 1989 un sistema denominado de `hipertexto`[9], que permitiera el compartimiento eficiente de información entre los miembros de la

[9] El prefijo **hiper** denota exceso ó superioridad; **hipertexto** describe un sistema de información compuesto de documentos de texto con "marcas" ó "señales" especiales (ligas), que al ser seleccionadas nos conducen a otro documento totalmente diferente al original, generalmente relacionados en sus contenidos.

comunidad física europea. Los puntos importantes que Berners-Lee expresó en su propuesta eran:

- Una interfaz de usuario que fuera consistente en cualquier `plataforma` y `sistema operativo`, que permitiera a los usuarios accesar información de diferentes computadoras geográficamente distantes.

- Un esquema de dicha interfaz para accesar una gran variedad de tipos de documentos y protocolos de información.

- Proveer **acceso universal**, que permitiera a cualquier usuario de la red accesar cualquier información dentro de la misma.

Para 1990 existía ya un prototipo de este sistema (llamado www) corriendo en varias máquinas del CERN, pero fue hasta 1991 que dicha interfaz fue integrada a la red del laboratorio suizo. Hasta este año, la interfaz desarrollada era únicamente basada en texto, pero empleaba plenamente el sistema *hipertexto* propuesto. El CERN comenzó a difundir la existencia de su sistema a través de Internet (e-mail, grupos de discusión, ftp, gopher, etc...), logrando que su popularidad fuera aumentando paulatinamente.

En 1993 se desarrolló la primera interfaz gráfica (*cliente www* ó `paginador`[10]), llamada **Mosaic.** A partir de este momento, el desarrollo de Berners-Lee adquirió la capacidad de transmisión de información gráfica y animada, pues era posible incluir en los documentos –además de hipertexto– imágenes, sonidos, video, etc., dando lugar a la `hipermedia`. En julio de 1994, el Instituto Tecnológico de Massachussets (**MIT**) y el CERN crearon la **Organización WWW**, conocida ahora como el **Consorcio World Wide Web** ó **W³C**[11], el cual guía el desarrollo técnico y estándares para la evolución del Web[12].

Actualmente W³C está conformado por universidades e industrias privadas de todos los países del globo. La popularidad alcanzada por WWW desde 1993 ha sido increíble, pues ha utilizado la red global para darse a conocer en todo el mundo. En la actualidad se calcula que existen más de 300,000 servidores web, haciendo de ésta **la tecnología más rápidamente asimilada y desarrollada en la historia de la humanidad**. En la actualidad, W³ supera el tráfico generado por gopher y ftp juntos (en términos de bits transmitidos).

La popularidad que tiene el Web ha degenerado la concepción de éste en el entendimiento popular, pues a pesar de ser un servicio más de Internet, en ocasiones es visto como un equivalente del mismo. **WWW es un sistema muy distinto de la red global:** el Web **NO es una red**, sino una aplicación (un conjunto de programas de software); además, puede ser usado en diferentes tipos de redes individuales sin acceso a Internet (`Intranet`) o incluso en computadoras individuales.

---

[10] El nombre en inglés de "paginador" es **browser**, cuyo uso es seguramente más común en el argot informático.
[11] Puede visitarse la dirección del Consorcio WWW en **http://www.w3.org.**
[12] El término **web** con minúscula se refiere al hipertexto local que crea un desarrollador (un documento hipertexto cualquiera); en cambio **Web** con mayúscula, lo ocupo como sinónimo de WWW.

**Definición formal.**

WWW es un ***sistema de comunicación e información basado en hipertexto***, empleado comúnmente en *Internet*, con comunicaciones de datos operando acorde a un *modelo cliente/servidor*. Los *clientes web ó paginadores* pueden accesar información *hipermedia* y *multiprotocolo* empleando un *esquema de direcciones* (que es el mismo sistema de *dominios* empleado por TCP/IP).

El esquema *cliente/servidor* expuesto con anterioridad, es resumido en la figura I.6, donde se muestra el flujo de una petición de un cliente a un servidor, y la transmisión de resultados del servidor al cliente. Un cliente puede accesar varios servidores empleando el ó los protocolos que ambos entienden.

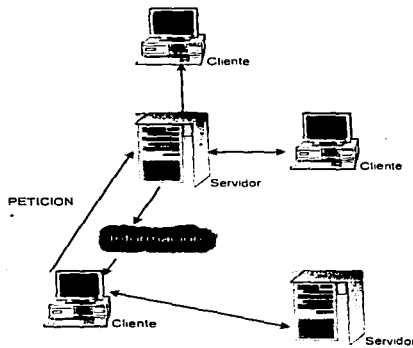



**Figura I.6.** Modelo cliente-servidor para comunicaciones de datos

**Elementos de WWW.**

La *unidad de trabajo* de WWW es el ***documento*** (mejor conocido como **página**), se encuentra escrito en **HTML** (**Hipertext Markup Language**; Lenguaje de marcación en hipertexto), que es una derivación simplificada de **SGML** (**Standard Generalized Markup Language**; Lenguaje de marcación estándar generalizado), un estándar internacional para el procesamiento de información en formato texto. Expongo ampliamente HTML en el capítulo III de éste trabajo de tesis.

Para accesar los recursos disponibles en WWW (documentos, gráficos, sonidos, etc.) se emplea una **dirección** que identifica de manera única dichos recursos, conocida como **URL (Uniform Resource Locator**; Localizador Uniforme de Recursos). Un URL es una cadena de texto cuyo formato es:

**esquema://host:puerto/URI**

donde:

*esquema* es una de las reglas ó protocolos que permitirán mandar y/o recibir información, serán válidos algunos protocolos de Internet como ftp, gopher, WAIS, telnet, etc.; sin embargo, para propósitos del presente trabajo, emplearemos exclusivamente el **protocolo HTTP** (lenguaje nativo de WWW);

*host* es el **dominio completo** de la computadora en que se encuentra el recurso deseado; por ejemplo **indy.aragon.unam.mx** es un dominio completo, no así sólo **indy**; WWW emplea el mismo tipo de direccionamiento que TCP/IP (véase capitulo I);

*puerto*[13] es un número que identifica el puerto a través del cual el servidor realiza las transacciones deseadas; este número deberá especificarse en caso de que el protocolo indicado se encuentre instalado en algún puerto diferente al estándar, y

*URI* es la "ruta" ó "path" del recurso dentro de un host; es decir, es una "referencia completa" al recurso que se desea accesar. La ruta especificada es una `ruta virtual` establecida al momento de la instalación del servidor web.

Por ejemplo, el URL **http://indy.aragon.unam.mx/tesis/indice.html**, nos dice que se empleará el protocolo **http**; se accesará a un host cuyo dominio completo es **indy.aragon.unam.mx**; que este host hará sus transacciones http a través del puerto estándar (puerto 80), por lo que no se especifica, y que el recurso que se consultará (ó URI) será **/tesis/indice.html**.

Cualquier servidor web ó documento dentro de éste, es referenciado mediante su URL único, ésto implica que una página específica dentro de un sistema de páginas web puede ser accesada desde cualquier parte en WWW, basta con conocer el *URL* correcto para desplegar la página deseada.

**Funcionamiento.**

El protocolo nativo de WWW es **HTTP (Hipertext Transfer Protocol**; Protocolo de Transferencia de Hipertexto). HTTP es un ***protocolo de conmutación de paquetes con conexión virtual no dedicada***; es decir, cada circuito virtual creado es eliminado una vez que se han cumplido las cuatro etapas que conforman una transacción http (expuestas abajo); además, no retiene información alguna acerca de sesiones ó

[13] En los servidores basados en el sistema operativo UNIX, todos los servicios básicos (telnet, ftp, gopher, e-mail, http, etc.) son identificados mediante un número de puerto estándar. Por ejemplo: telnet recibe el puerto 23; ftp el 21 (para control) y el 20 (para datos), y http (WWW) el 80.

peticiones anteriores de ningún cliente. Cuando este protocolo trabaja en Internet, lo hace sobre el conjunto de protocolos TCP/IP, permitiendo que sea éste el que se encargue de la conexión inicial entre el cliente y servidor; una vez establecida la conexión, HTTP se encarga de determinar el objetivo de la misma (servir o recibir documentos web). Se ubica en la capa de aplicación del modelo DOD (ver figura I.3). Existen cuatro fases secuenciales en una transacción HTTP: conexión, petición, respuesta y desconexión.

**Conexión (connection).** En esta fase, el cliente intenta conectarse a la *dirección Web* ó *URL* del servidor (en realidad al software de servidor que se está ejecutando en la máquina remota). El software de cliente reconoce el momento en que la conexión ya se ha realizado, pasando a la siguiente fase.

**Petición (request).** Una vez completada la conexión, el cliente envía una petición al software de servidor para ejecutar una acción especifica. Esta petición, incluye la siguiente información: *Método de petición* ***(request method)***, que se refiere a la manera en que se comunicarán cliente y servidor, así como la forma en que ambos llegarían a intercambiar información; *URI solicitado* ***(request URI)***, refiriéndose al recurso exacto al que se aplicará la petición del cliente; *Cabecera de la petición* ***(request header files)***, que contiene información adicional necesaria que determina el comportamiento final del servidor. Los tres aspectos anteriores serán explicados con mayor amplitud en el capítulo IV de esta publicación (Programación de CGI).

**Respuesta (response).** La acción real de una conexión HTTP es realizada en esta fase; aquí, el software del servidor intenta satisfacer la petición del cliente. Se responde al cliente en forma de una línea de estado, incluyendo información acerca del servidor y los datos solicitados, o bien, un mensaje de error.

**Desconexión (close).** En esta fase, la conexión entre el cliente y el servidor es finalizada. Por lo general, cada transacción comienza con el cliente que establece una conexión, y termina con el servidor finalizando la misma después de atender a la petición del cliente.

En WWW, el **software de cliente** recibe el nombre de ***paginador***, cuya tarea es la de desplegar documentos *web* y permitir al usuario la selección de `ligas de hipertexto`. Existen paginadores para diferentes sistemas operativos y plataformas de cómputo; los hay basados en texto (que han comenzado a caer en desuso) y con capacidad gráfica; éstos últimos son los más comunes y los que permiten apreciar mejor las ventajas que ésta tecnología nos proporciona. Mediante un paginador gráfico, es posible "navegar" a través de Internet sin necesidad de emplear comandos, basta con el uso del mouse para posicionarse sobre la liga de hipertexto deseada, misma que nos llevará a algún otro documento web.

**Importancia de WWW.**

WWW, gracias a los elementos que lo conforman, así como la interfaz gráfica (fácil de usar y muy atractiva) que proporciona, se ha convertido en un flexible "sistema

de comunicación" que puede ser empleado en varios contextos: desde comunicación individual –mediante páginas personales– hasta comunicación grupal y masiva. Además, desempeña varias funciones interesantes:

- **Entrega de información.** Un paginador web permite al usuario la visualización de información en hipertexto; la estructura de éste nos permite ser **selectivos** en cuanto a la información que recibimos dadas las múltiples opciones de documentos que es posible accesar.

- **Comunicación.** Los usuarios de WWW pueden crear "foros" para compartir y discutir información, así como auxiliar a otros usuarios a hacer contacto con las personas ó documentos que ellos desean.

- **Interacción.** Mediante el uso de *compuertas* ó **CGI (Common Gateway Interface**; Interfaz Común basado en Compuertas) el Web proporciona interacción dentro de una aplicación cualquiera. Las compuertas pueden incluso permitir a un usuario modificar ó incorporarse a una estructura de información.

- **Procesamiento de información.** *La programación de compuertas puede* ***proveer una interfase a otras aplicaciones y programas para el procesamiento de información.*** *Basado en selecciones del usuario, una aplicación WWW puede regresar información procesada en otros programas o aplicaciones* (un servidor de Base de Datos, por ejemplo).

Actualmente, existen gran variedad de aplicaciones desarrolladas en WWW, que van desde páginas personales hasta complejos programas de reservaciones de hotel y registro de vuelos aéreos. Todas ellas no sólo se basan en la tecnología Web, sino que mediante el uso de CGIs explotan las capacidades de otros programas, incrementando la eficiencia y utilidad de los mismos. Esta interacción con otras aplicaciones es totalmente transparente para el usuario, sin embargo, para un diseñador implica una serie de técnicas y procesos ordenados que debe seguir para alcanzar su meta; estas técnicas son el tema central del siguiente capítulo.

# 2

# Proceso de desarrollo de aplicaciones

En el capítulo anterior conocimos la historia, funcionamiento y servicios de Internet; se señaló que el desarrollo de aplicaciones para la red global puede conseguirse haciendo uso de sus servicios: ftp, telnet, gopher, etc.; sin embargo, la utilización de algunos de ellos implica el manejo de interfaces no muy amigables, e incluso de comandos confusos y difíciles de aprender para personas que posiblemente tengan un conocimiento pobre de la computadora, o bien, vean a ésta como un mal necesario en sus actividades.

La naturaleza cliente-servidor de estos servicios obligan al desarrollador a contemplar aspectos importantes tanto del lado del cliente como del servidor[1]. La programación del servidor dependerá directamente del problema a resolver; variará en lenguaje, tecnología y metodología de desarrollo dependiendo de la aplicación a desarrollar: bases de datos, generación de estadísticas, consulta remota de servidores, obtención de información o datos, etc. Nuestro estudio acerca de la metodología, lenguaje y demás detalles de la programación del servidor, la abordaremos al momento de definir nuestro caso de estudio (capítulo V).

Por lo que respecta al desarrollo de la parte cliente de la aplicación, es necesario seleccionar la interfaz que habrá de emplearse ó **API** (**Aplication Program Interfase**; Interfaz del Programa de Aplicación); para la elección de ésta, deberán comprenderse los servicios de Internet (recordemos que uno de nuestros objetivos es el desarrollo de aplicaciones en la red global). La tabla II.1 muestra algunas reglas que el software de cliente debe cumplir en una aplicación cliente-servidor, confrontadas con los servicios más importantes de Internet.

Para un programador de aplicaciones resulta siempre importante desarrollar sistemas de fácil manejo, pues ello implica más posibilidades de penetración y aceptación de sus productos en un mercado ampliamente disputado, por lo que programar en los servicios de Internet no apropiados, simplemente le dificultan más el camino. Basándonos en ésto y rescatando el servicio mas agraciado en la tabla II.1, nuestro ambiente de desarrollo recae en una herramienta de reciente creación y gran poder: **World Wide Web** ó **WWW**.

---

[1] La arquitectura cliente-servidor distingue tres entidades diferentes: la red, el cliente y el servidor. El desarrollador de aplicaciones deberá preocuparse únicamente de las dos últimas, ya que la entidad red se refiere a la infraestructura donde corre la aplicación, que a pesar de estar íntimamente ligada con aquella, no es de la injerencia del programador pues las aplicaciones en Internet son independientes de la plataforma e infraestructura de cómputo empleada. Véase TCP/IP en el capítulo I.

| Aspecto de Selección | E-Mail | Telnet | FTP | Archie | Gopher | Veronica | WAIS | WWW |
|---|---|---|---|---|---|---|---|---|
| Portable. | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Interactúa con servidores de otros servicios | | | | | | ✓ | | ✓ |
| Realiza transacciones de información con el servidor en tiempo real. | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Aplicaciones de fácil desarrollo, aprendizaje y uso. | | | | | ✓ | | | ✓ |
| Permite hacer pruebas de la aplicación localmente (sin conexión a la red) | | | | | | | | ✓ |
| Proporciona las herramientas suficientes para diseñar interfaces de usuario | | | | | ✓ | | | ✓ |
| Multimedia: gráficos, video, audio, etc. | | | | | | | | ✓ |

**Tabla II.1** Algunas reglas para la evaluación del software cliente en una aplicación cliente-servidor.

WWW provee una interfaz amigable para el usuario, intuitiva y de gran capacidad de presentación (hipertexto, gráficos, sonidos, etc.), satisfaciendo las tendencias actuales del desarrollo de aplicaciones, en donde se ha hecho necesaria la creación de sistemas eficientes, de fácil aprendizaje y manejo, cuyo uso se traduzca en un incremento en la productividad. World Wide Web hace posible ésto, proveyendo no sólo una interfaz amigable, sino también un medio muy eficiente de comunicación y una excelente plataforma para el desarrollo de aplicaciones dentro de Internet.

Ahora bien, una vez seleccionada la interfaz que habremos de emplear como base para el desarrollo de la parte cliente de nuestra aplicación (cualquiera que ésta sea), cabe mencionar que la gran capacidad de WWW como un medio de comunicación hace que el desarrollo de aplicaciones dentro del mismo implique una metodología especial (y no sólo saber escribir HTML). El desarrollo de sistemas de calidad para WWW requiere de procesos de información bien razonados, dinámicos y creativos, poniendo especial atención en las necesidades y experiencias de los usuarios, cuidando los importantes detalles que el manejo de una herramienta con tan destacable poder de comunicación tiene.

## II.1 Metodología de desarrollo en WWW.

Los procesos comprendidos en el desarrollo de aplicaciones en WWW, pueden parecer a primera vista gran cantidad de trabajo extra; sin embargo, un ***sistema de páginas hipertexto*** **bien desarrollado** vale mucho más que un conjunto de paginitas

unidas sin razonamiento alguno. Para desarrolladores casuales[2] del Web, esta metodología puede incluso ayudarles a mejorar la estructura de la información que emplean en sus desarrollos, mejorando la calidad y efectividad de los mismos.

Los procesos y elementos que intervienen en el desarrollo de aplicaciones para el Web están interconectados, y las decisiones que los desarrolladores toman dependen de éstas interconexiones. Existe cierta redundancia en esta metodología; si alguno de los elementos o procesos es débil, algún otro elemento o proceso más robusto puede llegar a compensarlo. Por ejemplo, una buena implementación puede compensar un mal diseño, sin embargo, es importante evitar éstas debilidades.

Los procesos en el desarrollo de aplicaciones en WWW son:

- **Planeación.**
- **Análisis.**
- **Diseño.**
- **Implementación.**
- **Promoción.**
- **Innovación.**

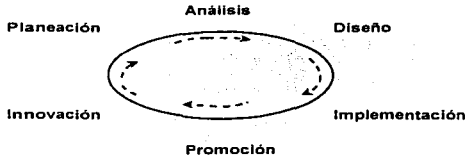



**Figura II.1.** Metodología de desarrollo de aplicaciones en WWW.

## II.2 Planeación.

La planeación es un aspecto crucial en el desarrollo de aplicaciones, pues es aquí donde son tomadas varias decisiones que afectan el diseño, implementación y posterior promoción de nuestra aplicación.

[2] La frase *desarrolladores casuales* se refiere a personas que no se dedican profesionalmente al desarrollo de aplicaciones.

## Principios de la planeación en WWW.

Los principios de la planeación guían al desarrollador a entender los límites de la aplicación, así como definir un punto central del cual partir para la materialización de ésta. La naturaleza dinámica de WWW, provoca que la planeación sea comprendida como un proceso continuo, en el que entidades de diversos autores, y relaciones de información continuamente cambiante sean tomadas en cuenta.

### Límites de la planeación del Web.

Existen ciertos factores sobre los que el desarrollador no tiene control alguno; sin embargo, el primer paso en el proceso de planeación será reconocer estos factores y considerar de qué manera pueden limitar a ésta para la aplicación en particular. Estos factores son:

**Comportamiento del usuario.** El Web es un sistema dinámico, basado en selecciones del usuario, por lo que no hay forma en que un desarrollador pueda controlar la manera en que se accesará y empleará la información de la aplicación.

**Paginador y despliegue del usuario.** Debido a la naturaleza cliente-servidor del Web, es imposible conocer el paginador que el usuario estará utilizando para desplegar en su computadora la aplicación WWW. Existen numerosos paginadores con capacidad gráfica, algunos basados en texto, así como nuevos productos que seguramente proveerán más y diferentes capacidades de las que los paginadores actuales nos proporcionan. En la planeación del Web, los desarrolladores necesitan considerar qué información será esencial, de tal manera que nada de ella se pierda cuando se empleen paginadores con capacidades de despliegue inferiores a las que uno espera (que no desplieguen gráficos, o bien, que no soporten algunas extensiones HTML).

**Ligas dentro y fuera del Web.** En cualquier página podemos encontrar ligas que hacen referencia a recursos de WWW e Internet fuera del sistema desarrollado. Estas ligas se encuentran totalmente aisladas del control del desarrollador (pues pertenecen a su propia aplicación web), así que al incluirlas, uno podría encontrarse con que la liga hace referencia a páginas viejas, o que han cambiado debido a las disposiciones de sus autores.

### Oportunidades en la planeación del Web.

Así como existen elementos sobre los que el desarrollador tiene poco o ningún control, existen algunos otros sobre los que se tiene un dominio absoluto:

El planeador deberá distinguir los puntos del sistema en que éste permita la interacción del usuario, a través de *formas* u otro método de intercambio de información; deberá **definir las políticas y propósitos de la información que se intercambie con los usuarios**. Planificar la interactividad de un sistema implica la identificación y análisis

de la audiencia de la aplicación; así como la definición de las necesidades del usuario y los mecanismos mediante los cuales dichas necesidades serán satisfechas.

El comportamiento del usuario no puede ser previsto pero sí canalizado, por lo que es obligación del planeador **definir** ligas de hipertexto, señales u otro tipo de indicaciones que permitan al usuario **conocer fácilmente todos los caminos posibles dentro de la aplicación**, antes de que éste decida abandonarla por falta de orientación dentro de la misma.

Es necesario **identificar las necesidades de actualización de información en la aplicación**. WWW es un sistema de información de rápida y fácil actualización, por lo que es necesaria una actitud dinámica que permita a los usuarios tener la certeza de que los datos proporcionados por la aplicación son lo suficientemente recientes como para confiar en ellos.

## Técnicas y procesos en la planeación.

La planeación por lo regular se presenta dentro de un contexto mucho más general y amplio que el de la simple identificación de componentes técnicos de un conjunto de páginas HTML; frecuentemente –en especial para grandes organizaciones– la comunicación a través de WWW es parte de un esfuerzo estratégico para alcanzar a los usuarios.

### Planeación del sistema.

El primer paso en la planeación del sistema web es **definir** la manera en que éste pudiera jugar un papel relevante en las necesidades de comunicación de la organización. Este proceso de definición puede comenzar con la evaluación de los medios de comunicación que ya se estuvieran empleando (radio, televisión, prensa, etc.); nuestra aplicación no necesita reemplazar todos los métodos existentes, sino complementarlos y sustituir sólo algunos de ellos (los menos eficientes).

Después de la definición sigue la **integración**, donde la aplicación web deberá amalgamarse con la infraestructura de comunicación ya existente en la organización (si es que existe), por ejemplo, algún servidor ftp ó gopher que también sean empleados como medio de comunicación del organismo. El objetivo de esta integración es la elaboración de un plan para "ligar" los elementos de nuestra aplicación web a flujos de información ó comunicación ya establecidos.

Finalmente, sigue la **diferenciación**, donde la aplicación web deberá mostrar ventajas claras sobre alguno de los medios tradicionales de comunicación de la organización, o de otra forma, **convendrá detener el desarrollo del sistema**. La tecnología de punta implicada en WWW no significa que las aplicaciones web sean la solución a todas las necesidades de cualquier organismo, sino que es necesario

distinguir y evaluar las tareas en que el sistema web cumple satisfactoriamente, y decidir si se continúa con su planeación o su uso.

**Técnicas individuales en la planeación del Web.**

**Información de la Audiencia.** Para que exista una comunicación precisa, es necesario que un desarrollador sepa **lo que se quiere comunicar y a quién.** La información que se tenga del "destino final" ó auditorio, es crucial para el establecimiento de una comunicación exitosa con el usuario, asegurando una mejor transmisión de los mensajes o información que se desea dar a conocer.

**Establecimiento del objetivo.** Establecer un objetivo general sirve como guía a lo largo de todo el desarrollo de la aplicación; sin un objetivo, los analistas no cuentan con ninguna base para evaluar el sistema y decidir si éste trabaja eficientemente o no.

**Dominio de la información.** Es necesario conocer la amplitud, el dominio de la información que habrá de emplearse para el desarrollo del web; deberá incluirse la que se proporcionará al usuario, así como la que los desarrolladores deberán conocer para realizar un buen trabajo. Además, deberán hacerse consideraciones acerca de las formas en que ésta será obtenida, así como de planes para la actualización de la información que necesite ser renovada continuamente.

## II.3 Análisis.

Las técnicas de análisis ayudan a verificar los elementos del Web en un estado operativo ó bien en planeación. Estos procesos de análisis contemplan desde la validación técnica de una implementación HTML, hasta el estudio del contenido y diseño de una aplicación ya existente o bien, que apenas se encuentre en planeación.

### Principios del análisis en WWW.

Los objetivos generales que un analista persigue son:

- Verificar que la aplicación trabaje **retóricamente** (¿La aplicación cumple con los objetivos iniciales y está enfocada a la "audiencia" planeada?), **técnicamente** (¿La aplicación es funcional y está implementada con las especificaciones HTML previstas?), y **semánticamente** (¿Es correcto, relevante y completo el contenido informativo de la aplicación?).

- Hacer recomendaciones a los demás procesos del desarrollo de la aplicación: recomendar una nueva planeación (total o parcial) de la aplicación; recomendar

mantenimiento a los programadores; dar reportes a los promotores; dar puntos de vista a los innovadores para mejorar el contenido u operación de la aplicación.

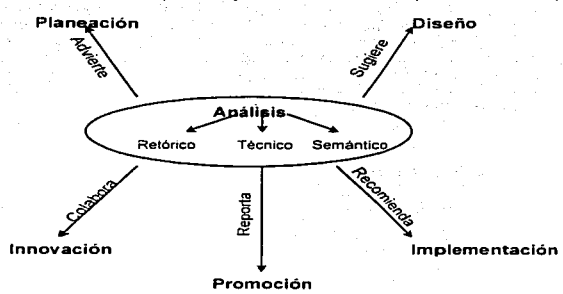



**Figura II.2.** Proceso de análisis.

Así, el analista actúa como revisor, evaluador y auditor en el proceso de desarrollo de la aplicación web. Los analistas deben poner especial atención en el grado en que la aplicación cumple con los siguientes principios:

**Un servicio continuo y global.** Debido a que la aplicación se encuentra en Internet, el analista deberá considerar la visita de una audiencia multinacional y multicultural las 24 horas del día (la mayoría de los servidores en la red global nunca se apagan).

**Ligas con significado y operación.** Las ligas empleadas en la aplicación deberán apoyar a un entendimiento más rápido y claro por parte del usuario que visita la aplicación. Se debe asegurar la operación y disponibilidad de las ligas empleadas en el mayor grado posible.

**Competitividad.** Es necesario que la aplicación se encuentre lo suficientemente actualizada para evitar ser superada por aplicaciones de la competencia con objetivos similares al nuestro.

## Procesos del análisis.

**Análisis de la Información.**

La información sobre la que se centra éste proceso no es únicamente la que se ofrece al usuario en la aplicación, sino que se refiere también a la información obtenida

externamente por el equipo de desarrollo: tipo de gente que usa la aplicación; uso más frecuente; opiniones sobre el sistema; frecuencia de uso, etc. Esta información aunque no es desplegada en el sistema, resulta importante para decidir si es necesario un cambio en el mismo y qué proceso deberá realizarlo. El análisis de esta información nos sirve para evaluar si la aplicación web cumple con lo siguiente: **¿Intenta alcanzar una audiencia que sí tiene acceso al Web?**; **¿Contribuye con nueva información a Internet?**; **¿Es consistente?** (es decir, que sus objetivos se apegan a las especificaciones existentes); **la información que ofrece ¿Es correcta?**; **¿Es accesada acorde a lo establecido en la planeación y diseño?**; **¿Satisface las necesidades de los usuarios?**.

Análisis del diseño de la aplicación.

**Desempeño.** Cuando un analista desea evaluar el desempeño de cierta aplicación web deberá contemplar lo siguiente: **tiempo de retardo** (se refiere al tiempo de espera del usuario para ver una página de nuestra aplicación –si ésta emplea recursos como audio ó video, el tiempo aumentará– y si dicha espera se justifica por la calidad de la información recibida); **legibilidad** (es necesario que los recursos empleados para "adornar" la aplicación –gráficos, texturas, ligas, etc.– no entorpezcan la legibilidad de la misma), y la **disponibilidad de uso** (que la información generada esté disponible para los paginadores y usuarios especificados al momento de necesitarse).

**Estética.** Este aspecto por ser muy subjetivo, es difícil de evaluar, sin embargo, pueden emplearse ciertos lineamientos como un diseño balanceado y coherente que ayude al usuario a enfocar su atención al contenido de la aplicación; que las páginas exhiban patrones y claves repetidos que demuestran uniformidad, consistencia y expresividad a lo largo de la aplicación; colores amigables, etc.

**Uso.** El analista deberá verificar la facilidad de uso de la aplicación, así como la utilidad de la información que ésta proporciona al visitante.

**Semántica.** En ocasiones existen señalizaciones dentro de la aplicación que no tienen razón de ser, y sólo propician una navegación "redundante" y confusa al usuario; además debe verificarse que los gráficos e iconos empleados no sean malinterpretados por usuarios de culturas y tradiciones diferentes.

Análisis de la Implementación.

Esta etapa hace referencia a la validación técnica del HTML en que está escrita la aplicación; se refiere a la forma en que son empleadas en el sistema las extensiones HTML (formas, frames, tablas, etc.) pero principalmente a las ligas tanto internas como externas de las que éste hace uso, con el propósito de ofrecer siempre acceso a documentos web existentes y actualizados, tanto dentro de nuestra aplicación como fuera de ella.

## II.4 Diseño.

El objetivo principal de un diseñador es crear lineamientos para la implementación de la aplicación: que tenga los "elementos idóneos" para que el usuario obtenga información al nivel deseado; que esté en un conjunto de páginas que lo guíen eficientemente hasta ella, etc. En el diseño en WWW, no existe una metodología específica a seguir, particularmente porque el proceso de diseño, al igual que los otros procesos en el desarrollo de aplicaciones en el Web, pueden continuar incluso después de que la aplicación haya sido liberada; sin embargo, el diseñador debe tener conocimiento de las diferentes metodologías y debe estar preparado para emplear cualquiera de ellas en el momento en que sea necesario.

### Principios del diseño en WWW.

Un diseñador deberá considerar los siguientes principios al momento de realizar su trabajo:

**Construir un significado asociativo.** Aprovechar el poder del hipertexto para marcar ligas a información relacionada; los diseños deberán contener ligas a información de contexto, así como al grueso de la misma (interna o externamente).

**Mantener la competitividad.** Es necesario que los diseñadores se aseguren de requerir de los usuarios el mínimo costo posible (tiempo de obtención de la información; esfuerzo requerido para obtener los datos deseados, etc.); uso eficiente de los recursos (tener mesura en el uso de gráficos, sonido, vídeo, etc.); emplear aspectos de la tecnología web que permitan una aplicación eficiente, elegante y fácil de mantener.

**Enfocarse a las necesidades del usuario.** La aplicación no debe crearse al gusto de los desarrolladores, sino debe ser acorde a las necesidades de la audiencia identificada en los procesos anteriores al diseño; el grado en el que la aplicación cumpla con lo anterior, será determinado por el proceso de análisis.

**Reconocer las múltiples entradas de un sistema web.** Como hemos visto anteriormente, el acceso a una aplicación web no está restringido a un solo punto, sino que mediante la dirección URL, un paginador puede accesar cualquier documento dentro de nuestro sistema; es necesario contar con los "señalamientos" indicados, para que cualquier usuario pueda navegar libremente por cualquier página (incluso accesar al sistema por cualquiera de ellas) y no perderse.

**Crear una apariencia consistente, agradable y eficiente.** El diseño de la aplicación deberá intentar dar al usuario la impresión de una organización común y coherente en todas sus páginas, con una presentación visual consistente.

**Soportar la interactividad con el usuario.** El usuario debe tener siempre la facilidad de contactar al grupo de desarrollo, para atender dudas o sugerencias; además,

debe contemplar el uso de formas y/o compuertas (CGIs) dependiendo del objetivo de la aplicación.

## Metodologías de Diseño.

No existe una verdadera metodología de diseño a seguir en el desarrollo de aplicaciones en WWW; sin embargo, un diseñador puede escoger entre una variedad de métodos diversos, dependiendo de la complejidad de la aplicación. Al desarrollar un simple sistema de páginas web, podremos emplear dos acercamientos:

- **Diseño top-dow.** Se emplea cuando se tiene una idea clara a priori del sistema, pues el diseñador comienza con una página inicial ó frontal llamada *página base* ó *home page*, a partir de la cual se desprenderán la totalidad de páginas que conforman el sistema.

- **Diseño bottom-up.** Es útil cuando el diseñador no conoce la apariencia final que deberá tener el sistema de páginas, pero conoce claramente cómo deben de verse y las funciones específicas que debe cumplir cada una de ellas; así, procederá a unir las páginas específicas en estratos superiores que las contengan, hasta llegar a conformar la *página base* del sistema.

Para aplicaciones más complejas, donde intervengan no sólo elementos de WWW (como en la interacción con otras aplicaciones mediante compuertas ó CGIs), la metodología de diseño deberá comprender también a la aplicación ajena al Web, por lo que ésta dependerá directamente del tipo de aplicación interactuante empleándose diagramas de estado, diagramas de flujo de datos, etc. Lo anterior lo podremos ver en el capítulo V de este trabajo de tesis.

## Técnicas de Diseño.

Estas técnicas ayudan al diseñador a considerar aspectos que son muy importantes para el manejo y buen despliegue de la información mostrada al usuario.

**Agrupamiento de la información en bloques de un tamaño adecuado.** El diseñador deberá agrupar moderadamente la información desplegada al usuario en cada una de las páginas que componen la aplicación: no deberá desplegarse mucha información en una sola página, en caso de ser necesario, debe hacerse uso de la capacidad *hipertexto* de WWW.

**Establecer una apariencia uniforme en las páginas que componen la aplicación.** Es importante que las páginas que componen nuestro sistema obedezcan a un formato o patrón uniforme que haga sentir la pertenencia de cada una de ellas a un sistema global, a pesar de hablar de temas completamente diferentes; además, ésta uniformidad es útil al usuario al facilitarle la localización de elementos de navegación propios de la aplicación.

**Creación de páginas índices.** En aplicaciones grandes, compuestas de gran cantidad de documentos, es imposible pedir al usuario que navegue a través de todos ellos para llegar a una página en especial: todo un laberinto; es por ésto que se necesitan páginas índice, es decir, documentos con ligas hipertexto que guíen al usuario a lugares específicos con la información de su interés.

## II.5 Implementación.

El reto para un implementador ó codificador es convertir un diseño web a una aplicación que trabaje; traduciendo todas las especificaciones arrojadas por los procesos anteriores en código HTML. En caso de que el sistema de páginas web interactúe con alguna otra aplicación externa mediante compuertas CGI, es en esta etapa cuando también se genera el código correspondiente, el cual a su vez obedece al conjunto de lineamientos establecidos por las etapas anteriores del proceso de desarrollo. Expongo ampliamente todos los detalles relacionados con CGI en el capítulo IV de este trabajo.

### Procesos de la Implementación.

**Trabajo en conjunto.** La mayor parte del tiempo el implementador se encontrará ocupado escribiendo código HTML; sin embargo, deberá dedicar parte de ese tiempo al trabajo con otras personas: planeadores, diseñadores, analistas, promotores, usuarios representativos, etc.; la finalidad es poder establecer vías de comunicación entre ellos que permitan un desarrollo más fácil y que se apegue más a los objetivos iniciales de la aplicación.

**Fase de pruebas.** Una vez que el implementador ha escrito el código de la aplicación, se tiene un *prototipo* de la misma, así que el siguiente paso es la realización de pruebas para ver si es que ya se puede "liberar" o es necesario corregirla (canalizándola al proceso de desarrollo correspondiente). Las pruebas a realizar deberán efectuarse no solamente por el implementador, sino por los diseñadores, planeadores, usuarios representativos, etc.

**Mantenimiento continuo.** El implementador deberá realizar un mantenimiento continuo que puede nunca terminar, pues el crecimiento y cambio son características inherentes de las buenas aplicaciones web. Un proceso de codificación, prueba,

actualización y comunicación continuos con otros desarrolladores y usuarios, es primordial.

## II.6 Promoción.

Una vez que se ha finalizado la construcción de la aplicación, es necesario darla a conocer al mundo entero, empleando para ello la misma red global o algún otro medio de comunicación de la organización, utilizando técnicas que permitan alcanzar objetivos como: informar al público en general de la existencia de la aplicación, así como su oferta; atraer el interés de la audiencia planeada y darles a conocer la manera en que el sistema satisface sus necesidades de información, etc. Un promotor deberá tener habilidades en relaciones públicas y comunicación interpersonal y masiva.

Los usuarios del Web han experimentado una verdadera "lluvia" de información", pues día a día aparecen nuevos `sitios web`, algunos de los cuales capturan y "monopolizan" la atención de los usuarios: no existe ninguna página a la cual acudir para enterarse de las nuevas aplicaciones en la red; sin embargo, existen estrategias útiles que el promotor podrá emplear para publicitar el sistema:

**Medios de comunicación existentes.** El promotor podrá emplear los medios de comunicación existentes de la organización (si los hay) para publicitar la locación de la aplicación en la red: folletos; anuncios en revistas, periódicos, televisión; etc. Sólo será necesario añadir a la publicidad cotidiana la línea correspondiente al URL de nuestra aplicación, proporcionando con ésto una posibilidad para ampliar la informaciór presentada en los medios convencionales.

**Servicios en línea.** Existen una gran variedad de sistemas web que permiten incluir en ellos referencias de aplicaciones en WWW, bastará con informar al servicio (comúnmente llamados `motores de búsqueda`) acerca del tema central de la aplicación, así como su dirección web correspondiente[3].

**Apoyo de otras aplicaciones.** En caso de que se tenga contacto con los administradores de otras aplicaciones web (preferentemente con objetivos afines), es posible solicitarles la colocación de alguna liga que haga referencia al sistema que deseamos promover; es más, en caso de que ya se cuente en la misma organización con una aplicación web, podría incluirse allí la liga hipertexto que conduzca al nuevo sistema.

**Patrocinio.** Todas las páginas en Internet generan gastos constantes para las personas que hacen posible su existencia (energía eléctrica, equipo de cómputo, infraestructura de telecomunicaciones, etc.); en muchas ocasiones estos gastos son

[3] Algunos ejemplo de motores de búsqueda los tenemos en **http://www.yahoo.com**, **http://www.lycos.com**, **http://www.infoseek.com**, etc.

absorbidos por empresas, cuyo único objetivo es que sus patrocinados incluyan ligas hipertexto a sus diferentes aplicaciones web.

**Anuncios clasificados.** Existen servicios en el Web que hacen las veces de anuncios clasificados; es decir, por medio de un pago establecido (en ocasiones es totalmente gratuito) éstas páginas hacen referencia a diversas aplicaciones (por ejemplo http://www.geocities.com).

## II.7 Innovación.

Diariamente el Web aloja más y más información; nuevos usuarios con necesidades de información únicas; oportunidades para servicios adicionales de todo tipo, etc. Así, la existencia de un proceso de innovación puede ayudar a los desarrolladores de una aplicación a extender y mejorar el servicio proporcionado a los usuarios de su sistema, así como impulsar la utilidad y consistencia del mismo.

La innovación es un proceso creativo y dinámico que no puede ser encapsulado en una serie de pasos a seguir; implica el constante monitoreo y conocimiento de las necesidades de los usuarios conforme éstas vayan apareciendo, así como el desarrollo de estructuras que nos permitan conocer dichas necesidades.

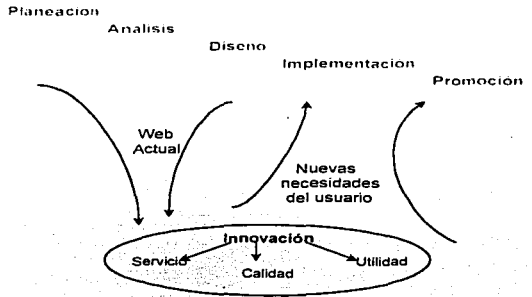



**Figura II.3.** La innovación trabaja conjuntamente con todas las demás etapas del proceso de desarrollo.

**Monitoreo de las necesidades de información.** La idea central de la innovación es dejar correr la aplicación de acuerdo a lo previsto; dentro del sistema deberán existir medios por los que el usuario podrá comunicarse con los administradores de la aplicación (a través de correo electrónico o formas de registro de opinión), gracias a los cuales los desarrolladores podrán identificar las nuevas necesidades de información de los usuarios, posibilitando así la modificación de la aplicación de acuerdo a las nuevas condiciones y necesidades de uso que deseen satisfacer.

**Vigilar la calidad de la información arrojada por la aplicación.**[4] Es necesario que la información proporcionada al usuario por la aplicación sea: **correcta** (que esté dentro del ámbito, propósito y contexto de su presentación), **accesible a cualquier usuario** (que las limitantes de despliegue de algunos paginadores no impidan al usuario conocer datos importantes de la aplicación), **útil, entendible** y **actualizada**.

**Considerar los avances tecnológicos pertinentes para su inclusión en la aplicación.** Los cambios tecnológicos frecuentemente ayudan a cubrir de manera más eficiente las necesidades del usuario, así como mantener vivo el interés y apariencia de una aplicación; sin embargo, las innovaciones tecnológicas nunca deben ser comparadas con el progreso. El mejoramiento de una aplicación puede ser alcanzado de una manera más satisfactoria mediante el rediseño de la misma, y no mediante la adopción de nuevas técnicas, cuyo uso implica riesgos económicos y funcionales para las empresas que las adoptan. Los innovadores deberán tener cuidado al recomendar la adopción de alguna tecnología de punta, pues su uso se justifica sólo tras poner en la balanza el nuevo grado de satisfacción obtenido por el usuario contra los riesgos y costos implicados.

[4] La calidad en la información de una aplicación web implica un proceso continuo de planeación, análisis, diseño, implementación, promoción e innovación que asegure que dicha información satisfaga las necesidades del usuario en términos del contenido e interfaz de la misma.

# 3

# Implementación: HTML

En el capitulo anterior se señaló que el desarrollo de aplicaciones en WWW se materializa al momento de la codificación del sistema (dentro del proceso de implementación), para ello es necesario el uso del lenguaje nativo del Web: HTML.

**HTML (Hypertext Markup Language**; Lenguaje de Marcación de Hipertexto) es un lenguaje empleado para "marcar" ó "señalar" las partes estructurales de un documento (párrafos, listas, encabezados, bloques, etc.); así, los **paginadores** (programas que "interpretan"[1] documentos HTML) identifican estas partes estructurales, desplegando su significado en una forma entendible al usuario.

HTML es un subconjunto de instrucciones de un estándar internacional llamado **SGML (Standard Generalized Mark-Up Language**; Estándar General de Lenguajes de Marcación)[2]. El objetivo principal que persigue SGML es **colaborar en el formateo de información en línea de manera que su distribución, búsqueda y recuperación electrónicas sean independientes de los detalles de la apariencia final del documento**. SGML se emplea para definir HTML en todos sus niveles[3].

Es importante resaltar que la intención de la codificación en HTML es enriquecer con elementos diversos, propios del lenguaje (ligas hipertexto, imágenes, sonidos, fondos, texturas, etc.) un **texto cualquiera**; por ésto, es necesario que sus programadores sigan reglas especificas de **marcación ó señalización** para las partes del documento a las que se desee aplicar dichos elementos. Los documentos HTML son creados e interpretados en archivos planos (texto ASCII normal), sin caracteres de

[1] Los paginadores cumplen la función de un intérprete, no de un compilador.
[2] SGML es un estándar ISO 8879 1986.
[3] Actualmente existen cuatro niveles HTML: 0, 1, 2 y 3. Los niveles 0 y 1 son un subconjunto de las instrucciones totales de HTML que **TODOS** los paginadores entienden (sean gráficos o basados en texto), sin embargo están limitados en poder de comunicación y despliegue. El nivel 2 comprende todas las instrucciones de los niveles 0 y 1, así como extensiones de algunas de ellas, y la inclusión de otras nuevas. El nivel 3 es el más reciente (aún en desarrollo al momento de escribir este trabajo, octubre 1996) y comprende, al igual que el nivel 2, todas las instrucciones de sus antecesores, ampliaciones de algunas de ellas, y la incorporación de nuevas capacidades. Los niveles HTML pueden verse como versiones del lenguaje: cada nuevo nivel contempla a su antecesor y añade características adicionales, basadas éstas en un avance tecnológico de hardware (poder de cómputo) y software (nuevas versiones de paginadores más poderosos). Al considerar los diferentes niveles de HTML, un desarrollador tiene la certidumbre de saber cuáles elementos de su aplicación podrán ser desplegados en cualquier paginador sin problema y cuáles no; sirviendo ésto a una buena implementación, que cumpla con las especificaciones establecidas en los diferentes procesos de desarrollo del sistema (véase Capítulo II).

control especiales ó códigos binarios anexados, así que el desarrollador puede fácilmente modificar el contenido de estos archivos mediante cualquier editor de texto.

## III.1 Descripción general de documentos HTML.

### Etiquetas.

El elemento básico de cualquier documento HTML es la **etiqueta** ó **tag**, la cual indica el tipo de comando ó **elemento** que habrá de emplearse; por ejemplo: <HTML>, <HEAD>, <BODY>, etc. Una etiqueta se compone de un **nombre de elemento**, rodeado por los caracteres < y > (paréntesis angulares). Las letras usadas para describir el elemento NO son sensitivas, es decir, es irrelevante si se escriben con mayúsculas o minúsculas, así <TITLE>, <TitLE> y <title> son la misma etiqueta.

Siempre que se emplea una etiqueta indicando el inicio de un elemento, por ejemplo <BODY>, deberá indicarse el final de dicho elemento mediante: </elemento>; donde elemento es el nombre utilizado como indicador de inicio; en nuestro ejemplo, una etiqueta de fin de elemento sería </BODY>.

En resumen, la sintaxis completa para una etiqueta es la siguiente:

<elemento> . . . </elemento>

donde:

| | |
|---|---|
| < elemento> | es la etiqueta de inicio del elemento; |
| </elemento> | es la etiqueta de fin del elemento; |
| elemento | es el nombre del elemento que se desea emplear, y |
| . . . | se refiere a un texto cualquiera, incluso a otras etiquetas que se deseen anidar, las cuales son afectadas también por la acción de elemento. (por ejemplo, un texto cualquiera con letra negrita e itálica). |

Sin embargo, la mayoría de los elementos de HTML poseen una serie de atributos que permiten variar al gusto del codificador el efecto que el elemento empleado tendrá al momento del despliegue en el paginador; por lo que la sintaxis final de una etiqueta HTML (si el elemento cuenta con atributos) es:

<elemento [atributo1="valor1"] [atributoN="valor2"]> . . . </elemento>

donde:

| | |
|---|---|
| elemento | es el nombre del elemento que se desea emplear; |
| atributo1, atributoN | son los nombres de los atributos particulares de elemento. No todos los elementos cuentan con atributos; |
| valor1,valor2 | son los valores correspondientes a cada atributo particular de elemento, y |

. . . se refiere a un texto cualquiera, incluso a otras etiquetas que se deseen anidar, todas ellas afectados por la acción de elemento.

Un ejemplo de lo anterior lo podemos apreciar en:

```
<A Name="Ejemplo HTML" Href="http://indy.aragon.unam.mx/imagen.html">
                    . . . </A>
```

podemos ver que el elemento empleado es **A**, los dos atributos utilizados son **Href** y **Name**, cuyos valores son **http://indy.aragon.unam.mx/imagen.html** y **Ejemplo HTML** respectivamente, entrecomillados y precedidos de un signo = (igual); a su vez, es posible distinguir que se empleó una etiqueta de inicio y una etiqueta de fin. Los tres puntos representan un texto cualquiera, que es afectado por la acción del elemento **A**.

## Comentarios.

Cuando se codifica cualquier programa siempre es importante **documentar** el código del mismo, para lo que se usan comentarios (texto que no es tomado en cuenta por el intérprete); HTML no es la excepción, la indicación de un comentario es:

```
<!-- Este es un comentario que el interprete ignorará -->
```

## Estructura básica de un documento HTML.

Un documento HTML se divide básicamente en dos partes: una **sección de cabecera** (limitada por las etiquetas <HEAD> y </HEAD>) y un **cuerpo** (limitado por las etiquetas <BODY> y </BODY>), ambas limitadas por las etiquetas <HTML> y </HTML>. Lo anterior puede observarse en el siguiente ejemplo:

```
<!-- Los elementos HTML,HEAD y BODY se incluyen siempre -->
<!-- en cualquier documento HTML, en el orden en que aquí se muestra -->
<HTML>
<HEAD>
     <TITLE>Este es un ejemplo de HTML</TITLE>
</HEAD>
<BODY>
     <!-- Dentro de las etiquetas <BODY> y </BODY> se escribe el    -->
     <!-- grueso de instrucciones que conforman el documento HTML -->
     <P>
     Este es un ejemplo de documento HTML.
     </P>
</BODY>
</HTML>
```

El desplegado que el código anterior produciría es el siguiente (empleando el paginador Netscape Navigator versión 2.02):

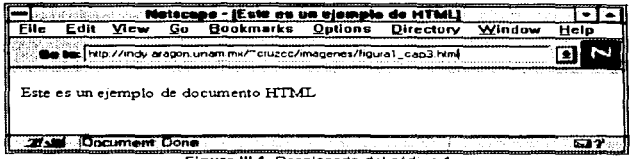



**Figura III.1.** Desplegado del código 1.

## III.2 Elementos HTML.

El desarrollo de documentos HTML no es nada complicado, pues como ya se expuso, basta con flanquear con las etiquetas de inicio y fin correspondientes a un texto cualquiera. La verdadera dificultad es conocer la gran cantidad de elementos HTML existentes, por lo que para facilitar esta tarea presento a continuación un cuadro resumen de la gran mayoría de ellos.

En los siguientes cuadros se emplea la notación:

| | |
|---|---|
| *cadena* | Cualquier cadena de caracteres. |
| *URL* | Un URL especifico. |
| . . . | Texto y/o una serie de elementos. |
| *valor1* \| . . . | Una serie de valores posibles para los atributos. |
| *N* | Un número cualquiera. |

### Elementos HTML nivel 0.

| Elementos de la sección de cabecera | | |
|---|---|---|
| **Etiqueta (inicio ... fin)** | **Atributos** | **Explicación** |
| <HEAD> . . . </HEAD> | - | Etiquetas que indican el inicio y fin de la sección de cabecera, las definiciones aquí escritas afectan a todo el documento HTML. |
| <TITLE> . . . </TITLE> | - | Rodean una cadena cualquiera que identifica el contenido del documento. Todo documento HTML debe tener un elemento TITLE. |
| <BASE> . . . </BASE> | | Empleadas para grabar el URL de la versión original de un documento; útil cuando los archivos fuente son transportados a un lugar fuera de su contexto. |
| | Href = "*URL*" | Define el URL base del documento. |
| <ISINDEX> | - | Marca el documento como uno índice; el servidor en el que se encuentre el documento debe tener un motor de búsqueda definido que soporte dicha operación. Esta etiqueta no necesita indicación de fin de elemento. |

**Tabla III.1.** Elementos de la cabecera, HTML nivel 0.

| Elementos del cuerpo | | |
|---|---|---|
| **Etiqueta (inicio ... fin)** | **Atributos** | **Explicación** |
| <BODY> . . . </BODY> | - | Etiquetas que indican inicio y fin de la sección del cuerpo; limitan el contenido de un documento HTML. |
| <A> . . . </A> | | Elemento "ancla", base para el ligado hipertexto de varios documentos. |
| | Href="*URL*" | Identifica el URL de la referencia hipertexto en la forma Href="*URL*", donde *URL* es el nuevo recurso que el paginador consultará cuando el usuario seleccione esta "liga". |
| | Name="*cadena*" | Crea un nombre para el elemento A; este nombre podrá ser utilizado dentro del documento o fuera de él por otro elemento A, referenciando la porción de texto identificada por *cadena*. |
| | Title="*cadena*" | Este atributo asigna un título al documento referenciado por Href; el paginador asignará este título al documento referenciado al momento en que éste sea consultado. |
| **Bloques de Caracteres: Elementos que "agrupan" texto en listas o bloques** | | |
| **Etiqueta (inicio ... fin)** | **Atributos** | **Explicación** |
| <PRE> . . . </PRE> | - | Indica un bloque de texto formateado, es decir, donde se respetan los saltos de líneas, espacios y tabuladores, con un tipo de letra de ancho fijo (como de maquina de escribir) |
| <UL> . . . </UL><br><OL> . . . </OL> | - | Empleadas para elaborar listas de información; AMBAS emplean el elemento LI para identificar cada uno de los componentes de la lista:<br>• UL indica una lista no numerada.<br>• OL indica una lista numerada. |
| <LI> | - | Identifica un elemento de una lista en OL ó UL. No necesita etiqueta de fin de elemento. |
| <DL> . . . </DL> | - | Identifica una lista de definición o glosario; emplea DT para señalar términos y DD para definiciones. |
| <DT> | - | Identifica un termino en una lista de definición. |
| <DD> | - | Indica una descripción en una lista de definición. |
| **Encabezados** | | |
| **Etiqueta (inicio ... fin)** | **Atributos** | **Explicación** |
| <H1> . . . </H1> | - | Estas etiquetas dan un formato predeterminado al texto que se encuentre entre ellas, variándolo en tamaño y propiedades; el encabezado uno es el más grande y el seis el más pequeño. |
| <H2> . . . </H2> | - | Encabezado tipo 2. |
| <H3> . . . </H3> | - | Encabezado tipo 3. |
| <H4> . . . </H4> | - | Encabezado tipo 4. |
| <H5> . . . </H5> | - | Encabezado tipo 5. |
| <H6> . . . </H6> | - | Encabezado tipo 6. |
| **Separadores** | | |
| **Etiqueta (inicio ... fin)** | **Atributos** | **Explicación** |
| <HR> . . . </HR> | | Divide el texto en secciones con una línea horizontal. No necesita etiqueta de fin de elemento. |
| <P> . . . </P> | - | Identifica el inicio de un párrafo; la etiqueta de fin de elemento </P> es opcional. |

**Tabla III.1.** Elementos del cuerpo, HTML nivel 0.

| Elementos del cuerpo (continuación) | | |
|---|---|---|
| **Espaciadores** | | |
| **Etiqueta (inicio ... fin)** | **Atributos** | **Explicación** |
| <BR> | - | Forza un salto de línea en el desplegado hecho por el paginador. No requiere etiqueta de fin de elemento |
| **Imágenes** | | |
| **Etiqueta (inicio ... fin)** | **Atributos** | **Explicación** |
| <IMG> . . . </IMG> | | Posiciona una imagen gráfica en un documento HTML en la posición en que la etiqueta <IMG> aparezca. |
| | Src="*URL*" | Identifica el archivo fuente de la imagen (a través de su URL) |
| | Alt="*cadena*" | *cadena* aparecerá si es que el paginador empleado no tiene capacidad de desplegado de imágenes; es decir, en lugar de aparecer el gráfico, aparecerá la cadena. |
| | Align="TOP \| MIDDLE \| BOTTOM" | Indica la relación de posición entre el gráfico y el texto que se encuentre entre las etiquetas de inicio y fin de elemento; los valores incluyen:<br>• TOP: El texto en seguida del gráfico se alineará a la altura del borde superior del gráfico.<br>• MIDDLE: El texto se alineará a la altura media del gráfico.<br>• BOTTOM: El texto se alineará a la altura del borde inferior del gráfico. |
| | Ismap | Identifica una imagen como un "mapa sensitivo", en donde las regiones del gráfico son "mapeadas" a URLs predefinidos. La elaboración de un mapa sensitivo comprende más cosas que sólo HTML, como conocimientos de CGIs y administración de servidores WWW. |

**Tabla III.2.** Elementos del cuerpo, HTML nivel 0 (continuación)

**Ejemplo.**

A continuación muestro un ejemplo de algunas etiquetas HTML nivel 0, seguido de la imágen que sería desplegada en el paginador Netscape Navigator versión 2.02.

```
<!-- Este es un documento ejemplo de elementos HTML nivel 0 -->
<!-- Autor:  Cesar Cruz Cortes -->
<!-- Fecha última actualización: 20-Octubre-1996 -->
<HTML>
<HEAD>
        <TITLE>Ejemplo de Elementos HTML nivel 0</TITLE>
        <BASE Href="http://indy.aragon.unam.mx/tesis/ejemplo_nivel0.html">
</BASE>
</HEAD>
<BODY>
        <A Href="ejemplo_nivel1.html" Name="NIVEL1"> Esta es una liga al siguiente
        ejemplo (esta dentro de mi servidor) </A><BR>
        <A Href="http://serpiente.dgsca.unam.mx" Name="UNAM"> Esta liga me lleva
        directamente a la pagina web principal de la UNAM</A><BR>
```

```
        <!-- La siguiente etiqueta dibuja línea horizontal en la pantalla (sin atributos) -->
        <HR>
        <PRE> Este es  un   ejemplo                    de texto
              formateado.  Respeta  e s p a c i o s    y
                    tabula-          dores</PRE>
        <HR>
        <OL>La siguiente es una lista numerada:
              <LI>Elemento Uno.
              <LI>Elemento Dos.
        </OL><P>
        <UL>La siguiente es una lista no numerada.
              <LI>Elemento A.
              <LI>Elemento B.
        </UL><P><HR><P>
        <H1>Encabezado Tipo 1</H1>
        <H3>Encabezado Tipo 3</H3>
        <H6>Encabezado Tipo 6</H6><HR>
</BODY>
</HTML>
```

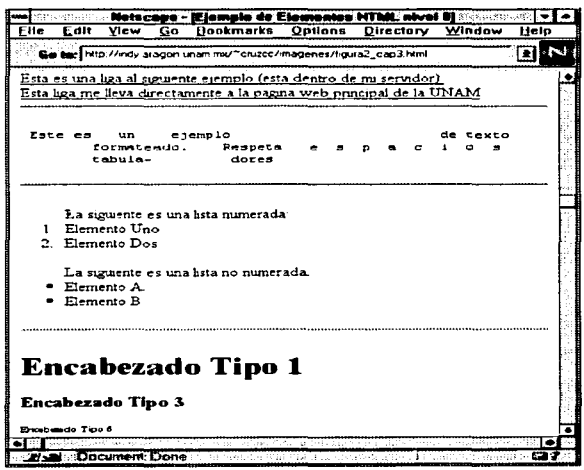



**Figura III.2.** Despliegado de elementos HTML nivel 0.

## Elementos HTML nivel 1.

HTML nivel 1 contempla todos los elementos del nivel 0 e incluye algunos otros, pero todos de la sección del cuerpo, ninguno de la cabecera.

| Elementos del cuerpo | | |
|---|---|---|
| **Etiqueta (inicio ... fin)** | **Atributos** | **Explicación** |
| <CITE> . . . </CITE> | - | Delimita una cita. |
| <CODE> . . . </CODE> | - | Delimita el código de un programa de computadora. Al incluir el código, este no se confunde con las etiquetas propias de HTML. |
| <EM> . . . </EM> | - | Delimita texto enfatizado. |
| <STRONG> . . . <STRONG> <B> . . . </B> | - | Delimita texto enfatizado con negrita. |
| <TT> cadena </TT> | - | Delimita texto tipo maquina de escribir (de longitud fija). |
| <I> . . . </I> | - | Delimita texto en letra itálica. |

**Tabla III.3.** Elementos del cuerpo, HTML nivel 1.

**Ejemplo.**

A continuación muestro un documento HTML que emplea los elementos del nivel 1, seguido de su desplegado empleando Netscape Navigator versión 2.02.

```
<!-- Este es un documento ejemplo de elementos HTML nivel 1 -->
<!-- Autor: Cesar Cruz Cortes -->
<!-- Fecha última actualización: 20-Octubre-1996 -->
<HTML>
<HEAD>
        <TITLE>Este es un ejemplo de HTML Nivel 1</TITLE>
</HEAD>
<BODY>
        <HR>
        <CITE>Este texto indica una cita.</CITE><BR>
        <EM>Este texto esta enfatizado</EM><BR>
        <STRONG>Este texto esta escrito en negritas</STRONG>
        <B>... y este también </B><BR>
        <I>Este texto está escrito en itálicas</I><BR>
        <TT>Estas letras parecen de maquina de escribir</TT><HR>
        <CODE>En las siguientes líneas pueden incluirse códigos de cualquier lenguaje,
                es decir, se    aceptan caracteres especiales como !"·$%&/()=?¿|\# *
                entre   otros.
        </CODE><HR>
</BODY>
</HTML>
```

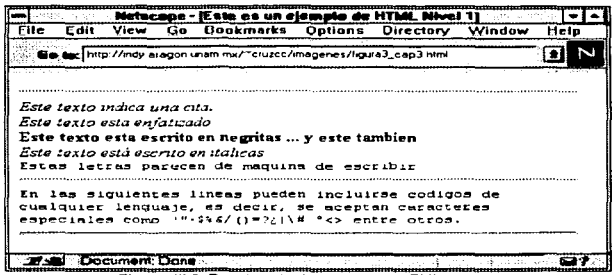



**Figura III.3.** Despliegado de elementos HTML nivel 1

## Elementos HTML nivel 2.

El nivel 2 de HTML proporciona únicamente una estructura que se conoce como **forma** (del tipo de lás formas de registro con las que lidiamos cotidianamente); ésta estructura se integra a partir de varios elementos con sus correspondientes atributos, los cuales expongo a continuación:

| Elementos de la forma | | |
|---|---|---|
| **Etiqueta (inicio ... fin)** | **Atributos** | **Explicación** |
| <FORM> . . . </FORM> | | Delimita el contenido de una forma |
| | Action="*URL*" | Identifica el URL del programa o script que acepta el contenido de una forma para ser procesado (CGIs). Si este atributo está ausente, el URL BASE de la forma es empleado (véase etiqueta <BASE> en HTML nivel 0) |
| | Method="GET \| POST" | Indica la variación en la forma de recibir la información que utilizará el programa ó script indicado en la etiqueta Action |
| <INPUT> . . . </INPUT> *(continúa . . . )* | | Empleada para capturar información proveniente del usuario. </INPUT> es opcional. |
| | Align="TOP \| MIDDLE \| BOTTOM" | Empleada sólo con Type="IMAGE" (véase más adelante), y define la relación de la imagen involucrada con el texto que le sigue. |
| | Checked | Ocasiona que el estado inicial de una `caja de verificación (checkbox)` ó un `botón de radio (radiobutton)` tengan un estado de seleccionado. Si este atributo no se indica, el estado inicial es NO seleccionado. |

**Tabla III.4.** Elementos de la forma, HTML nivel 2

| **Elementos de la forma (continuación)** | | |
|---|---|---|
| **Etiqueta (inicio ... fin)** | **Atributos** | **Explicación** |
| <INPUT> . . . </INPUT><br>*( . . .continuación)* | Maxlength="*N*" | Indica un número máximo de caracteres (*N*) que el usuario puede introducir en un campo de texto. El valor por default es "ilimitado". |
| | Name="*cadena*" | Identifica un nombre simbólico que es empleado en la transmisión e identificación de salida de este elemento dentro de la forma. |
| | Size="*N*" | Especifica el ancho del campo de texto que será desplegado al usuario. |
| | Src="URL" | Identifica el archivo fuente de la imagen empleada cuando el atributo Type tiene el valor de "IMAGE". |
| | Type="CHECKBOX \| HIDDEN \| IMAGE \| PASSWORD \| RADIO \| RESET \| SUBMIT \| TEXT" | Identifica el tipo del campo de entrada de datos:<br>• CHECKBOX es empleado para recolectar datos que pueden tener múltiples valores al mismo tiempo.<br>• HIDDEN se emplea para valores establecidos en la forma por el programador y no por el usuario (variables).<br>• IMAGE es un campo de imágen empleado para transmitir la forma cuando el usuario hace `click` sobre la imágen, la forma es transmitida al URL especificado en Action, y las coordenadas `x` y `y` al momento del click, son transmitidas también como parte de la información recolectada por la forma.<br>• PASSWORD es un campo en el que el usuario escribe texto, pero el texto no es desplegado en la pantalla (dependiendo del paginador, podría aparecer como "******").<br>• RADIO es empleado para colectar información donde sólo hay un posible valor a escoger de un conjunto de alternativas. El atributo "checked" (arriba) puede indicar el valor inicial de este elemento.<br>• RESET dibuja un botón en la pantalla para limpiar la forma y regresarla a sus valores por default. El atributo "value" (abajo) determina la cadena que será desplegada en el botón citado.<br>• SUBMIT dibuja un botón en la pantalla para transmitir la forma al URL determinado. El atributo "value" (abajo) determina la cadena que será desplegada en el botón.<br>• TEXT se emplea para escribir una simple línea de texto; emplea los atributos "size" y "maxlength" (arriba). Para la entrada de múltiples líneas, use "textarea" (abajo). |
| | Text | Identifica la entrada de datos como una línea de texto. |
| | Value="STRING" | Indica el valor inicial desplegado en un campo, o bien, el valor de un campo cuando éste es seleccionado (Type = "RADIO" debe usar este atributo). |

**Tabla III.4.** Elementos de la forma, HTML nivel 2 (continuación).

| Elementos de la forma (continuación) | | |
|---|---|---|
| **Etiqueta (inicio ... fin)** | **Atributos** | **Explicación** |
| <SELECT> . . . </SELECT> | | Se emplea para presentar al usuario una serie de alternativas a elegir. El elemento OPTION (abajo) se emplea para definir cada alternativa |
| | Name="*cadena*" | Identifica el nombre lógico que será transmitido y asociado con el dato resultado de la selección hecha por el usuario |
| | Multiple | Por default, el usuario puede realizar sólo una selección de un grupo en un elemento SELECT. Si se emplea el atributo Multiple, el usuario podrá seleccionar una ó más opciones |
| | Size="*N*" | Especifica el número de elementos a seleccionar visibles. Si *N* es mayor a uno, el desplegado visual será una lista |
| <OPTION> . . . </OPTION> | | Ocurre solo dentro del elemento SELECT y se emplea para representar cada opción de SELECT |
| | Selected | Indica que esta opción será seleccionada inicialmente |
| | Value="*N*" | Si se presenta, *N* será regresado por el elemento SELECT si ésta opción es elegida; de otra manera, el valor regresado es aquel especificado por el elemento OPTION |
| <TEXTAREA> . . . </TEXTAREA> | | Permite la escritura de múltiples líneas de texto por parte del usuario. El usuario dispone de un área limitada para entrar dicho texto |
| | Name="*cadena*" | Identifica el nombre lógico que será asociado al texto introducido por el usuario |
| | Rows="*N*" | *N* es el número de renglones que tendrá el área de que dispone el usuario para ingresar el texto |
| | Cols="*N*" | *N* es el número de columnas que tendrá el área de que dispone el usuario para ingresar el texto |

**Tabla III.4.** Elementos de la forma, HTML nivel 2 (continuación).

**Ejemplo.**

A continuación aparece el código de un documento HTML utilizando formas, en seguida el desplegado del documento, empleando el paginador Netscape Navigator versión 2.02.

```
<!-- Este es un documento ejemplo de elementos HTML nivel 2 (Formas) -->
<!-- Autor:  Cesar Cruz Cortes -->
<!-- Fecha última actualización: 20-Octubre-1996 -->
<HTML>
<HEAD>
   <TITLE>Ejemplo de Elementos HTML Nivel 2 -FORMAS-</TITLE>
   <BASE Href="http://indy.aragon.unam.mx/tesis/ejemplo_nivel2.html"> </BASE>
</HEAD>
<BODY>
   <H3>Este es un ejemplo de forma</H3><HR>
   <FORM Method="POST"
      Action="http://indy.aragon.unam.mx/cgi-bin/cruzcc/forma1.c>
```

```
    <P>

<INPUT Type="HIDDEN" Name="Direccion" Value="cruzcc@indy.aragon.unam.mx">
<INPUT Type="HIDDEN" Name="Motivo" Value="Respuesta de Forma HTML nivel 2">

Su edad: <INPUT Type="Text" Name="Edad" Size="2"><BR><P>

Su Sexo:
<INPUT Type="Radio" Name="Sexo" Value="M">Masculino
<INPUT Type="Radio" Name="Sexo" Value="F">Femenino<BR><P>

Elija el nombre de las personas de quien ud. ha oido o conocido
personalmente de entre las siguientes:<BR>

<INPUT Type="Checkbox" Name="Conocidos" Value="CSG">Carlos Salinas
    de Gortari<BR>
<INPUT Type="Checkbox" Name="Conocidos" Value="EMR">Enrique
    Muñoz Rocha<BR>
<INPUT Type="Checkbox" Name="Conocidos" Value="LDC">Luis
    Donaldo Colosio<BR>
<INPUT Type="Checkbox" Name="Conocidos" Value="Marcos">El Subcomandante
    Marcos<BR><P>

¿Cual es su paginador favorito?
    <SELECT Name="Paginador">
        <OPTION> Arena </OPTION>
        <OPTION> Cello </OPTION>
        <OPTION> Chimera </OPTION>
        <OPTION> Lynx </OPTION>
        <OPTION> MacWeb </OPTION>
        <OPTION> Mosaic </OPTION>
        <OPTION Selected> Netscape </OPTION>
        <OPTION> SlipKnot </OPTION>
        <OPTION> Viola </OPTION>
        <OPTION> Web Explorer </OPTION>
        <OPTION> Ninguno de ellos </OPTION>
    </SELECT><BR><P>

Adivina el password secreto:
<INPUT Type="PASSWORD" Name="Password"><BR><P>

Entre sus opiniones sobre la presente forma: <BR>
<TEXTAREA Name="Opinion" Rows="3" Cols="40">
Mi opinión es:
</TEXTAREA><BR><P>

<INPUT Type="SUBMIT" Value="Mandar la forma">
<INPUT Type="RESET" Value="Cancelar el envio">

</FORM>

<P>¡Gracias!<HR>
</BODY>
</HTML>
```

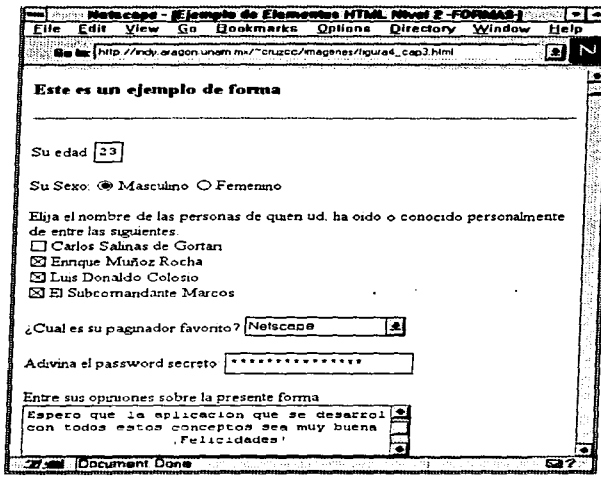



**Figura III.4.** Desplegado de elementos HTML nivel 2.

## Elementos HTML nivel 3.

Al igual que los niveles pasados, el nivel 3 considera todos los elementos especificados anteriormente, y añade extensiones a elementos ya existentes (todos ellos correspondientes a la sección del cuerpo del documento), además de una estructura especial llamada tabla.

| Elementos del cuerpo (Extensiones) | | |
|---|---|---|
| **Etiqueta (inicio ... fin)** | **Atributos** | **Explicación** |
| <BODY> . . . <BODY> | | Delimita el contenido de la sección del cuerpo del documento. |
| | Background="*URL*" | Identifica el URL correspondiente a un gráfico que hará las veces de "fondo" de la página. |
| | Bcolor="#RRGGBB" | Especifica un color sólido que hará las veces de fondo de la página. El color se especifica empleando un número `hexadecimal` (de 6 cifras), donde #000000 es el negro, y #FFFFFF es el blanco. La relación de colores puede verse en la tabla III.7. |
| | Text="#RRGGBB" | Especifica el color para el texto del documento; también es un número hexadecimal. |
| | Link="RRGGBB" | Especifica el color que tendrán las ligas hipertexto del documento. RRGGBB es un número hexadecimal. |
| | VLink="RRGGBB" | Especifica el color que tendrán las ligas hipertexto que ya hayan sido visitadas. RRGGBB es un número hexadecimal. |
| | Alink="RRGGBB" | Especifica el color que tendrán las ligas hipertexto activas. RRGGBB es un número hexadecimal. |
| <HR> . . . </HR> | | Divide el texto en secciones con una línea horizontal. No necesita etiqueta de fin de elemento. |
| | Size=*N* | Indica la altura de la línea horizontal en `pixeles`. |
| | Width=*N*\|*N*% | Indica el ancho de la línea horizontal en `pixeles` (con *N*) ó en porcentaje de la ventana (con *N*%). |
| | Align="LEFT \| RIGHT \| CENTER" | Indica la forma en la que se alinea el separador en caso de no ocupar todo el ancho de la pantalla de despliegue, a la izquierda (LEFT), derecha (RIGHT) o centrada (CENTER). |
| <BASEFONT> . . . </BASEFONT> | | Cambia el tamaño de la letra del documento. |
| | Size="*N*" | Especifica el tamaño de la letra del documento en un rango de uno (la más pequeña) a siete (la más grande): 1 . . 7. |
| <BLINK> *cadena* </BLINK> | | El texto de *cadena* es desplegado por el paginador de manera intermitente. |
| <CENTER> *cadena* </CENTER> | | Centra el texto de *cadena* en el renglón actual. |

**Tabla III.5.** Elementos del cuerpo (extensiones), HTML nivel 3.

| Elemento tabla | | |
|---|---|---|
| **Etiqueta (inicio ... fin)** | **Atributos** | **Explicación** |
| <TABLE> . . . </TABLE> | | Delimita el contenido de una tabla. |
| | Align=" BLEEDLEFT \| LEFT \| CENTER \| RIGHT \| BLEEDRIGHT \| JUSTIFY | El alineamiento horizontal de la tabla en la pantalla (no el contenido de la tabla) puede tomar uno de los siguientes valores:<br>• BLEEDLEFT alinea con respecto al borde izquierdo de la ventana.<br>• LEFT alinea con respecto al márgen izquierdo del texto.<br>• CENTER centra entre dos márgenes.<br>• BLEEDRIGHT alinea con respecto al borde derecho de la ventana.<br>• RIGHT alinea con respecto al márgen derecho del texto.<br>• JUSTIFY El elemento TABLE llena los espacios entre márgenes de texto. |
| | Border | Indica al paginador trazar un borde alrededor de la tabla; si este atributo no aparece, la tabla no tendrá ninguna división entre sus celdas. |
| | Width="*N*" | Especifica el ancho de la tabla, si es dado como "*N*%", la tabla es *N*% del ancho de despliegue del paginador. |
| | Colspec="*LN RN CN*" | Especifica la alineación de elementos en las columnas. Por ejemplo, Colspec="LN RN CN", dice al paginador que la columna 1 será alineada a la izquierda (left), la columna 2 a la derecha (right), y la columna 3 al centro (center). *N* especifica el ancho de la columna en unidades. |
| <CAPTION> *cadena* </CAPTION> | | Empleada para "etiquetar" (dar nombre) a una tabla o figura. |
| | Align=" TOP \| BOTTOM \| LEFT \| RIGHT" | Identifica la posición de la etiqueta con respecto a la tabla o figura. |
| <TR> *cadena* </TR><br><TH> *cadena* </TH><br><TD> *cadena* </TD> | | <TH> es empleada para etiquetar el encabezado (título) en la tabla. <TD> se emplea para etiquetar los datos en la tabla y <TR> identifica el contenido de un renglón completo de la tabla. |
| | Align="LEFT \| CENTER \| RIGHT \| JUSTIFY \| DECIMAL" | Identifica la alineación horizontal de los elementos en una línea de la tabla. |
| | Valign=" TOP \| MIDDLE \| BOTTOM \| BASELINE" | Identifica la alineación vertical de los elementos en una celda de la tabla. |
| | Colspan="*N*" | Identifica el número de columnas que tendrá de largo la presente celda. |
| | Rowspan="*N*" | Identifica el número de renglones que tendrá de alto la presente celda. |
| | Nowrap | Previene al paginador de no encimar los contenidos de las celdas. |

**Tabla III.6.** Elementos de la entidad TABLA, HTML nivel 3.

**Ejemplo.**

A continuación aparece un documento HTML ejemplo, donde se muestra el uso de las tablas y las extensiones que ofrece el nivel 3. En esta ocasión, el acostumbrado desplegado del código en el paginador Netscape Navigator versión 2.02, podrá apreciarse en la figura III.5.

```
<!-- Este es un documento ejemplo de elementos HTML nivel 3 -->
<!-- Autor: Cesar Cruz Cortes -->
<!-- Fecha última actualización: 20-Octubre-1996 -->
<HTML>
<HEAD>
   <TITLE>Ejemplo de Elementos HTML Nivel 3</TITLE>
   <BASE Href="http://indy.aragon.unam.mx/tesis/ejemplo_nivel3.html"> </BASE>
</HEAD>
<BODY Background="http://indy.aragon.unam.mx/tesis/unam.gif"
      Link="000000" Vlink="FFFF00">
   <H3>Este es un ejemplo sencillo de una tabla</H3>
   <HR Size=4 Width=75%><P>
   <CENTER><TABLE BORDER>
      <CAPTION>Tabla Ejemplo</CAPTION>
      <TR><TH Colspan="3">Eliminatoria Mundialista Francia '98</TH></TR>
      <TR><TH>Grupo 1</TH><TH>Grupo 2</TH><TH>Grupo 3</TH></TR>
      <TR><TD>México</TD><TD>E.U.A.</TD><TD>Canada</TD></TR>
      <TR><TD>Honduras</TD><TD>El Salvador</TD><TD>Guatemala</TD></TR>
      <TR><TD>Jamaica</TD><TD>Cuba</TD><TD>Costa Rica</TD></TR>
      <TR><TD>San Vicente</TD><TD>Belice</TD><TD>Panama</TD></TR>
   </TABLE></CENTER><P>
   <A HREF="http://www.diter.com.mx">Esta es una liga que no he visitado</A><BR>
   <A Href="http://www.yahoo.com">Esta es una liga que ya visité</A>
</BODY>
</HTML>
```

**Definición de colores**

En la tabla III.5 es posible apreciar que al momento de la especificación de colores ya sea de fondo, ligas, etc., se cita el uso de un número `hexadecimal` de la forma **#RRGGBB**. La siglas provienen del inglés **R**ed (rojo), **G**reen (verde) y **B**lue (azul). La cifra hexadecimal se refiere a la proporción de rojo, verde y azul que componen al color que se desea emplear en el elemento correspondiente. Algunos de los colores más comunes se muestran en la tabla III.7.

| Color | # Hexadecimal | Color | #Hexadecimal |
|---|---|---|---|
| Negro | #000000 | Magenta | #FF00FF |
| Azul | #0000FF | Amarillo | #FFFF00 |
| Verde | #00FF00 | Violeta | #EE82EE |
| Rojo | #FF0000 | Beige | #F5F5DC |
| Blanco | #FFFFFF | Naranja | #FFA500 |
| Cyan | #00FFFF | Oro | #FFD700 |
| Verde Limóm | #32CD32 | Café | #A52A2A |
| Café Intenso | #D2691E | Gris | #D02090 |
| Gris Claro | #D3D3D3 | Khaki | #F0E68C |

**Tabla III.7.** Códigos hexadecimales de los colores más comunes.

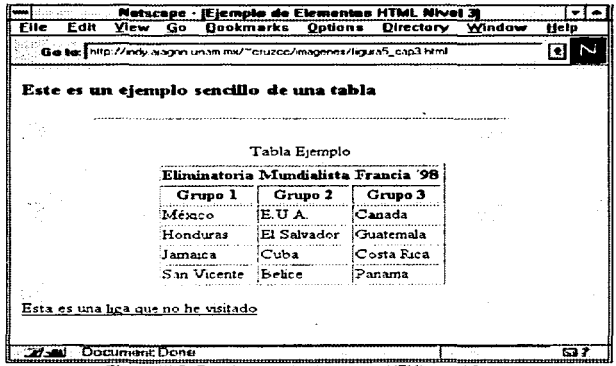



**Figura III.5.** Despliegado de elementos HTML nivel 3.

HTML es muy sencillo, proporciona todas las herramientas que se pudieran necesitar para el desarrollo de la interfaz con el usuario (característica inherente de un cliente en una aplicación cliente-servidor); sin embargo, existen elementos del lenguaje que necesitan el apoyo de agentes externos llamados CGIs ó compuertas (como en las formas, los mapas sensitivos, etc.), éstas amplian las capacidades de HTML al permitirle interactuar con aplicaciones externas, como Bases de Datos, servidores de archivos, lenguajes de programación externos, etc. La programación de compuertas ó CGIs, son el tema central del siguiente capítulo.

# 4

# Programación de CGI

La interfase común basada en compuertas (ó **CGI**; **Common Gateway Interface** por su traducción del inglés) amplia enormemente las capacidades del lenguaje HTML (como en las formas, los mapas sensitivos y algunos otros elementos expuestos en el capítulo anterior); sin embargo, su verdadera importancia radica en que los CGIs constituyen la manera en que un servidor HTTP puede "comunicarse" con programas de cualquier tipo en cualquier computadora local o remota. Su nombre describe hábilmente las funciones que cumple dentro de la tecnología WWW:

**Interfase.** Sus mecanismos estándares proveen un ambiente completo para los desarrolladores; no hay necesidad de aprender los minuciosos detalles del código fuente del servidor web; una vez que se ha comprendido la interfase cliente-servidor, es posible comenzar con la programación de CGIs, basta con conocer la manera en que fluye la información desde y hacia la compuerta programada.

**Común.** La idea central es que cada servidor y programa cliente, sin importar la plataforma de cómputo ó sistema operativo empleado, se apeguen a los mismos mecanismos estándares del flujo de datos entre cliente, servidor y compuerta. Esto permite un alto nivel de `portabilidad` entre una amplia variedad de computadoras y sistemas operativos.

**Compuerta.** A pesar de que un CGI puede ser un programa independiente (es decir, que su ejecución individual produciría una salida no dependiente de la tecnología WWW), en el Web actúa como un **mediador** entre el servidor HTTP y cualquier otro programa o aplicación que acepte instrucciones en forma de línea de comandos en tiempo real. Un ejemplo de esta "mediación" podemos observarla en una aplicación de *bases de datos SQL* que no posea ningún medio de comunicación con un servidor HTTP y desee integrar su información a WWW; para ello es necesario el uso de un "**programa compuerta**", un **CGI**.

Los CGIs operando en WWW van más allá del modelo estático que se ha expuesto hasta el momento, donde un cliente web se remite a hacer peticiones de documentos HTML "fijos" a un servidor HTTP. En su lugar, un CGI permite al servidor distribuir diferentes documentos HTML dependiendo de las peticiones de los clientes, e incluso genera los documentos hipertexto en tiempo real, es decir, al momento en que se recibe la petición del cliente. En el apéndice D se exponen todos los detalles que son necesarios considerar para lograr la ejecución de un CGI en cualquier servidor WWW.

# IV.1 Vista general del flujo de datos en un CGI.

En cualquier flujo de datos donde intervenga un programa compuerta, se presentan varias etapas: la primera es la transmisión de datos del cliente al servidor: la petición (a). El servidor transfiere la petición al CGI para su ejecución (b). Si existe alguna salida, ésta es transmitida de regreso al servidor (c) y después al cliente (d). La conexión inicial entre el cliente y el servidor es finalizada. Véase figura IV.1

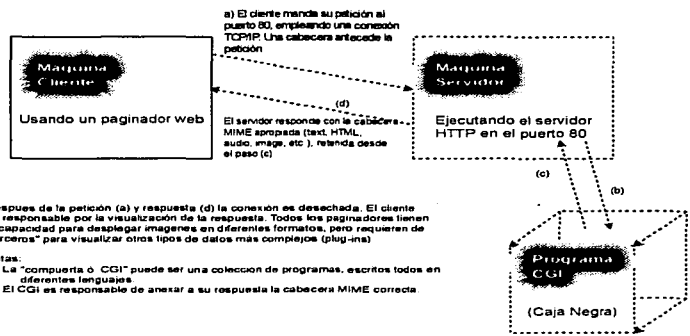



**Figura IV.1.** Visión esquemática del flujo de datos empleando CGIs.

Analicemos el flujo de datos más a detalle:

a) El cliente manda una petición al servidor. Esta petición deberá incluir el tipo de servicio deseado (HTTP, FTP, Telnet, etc.) y la locación ó dirección URL; además una serie de *datos de cabecera* expuestos más adelante en este capítulo.

b) El servidor HTTP recibe la petición, y decide qué hacer con ella a continuación. Puede regresar al cliente un documento HTML si es que la petición así lo indica, o bien, si el servidor reconoce el archivo referenciado como uno ejecutable (un programa CGI), éste será activado y recibirá los datos que le sean enviados por el servidor, los cuales son:

   - **Datos de "cabecera"** recibidos del cliente –si es que existen– y datos de cabecera propios del servidor. Esta información es utilizada por el programa a través de `variables de ambiente`.

- **Parámetros de ejecución** del programa –si es que hay alguno– anexados por el cliente. Esta información está disponible para el CGI a través de variables de ambiente, o mediante la `entrada estándar` de la compuerta.

- Cabe señalar que antes del envío de datos a la compuerta por parte del servidor, éste ***codifica*** la información enviada, por lo que corresponde al CGI hacer la ***decodificación*** pertinente durante su ejecución. Esta etapa de codificación es expuesta más adelante en el presente capítulo.

c) La compuerta ó CGI genera una respuesta en base a los datos recibidos, la cual es regresada al servidor. Esta respuesta puede ser de tres tipos:

- Un **mensaje de error**; generado por el servidor HTTP, el cual se produce al momento de presentarse alguna contingencia en la ejecución del programa.

- Un **mensaje nulo**. Esta situación se presenta cuando el programa compuerta terminó su ejecución sin problemas, pero no regresa información alguna al servidor. La existencia de este tipo de mensajes implica un mal diseño del programa por parte del desarrollador, así que es su responsabilidad evitarlos a como dé lugar.

- **Información producida por la compuerta**. En el momento en que ésta se presenta, el programa la transmite a través de su `salida estándar`, generando: una *cabecera* que entiende el servidor, y los datos de salida, que deberán ser acordes al *formateo MIME* (expuesto más adelante en el capítulo). Los datos de salida deben ser del tipo del que se indica en la cabecera mencionada.

d) El servidor recibe la información resultado de la ejecución del CGI –si es que hay alguna– y decide nuevamente qué hacer con ella, basándose en la cabecera recibida. En general, hay dos tipos de acciones que un servidor podría tomar:

- Proporcionar él mismo el recurso indicado en la salida del CGI (si la cabecera es del tipo **"Location"**).

- Mandar la totalidad de los datos de salida del CGI al cliente para que éste se encargue de su procesamiento y despliegue final al usuario (si la cabecera es del tipo **"Content - type"**).

e) Una vez que el cliente ha recibido la totalidad de los datos, la conexión HTTP es finalizada (se elimina el *circuito virtual* creado para la misma).

# IV.2 Métodos para el paso de datos a un CGI.

El servidor HTTP puede transmitir los datos recibidos del cliente hacia un CGI mediante dos métodos: **variables de ambiente**, ó **entrada estándar.**

## Variables de ambiente.

Existen diversas variables de ambiente que el desarrollador puede utilizar, pero todas ellas caen en dos categorías muy amplias.

Las variables de la primer categoría son independientes de la petición del cliente y siempre tienen el mismo valor; la información que contienen corresponde a propiedades del servidor HTTP, algunas de ellas son:

- **SERVER_SOFTWARE**.- Nombre y versión del servidor web. Formato : nombre/versión.
- **SERVER_NAME**.- El nombre de la computadora donde está instalado el servidor HTTP, su alias DNS, o bien, su dirección IP.
- **GATEWAY INTERFACE**.- El tipo y revisión de los CGIs empleados por el servidor. Formato: CGI/revisión.

El segundo tipo de variables depende de la petición del cliente, así como del tipo de servidor que la atiende:

- **SERVER_PROTOCOL**.- El protocolo empleado por la petición del cliente. Formato: protocolo/revisión.
- **SERVER_PORT**.- El número de puerto por el que fue enviada la petición del cliente.
- **REQUEST_METHOD**.- Se refiere a la manera en que el cliente mandará información a la compuerta: a través de variables de ambiente ó entrada estándar (Métodos GET ó POST).
- **PATH_INFO, QUERY_STRING**.- Véase la explicación más adelante en esta sección.
- **PATH_TRANSLATED**.- El servidor traduce la `ruta virtual` representada en PATH_INFO y la convierte a una `ruta absoluta`.
- **SCRIPT_NAME**.- Contiene una ruta virtual a la compuerta que se está ejecutando en el momento.
- **REMOTE_HOST**.- El nombre de la computadora donde se encuentra el cliente.
- **REMOTE_ADDR**.- La dirección IP del cliente.
- **CONTENT_LENGTH**.- La longitud del `buffer` de datos enviado por el cliente; los CGIs leen dicho buffer y emplean esta variable para "interrumpir" el flujo de datos de la entrada estándar una vez que se ha leído la longitud indicada por la misma.

De la lista anterior, podemos destacar dos variables que resultan esenciales para la programación de una compuerta de cualquier tipo: **QUERY_STRING** y **PATH_INFO**. El desarrollador puede colocar información en éstas variables de diferentes formas:

- A través de una **liga HTML** con argumentos. Por ejemplo:

```
<A HREF=http://indy.aragon.unam.mx/main.cgi?argumento1>
                    Presione aqui </A>
```

  En el ejemplo, la liga hace referencia al CGI llamado **main.cgi**, ubicado en el servidor http cuya dirección es indy.aragon.unam.mx; todo lo que se encuentra después del signo de interrogación **(?)** es colocado en la variable QUERY_STRING; en este caso su valor será "argumento1".

- A través de una **liga HTML** y una "ruta de acceso" ó `path`. Por ejemplo:

```
<A HREF=http://indy.aragon.unam.mx/main.cgi/arg1=valor1/arg2=valor2>
```

  En nuestro ejemplo, la liga hace referencia al mismo CGI y servidor del ejemplo anterior, pero ahora colocará todo lo que se encuentre después del nombre del CGI (**main.cgi**) en la variable PATH_INFO; así, su valor será '/arg1=valor1/ arg2=valor2".

- A través del elemento **FORM** de HTML. Por ejemplo:

```
<FORM METHOD=GET ACTION= "http://indy.aragon.unam.mx/forma.cgi">
  Nombre<INPUT NAME = "nombre"><BR>
  Apellido Paterno <INPUT NAME = "paterno"><BR>
  Apellido Materno <INPUT NAME = "materno"><BR>
  <INPUT TYPE=submit VALUE = "submit">
</FORM>
```

  Cuando empleamos el elemento FORM de HTML y usamos el método "GET", indicamos que todo la información vaciada en la forma se guarde en la variable QUERY_STRING.

## Entrada estándar.

Es obvio que existe carencia de privacía al transmitir datos a un CGI mediante variables de ambiente, pues al hacerlo, la información transmitida es expuesta en el URL correspondiente, dando lugar a la posibilidad de mostrar datos confidenciales al usuario. La solución a ésto es el empleo de la entrada estándar del programa, pues la

información es transmitida directamente al CGI y no es expuesta en ningún momento a la vista del usuario.

El uso de la entrada estándar del CGI puede lograrse al emplear el método "POST" del elemento forma; en el ejemplo visto anteriormente bastará con cambiar la primer línea por:

```
<FORM METHOD=POST ACTION="http://indy.aragon.unam.mx/forma.cgi">
.
.
```

Cuando hacemos uso de este método, el servidor HTTP –de forma automática– emplea la variable de ambiente CONTENT_LENGTH para guardar en ella la longitud total del flujo de datos que se está recibiendo por la entrada estándar. Es responsabilidad del desarrollador hacer uso del valor de dicha variable para asegurarse de leer el **número exacto** de caracteres recibidos, pues de lo contrario, el comportamiento del CGI es totalmente impredecible.

No es necesario apegarse completamente a un sólo método de paso de datos a la compuerta, ya que pueden emplearse cualquiera de ellos (o los dos) al gusto del desarrollador; sin embargo, es justo decir que existe una gran ventaja de seguridad, rendimiento y capacidad en el volumen de información transmitida cuando se hace uso de la entrada estándar del CGI.

## Codificación de los datos enviados al CGI.

Antes de que el servidor transmita los datos recibidos del cliente hacia la compuerta, existe una etapa de codificación de la información. Se codifica lo siguiente:

- Todos los espacios (caracteres en blanco) son cambiados por signos de adición **(+)**.
- Algunos caracteres son sustituidos por su equivalencia hexadecimal precedidos del símbolo de porcentaje **(%)**. Estos caracteres pueden consultarse en la tabla IV.1.
- Los campos dentro de las formas son concatenados por el símbolo ampersand **(&)**.

Por ejemplo, si una forma incluyera la siguiente sintaxis:

```
Campo 1<INPUT NAME=CAMPO1> Campo 2<INPUT NAME=CAMPO2>
```

y la información correspondiente a cada uno de los campos fuera: "campo 1 !@#$%" y "campo 2 ^&*()_+|", el servidor realizaría la siguiente codificación:

CAMPO1=campo+1+%21@%23%24%25&CAMPO2=campo+2+%5E%26*%28%29_%2B%7C

| Caracter | Equivalencia Hexadecimal |
|---|---|
| < | %3C |
| > | %3E |
| # | %23 |
| % | %25 |
| { | %7B |
| } | %7D |
| \| | %7C |
| \ | %74 |
| ^ | %5E |
| ~ | %7E |
| [ | %5B |
| ] | %5D |
| ' | %60 |
| ; | %3B |
| / | %2F |
| ? | %3F |
| : | %3A |
| = | %3D |
| & | %26 |

**Tabla IV.1.** Equivalencia hexadecimal de algunos caracteres ASCII.

Es importante destacar que el desarrollador deberá incluir en sus programas compuerta una rutina para la decodificación de la información recibida del servidor; así, la correcta traducción de los datos de entrada al CGI son responsabilidad absoluta del programador.

## IV.3 Encabezados MIME en el ambiente CGI.

En el momento en que un cliente web realiza una petición, el servidor HTTP la atiende y le da la respuesta adecuada en función de la información solicitada. Cuando se presenta una respuesta del servidor hacia el cliente (paso 4 del flujo de datos de un CGI, figura IV.1), ésta comienza con un encabezado que tiene una sintaxis definida; dicho encabezado es denominado **MIME** (**Multipart Internet Mail Extensions**; Extensiones Multiparte de Correo a través de Internet).

Cuando se trata de un documento HTML estático (que no incluye CGIs), el servidor mismo se encarga de el envío del encabezado MIME acorde al tipo de información regresada por la página. Para el desarrollador en Internet, se presenta una situación diferente, pues es él quien planea la función del CGI, y por tanto es responsable del envío al servidor y posteriormente al cliente del encabezado MIME correcto; así, la primera línea que se imprime en la salida estándar del programa compuerta es una cadena de la forma:

Content-type: <tipo>/<subtipo> <CR><CR>

donde:

| | |
|---|---|
| <tipo> | es el tipo de información que se regresará. |
| <subtipo> | es el subtipo de información que se regresará. |
| <CR> | es un salto de línea. |

Por ejemplo, en el lenguaje de programación C, la secuencia "\n"indica un salto de línea; por lo que la formación de un encabezado MIME del tipo texto, subtipo html sería:

```
printf ("Content-type: text/html \n\n");
```

El tipo "texto" comprende al conjunto estándar de caracteres ASCII imprimibles (un `archivo plano`), en modo texto; el subtipo "html" hace referencia a un texto con indicaciones HTML anexadas a él.

Existen varios tipos y subtipos que pueden ser incluidos en la formación del encabezado MIME, siempre con cuidado de que la información que se entregue a través del CGI corresponda al encabezado declarado. A continuación muestro la totalidad de tipos y subtipos que pueden incluirse en un encabezado MIME (RFC1521 y RFC1522)[1]. El prefijo "x"indica que la extensión no está considerada como un estándar y debe ser cambiada por cada usuario (en el programa cliente) de acuerdo a su preferencia:

| Tipos MIME existentes | | |
|---|---|---|
| **Tipo** | **Subtipo** | **Extensiones más comunes de archivos** |
| text | enriched<br>html htm<br>plain<br>tab-separated-value<br>rich text | html<br>txt |
| multipart | alternative<br>appledouble<br>digest<br>header-set<br>mixed<br>parallel | |
| message | external-body<br>news<br>partial<br>rfc822 | |

**Tabla IV.2.** Tipos y subtipos disponibles en los encabezados MIME

[1] Los RFCs referentes a el formato de encabezados MIME pueden consultarse en línea en: **http://www.cis.ohio-state.edu/htbin/rfc**

| Tipos MIME existentes (continuación) | | |
|---|---|---|
| **Tipo** | **Subtipo** | **Extensiones más comunes de archivos** |
| application | activemessage | |
| | andrew-inset | |
| | applefile | |
| | atomicmail | |
| | commonground | |
| | cybercash | |
| | dca-rft | |
| | dec-dx | |
| | eshop | |
| | iges | |
| | mac-binhex40 | hqx |
| | macwriteii | |
| | mathematica | |
| | msword | doc |
| | news-message-id | |
| | news-transmission | |
| | octet-stream | tar dump readme bin uu exe |
| | oda | oda |
| | pdf | pdf |
| | postscript | ps eps ai |
| | \remote-printing | - |
| | riscos | |
| | rtf | rtf |
| application | slate | |
| | wita | |
| | wordperfect5.1 | |
| | x-dvi | dvi |
| | x-pdf | pdf |
| | x-tar | tar |
| | x-tex | tex |
| | x-www-form-urlencoded | |
| | x-www-pgp-request | |
| | x-www-pgp-reply | |
| | x-www-local-exec | |
| | zip | zip |
| image | | gif |
| | | ief |
| | jpeg | jpeg jpg jpe |
| | rgb | rgb |
| | tiff | tiff tif |
| | xbm | xbm |
| | xpm | xpm |
| | x-xwindowdump | xwd |
| | x-pict | pict |
| audio | basic | au snd |
| | x-aiff | aif aiff aifc |
| | x-wav | wav |
| video | mpeg | mpeg mpg mpe |
| | quicktime | qt mov |
| | x-msvideo | avi |
| | x-sgi-movie | movie |

**Tabla IV.2.** Tipos y subtipos disponibles en los encabezados MIME (continuación).

Una vez que el programa cliente recibe el encabezado producido por la compuerta, se determina si la información recibida podrá ser desplegada correctamente por el paginador, o bien, si se necesita de *"programas auxiliares"* o `plug-ins` que ayuden a un correcto despliegue de la información recibida (como en el caso de archivo de audio, vídeo, aplicaciones, imágenes, etc.); si este es el caso, corresponderá a la configuración de cada programa cliente (paginador) conocer la locación exacta (en la máquina cliente) de las aplicaciones que sean necesarias.

## IV.4 Programación de una compuerta ó CGI.

Hasta este momento hemos revisado detenidamente la manera en que se comporta el flujo de datos desde y hacia un CGI; además, se han expuesto los diferentes detalles que un desarrollador debe considerar para una correcta implementación de un programa compuerta. En esta sección nos enfocaremos al desarrollo de un CGI sencillo que aterrice toda la teoría expuesta con anterioridad.

### Selección del lenguaje de desarrollo.

Se ha comentado que cualquier lenguaje de programación que pueda leer la entrada estándar y variables de ambiente es útil para el desarrollo de CGIs; sin embargo, las diferencias existentes entre ellos en cuanto a rendimiento, facilidad de uso, velocidad, seguridad, etc., hacen que su selección sea una tarea delicada, pues la ejecución de la compuerta es un rasgo de evaluación para juzgar la velocidad y desempeño general de la aplicación programada.

A nivel mundial, los lenguajes empleados típicamente para el desarrollo de CGIs (en plataforma UNIX) son varios: PERL, REXX, C, Tcl, Tk, etc. El más común de ellos es el lenguaje PERL, debido a su facilidad de uso y aprendizaje, además, es de dominio público; sin embargo, presenta debilidades serias en aspectos donde otros lenguajes como C le superan abismalmente. La tabla IV.3 presenta un análisis comparativo de características de varios lenguajes de programación (plataforma UNIX), que nos servirá para apoyarnos en la selección de uno de ellos.

Gracias a la tabla IV.3 podemos concluir que todos los lenguajes tienen ventajas y desventajas; sin embargo, considero que el lenguaje C es el más indicado para nuestra tarea, además proporciona características interesantes que habría que analizar:

| Aspecto de comparación | PERL | REXX | C | Tcl | Tk |
|---|---|---|---|---|---|
| Velocidad de ejecución | Media | Media | **Alta** | Baja | Baja |
| Modo de ejecución | Intérprete | Intérprete | **Módulo ejecutable** | Intérprete | Intérprete |
| Portable a maquinas que no posean su compilador ó intérprete | No | No | **Sí** | No | No |
| Necesita su código fuente para ser ejecutado | Sí | Sí | **No** | Sí | Sí |
| Facilidad para la obtención de rutinas y librerías ya desarrolladas | Alta | Media | **Alta** | Media | Baja |
| Fácil de aprender | Sí | Sí | **No** | Sí | No |
| Capacidad para el manejo de grandes volúmenes de información | Alta | Alta | **Alta** | Media | Baja |
| Amigable para el desarrollador | Sí | Sí | No | Sí | Sí |

**Tabla IV.3.** Tabla comparativa de los lenguajes más populares para el desarrollo de CGIs en Internet.

- Los programas en lenguaje C son compilados, ligados y se produce un archivo ejecutable con instrucciones en lenguaje máquina que hacen su ejecución cientos de veces más rápida que la de un programa interpretado (como sucede con Perl, REXX, Tcl, etc.).

- Un programa en C es mucho más portable que uno que hace uso de intérpretes (como Perl, Tcl, etc.); pues para el primero basta con copiar el archivo ejecutable a su nueva locación[2] (no importa si el compilador de C existe o no en ella), en tanto que para el segundo es necesario que el intérprete correspondiente esté presente en la nueva ubicación del programa.

- El archivo ejecutable del programa en C (archivo binario) evita que su código fuente sea explorado por gente curiosa, cuyo objetivo sea –posiblemente– sacar provecho de algún error que pudiera existir en el código del mismo. Los programas de lenguajes como Perl ó REXX son fácilmente explorables, pues el intérprete actúa directamente sobre el código fuente, el cual se encuentra siempre a la vista de cualquier persona, incluso de gente mal intencionada.

- Cualquier distribución de sistema operativo UNIX incluye sin costo alguno un compilador `ANSI C`; no es necesario buscarlo como distribución gratuita en Internet (tal como sucede con PERL, REXX, etc.), garantizando con ésto una base de lenguaje de programación con presencia consistente en diversas plataformas de cómputo.

- La única desventaja que presenta el lenguaje C es la dificultad existente en su uso (ésto para personas que no posean experiencia con él); sin embargo, bien vale la pena pasar un periodo de aprendizaje del mismo para poder emplear las

[2] Para poder transportar un programa ejecutable de una máquina a otra sin recompilarlo es necesario que ambas posean una misma arquitectura: SUN-SUN, SGI-SGI, HP*nnn*-HP*nnn*, etc. En caso de no ser así, basta con recompilar el código en la nueva computadora y se tendrá un modulo en lenguaje máquina listo para ser ejecutado.

grandes facilidades y cualidades que han hecho del lenguaje C uno de los más populares entre los desarrolladores a nivel mundial.

## Codificación del programa compuerta.

Para que un programa compuerta sea activado, y su salida se manifieste en forma de hipertexto, es necesario mandar ejecutarlo a través de una página HTML. Por ejemplo, en el documento cuyo despliegue es representado por la figura III.4, la línea:

```
<FORM Method="POST" Action="http://indy.aragon.unam.mx/cgi-bin/cruzcc/forma1.cgi>,
```

indica la utilización del elemento forma, así como la invocación del programa compuerta llamado **forma1.cgi** (el cual se encuentra en el directorio cgi-bin/cruzcc del servidor cuyo domino es indy.aragon.unam.mx). El código fuente de el CGI mencionado es:

```
#include <stdio.h>
#include <util.h>
#define MAX_CAMPOS 30
/*****************  Seccion de variables  *********************/
struct{
 char *nombre;
 char *valor;
}entradas[MAX_CAMPOS];
int data_size;
int indice;
/*******************************************************************/
main()
{
 printf("Content-type: text/html\n\n");      /* Impresion del encabezado MIME          */
 printf("<HTML>\n");                         /* Inicio de la pagina HTML               */
 printf("<TITLE>Valores de retorno de un CGI</TITLE>\n<BODY>\n");
 data_size=atoi(getenv("CONTENT_LENGTH"));       /* Lee largo de cadena recibida       */
                                                 /*por la entrada estandar             */
 printf("<BR>Las entradas recibidas son:<BR><P>\n");
 printf("<B>\n");
 for(indice=0;data_size && (!feof(stdin));indice++)
 {
  /* Funcion para la obtencion de una palabra a partir de la entrada estandar */
  entradas[indice].valor = obtiene_palabra(stdin,'&',&data_size);
  /* Funcion para la sustitucion del signo "+" por espacios */
  cambia_adicion(entradas[indice].valor);
  /* Funcion para la traduccion de algun codigo hexadecimal a caracter ASCII */
  decodifica_total(entradas[indice].valor);
  /* Funcion para la obtencion de una palabra a partir de la entrada estandar */
  entradas[indice].nombre = separa_campos(entradas[indice].valor,'=');
  printf("%s = %s<BR>",entradas[indice].nombre,entradas[indice].valor);
 } /* del for */
 printf("</B>\n");
 printf("</BODY></HTML>\n");            /* Fin de la pagina HTML */
}
```

En el código anterior es posible observar algunas propiedades características del CGI (expuestas en secciones previas). La compuerta realiza lo siguiente:

- Produce un encabezado MIME acorde al tipo de datos regresado al cliente por parte del servidor (en nuestro caso, texto plano con formateo hipertexto): printf("Content-type: text/html\n\n");
- Se ocupa de la elaboración (en tiempo real) de la página hipertexto que será desplegada en el cliente; es decir, produce el código HTML que dará origen al documento desplegado en el paginador del cliente: printf("<HTML>\n"); . . . . . printf("</BODY></HTML>\n");
- Se encarga de conocer la longitud de la cadena que recibirá a través de su entrada estándar mediante el uso de la variable de ambiente CONTENT_LENGTH: data_size=atoi(getenv("CONTENT_LENGTH"));
- Realiza la decodificación de los datos recibidos acorde a lo expresado en secciones anteriores; mediante las funciones: obtiene_palabra, cambia_adicion, decodifica_total, y separa_campos. Nota: El código fuente de estas funciones puede consultarse en el Apéndice C.
- Despliega en la página creada en tiempo real los datos que leyó a partir de la entrada estándar:
  printf("%s = %s<BR>",entradas[indice].nombre,entradas[indice].valor);

El resultado final, es decir, la página HTML producida (en tiempo real) por el CGI anterior –basándonos en los datos que es posible ver en la figura III.4– es el siguiente:

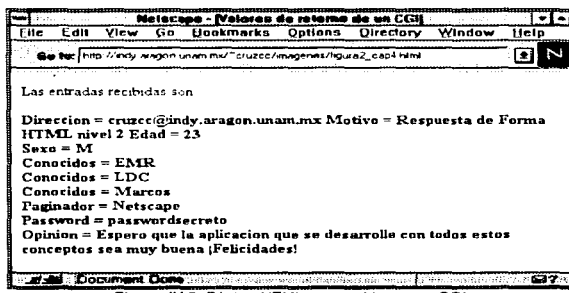



Figura IV.2. Página HTML construida por un CGI.

Es obvio que no siempre deberemos desplegar en pantalla todos los datos recibidos por la compuerta; es más, habrá ocasiones en los que ni siquiera despleguemos uno solo, pero todo ésto dependerá del diseño de la aplicación y en general del desarrollador, quien hará uso de la información recibida como mejor crea conveniente para un correcto desempeño de la aplicación programada.

## IV.5 Aspectos de seguridad en la programación de un CGI.

La seguridad es un aspecto delicado en las cuestiones referentes a Internet, pues la apertura mundial de ésta permite a cualquier persona desde cualquier parte del mundo accesar la máquina que desee dentro de la red global. Así, como WWW es un servicio de Internet, las páginas o documentos HTML (incluidos los CGIs) están totalmente expuestos a la exploración, uso y en ocasiones abuso de cualquier usuario de la red.

Las páginas HTML por si solas no representan un foco de peligro para la integridad de un sitio web, pues los elementos que éstas ocupan permiten únicamente acciones definidas y perfectamente controladas por el desarrollador. El problema surge cuando a los documentos HTML se le incorporan programas compuerta, pues aunque puede pensarse que el desarrollador logra controlar totalmente las acciones de su programa, no siempre es así, ya que llegan a darse situaciones especiales en las que el CGI no responde como debiera, o bien, el sistema operativo le permite realizar acciones que ponen en riesgo el servidor donde se encuentre instalado el *demonio httpd*.

En definitiva, ningún sistema es totalmente seguro, siempre existirá alguien que intente –y en ocasiones logre– corromperlo; sin embargo, resulta siempre importante conocer los "puntos débiles" inherentes al mismo, con la finalidad de evitarlos, o bien, trabajar más arduamente para fortalecerlos y evitar así –en la medida de lo posible– la irrupción de gente no deseada al mismo. A continuación menciono algunos puntos que el desarrollador debe tomar en cuenta al programar sus compuertas y así incrementar la seguridad de las mismas:

- Es necesario **evitar que los archivos fuente de nuestras compuertas puedan ser vistos por cualquier persona**. En caso de no evitarlo, permitimos a gente desconocida explorar los programas, estudiarlos, y encontrar detalles en los mismos que pudieran ser utilizados de manera maliciosa para obtener información restringida.

- **Nunca hay que confiar en los datos proporcionados por un usuario**, pues éstos podrían tener caracteres especiales que posiblemente variarían la función de la compuerta. La mejor forma de defenderse de ésto es verificar que los datos recibidos sean únicamente letras ó digitos. Por ejemplo, supongamos que de alguna manera el usuario se ha dado cuenta que cierta entrada de datos de nuestro CGI es ejecutada a manera de un comando UNIX, empleando el punto y coma (;) –que indica la continuación de la línea de comandos en UNIX– se

podrían introducir varias instrucciones que serían ejecutadas sin problema alguno por el sistem operativo (rm -rf *; mail snoopy@cracker.mx < /etc/passwd; etc...).

- **Nunca emplear la entrada proporcionada por el usuario como comando del sistema operativo.** En ocasiones es necesario ejecutar comandos del sistema operativo; en caso de necesitarse, es recomendable no emplear como parámetros la entrada proporcionada por el usuario, pues no sabemos qué comando ó secuencia de instrucciones introducirá y, por lo tanto, sería imposible defendernos de alguna actitud maliciosa de su parte. El ejemplo del punto anterior ilustra claramente lo aquí expuesto.

- **Cuidar el tamaño de la entrada estándar en lenguaje C.** Es importante hacer uso de la función malloc() en la lectura de la entrada estándar, pues permite asignar espacio en memoria de un tamaño acorde a la longitud de la cadena recibida como entrada (en lugar de crear un espacio estático y suponer ciegamente que la entrada nunca será mayor de cierto tamaño). Una táctica muy empleada es mandar como entrada de la compuerta cadenas de comandos exageradamente largas, en un intento por dañar el sistema; se ha sabido que de esta forma es posible truncar la ejecución de la compuerta y permanecer en sesión en el servidor http.

- **Evitar el uso de variables de ambiente propias del sistema operativo**[3], pues éstas podrían contener valores alterados; basta con que una persona maliciosa hubiese descubierto la función de dichas variables en el programa compuerta para intentar modificarlas. Por ejemplo, en el caso en que una variable de ambiente guardara la referencia a algún directorio (donde se almacenara información restringida), su alteración por un nuevo directorio implicaría guardar o leer datos desde una locación diferente a la originalmente establecida; para evitar ésto, es mucho más conveniente hacer uso de la `ruta absoluta` de dicho directorio directamente en el CGI.

Obviamente, el grado de seguridad necesario lo determina el tipo de aplicación a desarrollar, así como la importancia de la información que ésta maneje. Por ejemplo, no es necesario el mismo nivel de seguridad en una página HTML personal, que en un sistema de apartado de vuelos aéreos mediante tarjeta de crédito; la importancia de la información manejada es claramente distinta. Existen personas que afirman que Internet es muy insegura, y tienen razón, pues cualquier persona con conocimientos profundos de la red (y ganas de delinquir) podría obtener cualquier información que se estuviera transmitiendo por Internet. Los sistemas de seguridad para la red global implican métodos intrincados y confusos, meritorios de un estudio de tesis; sin embargo, el grado en que estos debieran influir en una aplicación cualquiera está dado —como dije anteriormente— por la importancia y confidencialidad de la información en juego.

[3] Nótese que no estoy hablando de las variables de ambiente propias del servidor HTTP, expuestas en secciones anteriores, sino de las variables de ambiente del servidor (del sistema operativo) que sirve de plataforma para el demonio httpd.

# 5

# Caso de estudio

Hasta el capítulo anterior, se ha estudiado con detenimiento las grandes oportunidades que para el desarrollo de aplicaciones nos proporcionan HTML y CGI en conjunto. Considero que tenemos ya los fundamentos teóricos suficientes para incursionar en el análisis, diseño, implantación y mantenimiento de un sistema estable y funcional para Internet, empleando –obviamente– TODA la tecnología expuesta en los capítulos anteriores.

La oportunidad de desarrollar una aplicación ilustrativa para el presente trabajo de tesis, fue brindada por el Centro de Cómputo de la Escuela Nacional de Estudios Profesionales Aragón (UNAM), partiendo de su necesidad de contar con sistemas en Internet que empleen tecnología confiable, capaces de auxiliar a la difusión de una imagen positiva de la Institución, así como de dar a conocer algunos de los servicios educativos con los que ésta cuenta.

El desarrollo de nuestro caso de estudio está comprendido en cinco etapas:

- Planeación.
- Análisis.
- Diseño.
- Construcción.
- Implantación.

## V.1 Planeación.

### Definición del problema.

La Escuela Nacional de Estudios Profesionales Aragón es un organismo de la Universidad Nacional Autónoma de México que tiene entre sus objetivos principales ser un centro educativo y de difusión del conocimiento humano que permita el desarrollo cultural de una de los zonas más marginadas de la Ciudad de México: Ciudad Nezahualcóyotl.

Actualmente, ENEP Aragón cuenta con dos servidores web en sus instalaciones[1], uno de ellos localizado en el Centro de Cómputo de la institución (figura V.1). En él es posible visitar páginas HTML que permiten conocer instalaciones del plantel, calendario de actividades semestrales, páginas personales, etc.; sin embargo, hasta Febrero de 1997, todos los documentos expuestos eran estáticos, carentes de una interacción en tiempo real con alguna otra aplicación ajena a WWW (no empleaban CGIs).

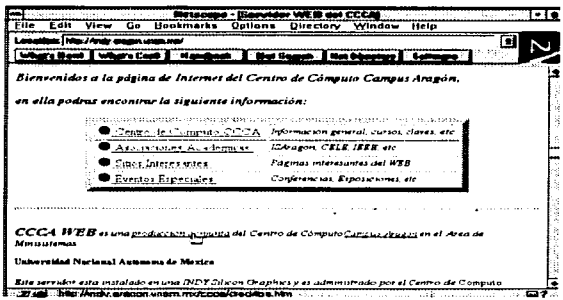



Figura V.1. Página Base (Home Page) del servidor WWW de ENEP Aragón (Febrero, 1997).

A pesar de no contar con una gran cantidad de lenguajes de desarrollo, ENEP Aragón posee los compiladores y aplicaciones proporcionados por los distribuidores del equipo de cómputo que se encuentra en sus instalaciones. La posibilidad de adquirir una infraestructura de software más robusta que la actual está totalmente descartada.

***Se necesita desarrollar un sistema de páginas HTML que se ejecute en el servidor web del Centro de Cómputo de ENEP Aragón, aprovechando toda la infraestructura de cómputo de la que dispone este organismo. El sistema mencionado deberá ser innovador y proporcionar a la institución elementos técnicos novedosos que mejoren los ya existentes. Además, deberá difundir una imagen positiva de la escuela, y mostrar al mundo entero los avances que en materia de desarrollo en Internet (WWW) se tienen.***

[1] Pueden visitarse en http://informatica.aragon.unam.mx y http://indy.aragon.unam.mx.

## Especificación de recursos.

Los recursos de hardware y software que tan amablemente fueron ofrecidos por la Coordinación del Centro de Cómputo de ENEP Aragón (CCCEA) para el desarrollo del sistema de páginas HTML son:

- Un servidor WWW (demonio httpd) de NCSA, corriendo en una estación de trabajo Silicon Graphics "Indy" R4600pc a 133MHz, plataforma UNIX (IRIX IV).
- Dos servidores Apollo 9000 de Hewlett Packard, modelos 720 y 730, con velocidades de 66 Mhz, plataforma UNIX (HP-UX)
- Una licencia de uso del manejador de Base de Datos All-Base de Hewlett Packard, ubicado en la máquina Apollo 9000 modelo 730 antes descrita.
- Compiladores de lenguaje C, Pascal y Fortran, así como el intérprete del lenguaje Perl, ubicados en la máquina Apollo 9000 modelo 730.
- Una conexión dedicada a Red UNAM, es decir, una vía libre de acceso a los servicios de Internet (a través de la propia infraestructura de la Universidad Nacional Autónoma de México)

Los demás recursos con los que cuenta el CCCEA (en materia de hardware y software) fueron restringidos debido a políticas internas de uso y control.

## Restricciones para el desarrollo de la aplicación WWW.

Las restricciones impuestas por el CCCEA para el desarrollo de nuestra aplicación no fueron muchas, éstas se resumen a continuación:

- Emplear únicamente los recursos expuestos en la sección anterior; en caso de requerirse algo extra, esto deberá ser notificado a la Coordinación para intentar resolver la petición en ese momento.
- Las claves de usuario asignadas NO tendrán privilegios especiales (como administrador o algún otro): serán claves genéricas como las de uso para estudiantes. En caso de requerirse algún servicio de administración, esto deberá notificarse a la persona responsable del equipo en cuestión.
- No se permite alterar la configuración actual de los servidores, ó topologías del equipo de cómputo empleado.
- No es posible incrementar la infraestructura de software existente en el CCCEA, es decir, no se adquirirán nuevos programas de cómputo. En caso de necesitarse

la instalación de programas `shareware`, ésta deberá ser notificada antes al administrador de la máquina donde se desee realizar, con el objeto de evaluar el software en cuestión y certificar su legalidad y utilidad al Centro de Cómputo.

## Propuesta del proyecto.

Apegándonos a las restricciones y recursos establecidos por el CCCEA, se propone el siguiente proyecto:

Un **Sistema de Apartado Remoto a Cursos (SARC)** que emplee la tecnología WWW (Internet) para cumplir sus propósitos.

**Justificación del sistema.**

Actualmente podemos encontrar en WWW muchas páginas HTML que permiten a las personas que las visitan registrarse como usuarios de sistemas diversos: permitiendo el pago de servicios, reservaciones de hotel, compra de productos, etc. Existe un factor común entre todos ellos, y no es precisamente el hecho de proporcionar el servicio que ofrecen, sino la popularidad que obtienen al hacerse propaganda a través de Internet.

El alcance de una compañía o institución educativa con presencia en WWW es de varios millones de personas, quienes después de visitar el sitio web, no sólo saben de la existencia de la institución anfitriona, sino que conocen su giro, servicios, organismos, personas involucradas e infinidad de detalles que antes les eran totalmente desconocidos.

ENEP Aragón tiene presencia en WWW a través de dos servidores web (*informática* e *indy*); sin embargo, dentro de todas las páginas que los conforman no existe ninguna información acerca de los cursos de cómputo que nuestra escuela ofrece. Podría pensarse que esto se soluciona con la simple inclusión de una página HTML estática que informara acerca de ello, pero ¿es atractiva una página HTML estática? ¿es justificable una página estática para un centro de estudios de nivel profesional con carreras afines a computación? ¿qué tan actualizada estaría la información de esta página? ¿sería necesario modificarla cada semestre para incluir o eliminar cursos? ¿cuánto tiempo tardaría esta modificación? ¿qué pasa con la gente interesada en apartar su lugar en un curso?.

Es innegable el hecho de que un Sistema de Apartado de Cursos en WWW reflejaría una infraestructura sólida de ENEP Aragón; daría confianza a las personas que llegaran a inscribirse a algún curso, y sería un medio muy eficiente de dar a conocer al mundo entero los servicios y oportunidades de desarrollo que ofrece nuestra escuela.

**Objetivos:**

- Emplear la tecnología WWW para el desarrollo de un Sistema de Apartado de Cursos a distancia; es decir, un sistema en el que el usuario sea capaz de consultar a través de Internet los cursos impartidos en el Centro de Cómputo de ENEP Aragón, y apartar (por un tiempo determinado) su lugar en algunos de ellos.

- Mostrar al mundo los avances que en materia de desarrollo de sistemas para Internet ha alcanzado ENEP Aragón.

- Permitir una mayor difusión tanto de ENEP Aragón como de los cursos que ésta imparte, en apoyo a los medios tradicionales empleados.

- Incluir en los documentos HTML que compongan el Sistema de Apartado de Cursos, elementos que permitan una mayor difusión de ENEP Aragón, así como la promoción de medios electrónicos existentes (concretamente ligas a otros servidores *WWW* y *gopher* de Aragón)

**Descripción General del Sistema.**

El Sistema de Apartado de Cursos propuesto, empleará como interfaz de usuario las facilidades proporcionadas por **HTML** para la construcción de "páginas hipertexto", las cuales podrán ser consultadas por cualquier persona que tenga acceso a WWW.

El control de las personas que hagan uso de **SARC** será llevado a cabo empleando el DBMS con que cuenta la Institución: **All-Base de Hewlett Packard**, cuya metodología de desarrollo de aplicaciones se apega al "*Diseño Relacional de Bases de Datos*", expuesto en secciones posteriores del presente capítulo.

El elemento que permitirá la "comunicación" entre las dos entidades ya mencionadas (las páginas HTML y el DBMS) será un **programa compuerta ó CGI**. Con esto, los documentos HTML desarrollados en SARC serán dinámicos, es decir, variarán en su respuesta dependiendo de las selecciones del usuario, realizando consultas en línea a la Base de Datos que se diseñará para este propósito. El **lenguaje de programación C** será la base para la implementación de las compuertas necesarias, pues éste lenguaje es consistente en todos los equipos presentes en el CCCEA.

Existe un detalle muy interesante que se presenta debido a las restricciones establecidas por el CCCEA: como NO se debe variar la configuración del equipo de cómputo actual, los elementos que necesitamos para el desarrollo de SARC se ubican en dos lugares distintos:

- El servidor WWW (demonio httpd) en la computadora **Indy de Silicon Graphics**.

- El servidor de Bases de Datos en la computadora **Apollo 9000, modelo 730 de Hewlett Packard**.

La interacción entre ambos servidores se conseguirá a través de un grupo de utilerías creadas por el Instituto Nacional de Estándares y Tecnología de los Estados Unidos (National Institute of Standards and Technology; por su traducción al inglés), llamado **EXPECT**[2]. Puede consultarse una ayuda rápida de Expect en el Apéndice A.

Paralelamente al desarrollo de la interfaz de usuario, se implementará con la misma tecnología una serie de facilidades que permitan la administración eficiente de SARC, esto con la finalidad de proporcionar al CCCEA un camino viable para la utilización y mantenimiento de el Sistema de Apartado, sin necesidad de requerir de su administrador, conocimientos muy profundos de las tecnologías implicadas en el desarrollo de la presente aplicación.

**Límites del Sistema.**

El Sistema de Apartado de Cursos no realizará ninguna tarea extra al simple apartado de un curso; es decir, únicamente asegurará por tiempo limitado el lugar de una persona en un curso determinado. No se implementará ninguna solución para la inscripción, control de alumnos, gestión de calificaciones ni ninguna otra tarea que no se relacione con apartar un curso.

## V.2 Análisis.

### Flujo Físico de la Información.

En la actualidad no existe ningún procedimiento en ENEP Aragón que se siga para el apartado de cursos, lo más cercano a esto es la inscripción, cuyo flujo físico de información (únicamente el necesario para que un alumno asegure su lugar en un curso) se ilustra en la figura V.2[3]

[2] Expect es empleado para automatizar la interacción con programas, en especial aquellos que esperan ("expect", por su traducción al inglés) algún dato del usuario a través del teclado. Por ejemplo: envío de archivos a través de ftp; interactuar con programas que soliciten passwords; o bien –como se aplicará en nuestro caso de estudio– para realizar la interacción entre dos servidores físicamente distintos.

[3] Obviamente el procedimiento de inscripción a un curso implica mucho más detalles que los representados en la figura V.2; sin embargo, cabe recordar que nuestro objetivo NO es automatizar la inscripción a cursos, sino conocer el proceso inicial del mismo (en el que el interesado asegura su lugar en un curso determinado) para así establecer los procedimientos que intervendrían en su apartado.

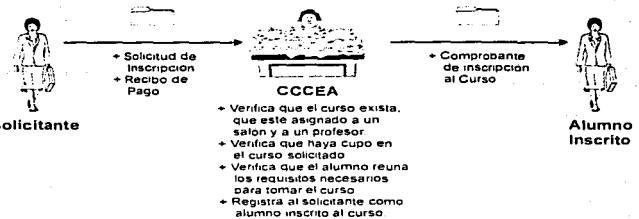



**Figura V.2.** Flujo Fisico de la Información para la Inscripción de un alumno en algun curso de ENEP Aragón.

**Apartar un curso implica asegurar un lugar en el mismo sin estar inscrito**. Así, gracias a la figura anterior, logramos establecer que el apartado de cursos cumple con casi todos los elementos de la inscripción, excepto por la entrega del recibo de pago por parte del alumno, y la entrega del comprobante de inscripción por parte de CCCEA. El flujo físico detectado para el apartado de cursos, es el siguiente:

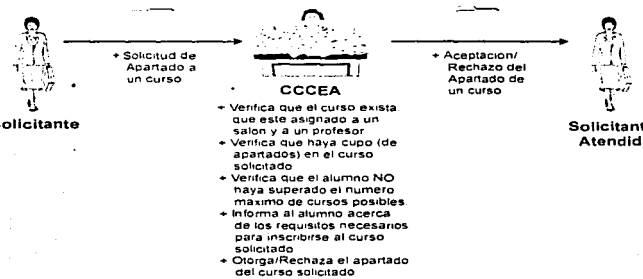



**Figura V.3.** Flujo Fisico de la Información para el Apartado de un curso.

Además del flujo físico establecido para el apartado de cursos, se ha determinado aplicar ciertas restricciones al procedimiento:

- Ninguna persona podrá realizar más de 2 apartados en un mismo periodo; evitando así el abuso en la utilización del sistema.
- El usuario deberá proporcionar al sistema algún dato que facilite su localización para su inscripción en caso de ser necesario (teléfono, e-mail, etc.). De no ser así, no se otorgará el apartado del curso solicitado.
- El apartado del curso sólo tendrá vigencia durante dos días, contados a partir del momento en que el usuario haya hecho uso del sistema; después de este periodo de tiempo, el apartado ya no será válido.

## Consideraciones para WWW.

SARC no está enfocado a ningún tipo de persona en especial: no es tendencioso ni intenta ser elitista; el único requisito que deberán reunir sus usuarios (además de tener acceso a Internet) será tener deseos reales de emplear el sistema para apartar los cursos que tomarán en nuestra escuela.

Uno de los objetivos principales de SARC es proporcionar información veraz y oportuna acerca de los cursos impartidos en ENEP Aragón, lo cual está ampliamente garantizado debido a la dinamicidad de las páginas HTML que conformarán el sistema; éstas obtendrán la información desplegada al usuario en tiempo real, pues gracias a la tecnología CGI, consultarán en línea una base de datos centralizada que contiene la información acerca de los cursos del plantel.

En lo referente a los documentos HTML que habrán de desarrollarse, es oportuno mencionar que emplearán elementos HTML nivel 3, para paginadores exclusivamente gráficos; esto aunque es excluyente para paginadores sin dicha capacidad, es realista al aceptar que un 97.5% de ellos a nivel mundial pueden desplegar gráficos[4]. A priori es posible establecer que los documentos hipertexto a desarrollar contarán únicamente con los elementos necesarios para permitir la interacción con el usuario sin proporcionarle demasiadas facilidades para intentar accesar servicios restringidos de la aplicación.

[4] Fuente: **ftp://nic.merit.edu** en el directorio /nsfnet/statistics.

## V.3 Diseño.

El diseño del Sistema de Apartado de Cursos se desarrollará en tres etapas principales, las cuales son:

- Diseño del Sistema de Apartado Remoto a Cursos (SARC).
- Diseño de la Base de Datos.
- Diseño de la interfaz del usuario, es decir, el conjunto de páginas HTML que se emplearán en el sistema.

### Diseño del Sistema de Apartado Remoto a Cursos (SARC).

Existe una técnica de diagramación denominada de "Flujo de Datos"[5], la cual representa el **camino (flujo)** que siguen los datos a partir de una **unidad externa de entrada** para proporcionar información a una **unidad externa de salida**; durante el flujo que siguen los datos, ésto son afectados por **procesos**, los cuales suelen apoyarse en **contenedores de datos** para realizar la afectación indicada. La simbología empleada en estos diagramas es la siguiente:

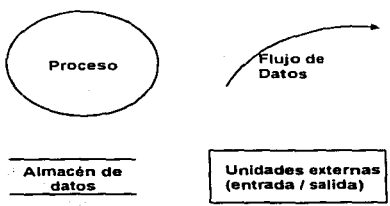



**Figura V.4.** Simbología empleada en un Diagrama de Flujo de Datos

[5] Una bibliografía para estudiar con detenimiento los Diagramas de Flujo de Datos la podemos encontrar en: **YOURDON**, Edward, "Análisis Estructurado Moderno", Editorial Interamericana, México, 1987, 3a edición.

El espíritu de esta técnica es la materialización de los procesos empleados en un sistema, expresando gráficamente los flujos de información comprendidos para convertir una entrada en una salida. Puede trazarse con diferentes niveles de detalle para mostrar los pasos del procedimiento amplia ó detenidamente.

En caso de que se realice el **refinamiento** de los diagramas de nivel superior; es decir, se tracen los detalles de los procesos que así lo requieran en un diagrama, es indispensable mantener el equilibrio entre las entradas y salidas del proceso, con las entradas y salidas que presente su refinamiento. Por ejemplo, imaginemos un proceso que recibe una entrada, pero produce dos salidas; el refinamiento de su DFD, sin importar los flujos ó número de procesos que comprenda, deberá respetar el hecho de recibir una sola entrada (de su nivel superior), y producir dos salidas (hacia su nivel superior); si ésto se cumple en todos y cada uno de los DFDs y sus refinamientos, estaremos hablando de un **sistema equilibrado**.

El primer paso de esta técnica es la elaboración de un "Diagrama de Contexto", que representa las relaciones existentes entre el sistema a diseñar (nuestro Sistema de Aparta Remoto a Cursos) y el medio exterior.

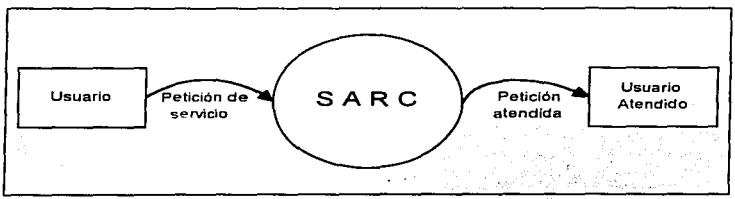
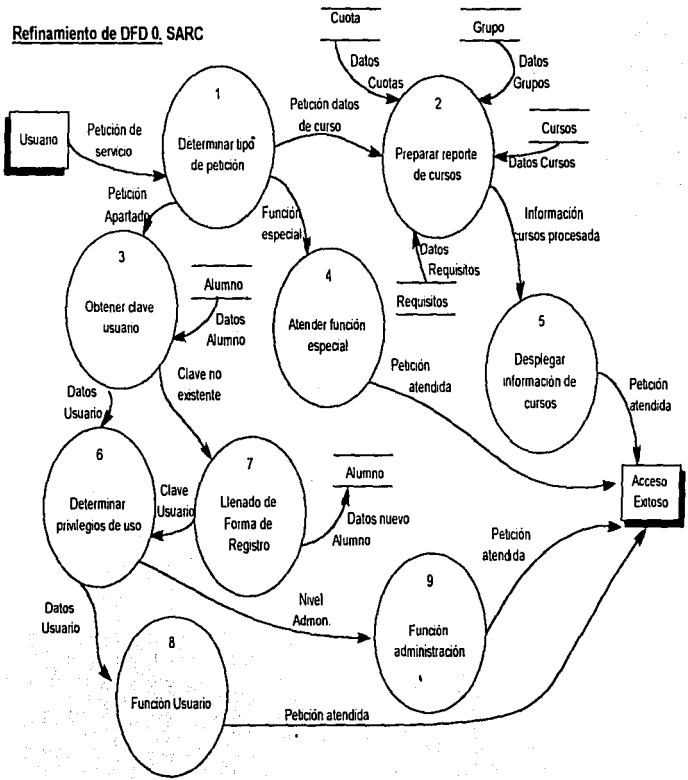



**Figura V.5. Diagrama de Contexto del Sistema de Apartado Remoto a Cursos.**

A continuación se presentan los `Diagramas de Flujo de Datos` (DFDs) que materializan el diseño efectuado para SARC, seguidos de una explicación breve de cada nivel.

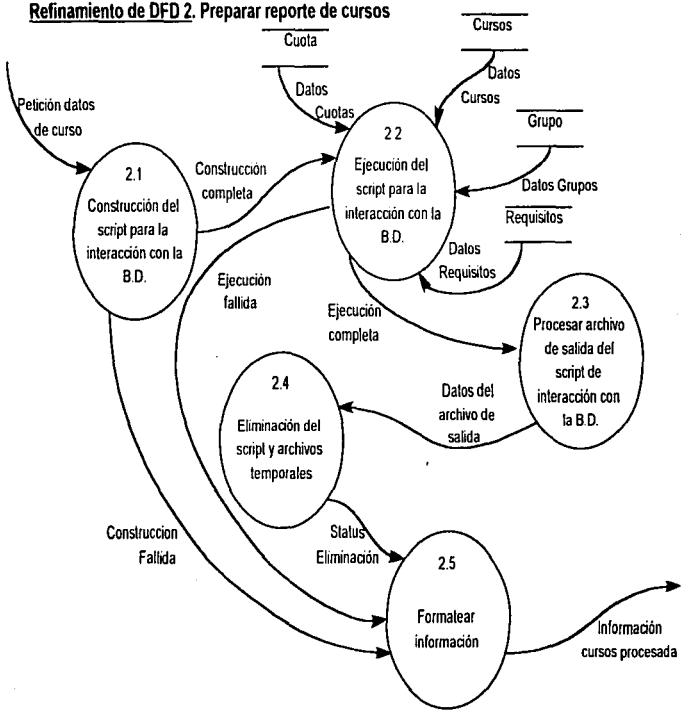

Refinamiento de DFD 0. SARC
Cuota
Grupo
Datos Cuotas
Datos Grupos
Usuario
Petición de servicio
1
Determinar tipo de petición
Petición datos de curso
2
Preparar reporte de cursos
Cursos
Datos Cursos
Petición Apartado
Función especial
Información cursos procesada
Datos Requisitos
3
Obtener clave usuario
Alumno
Datos Alumno
4
Atender función especial
Requisitos
5
Desplegar información de cursos
Clave no existente
Petición atendida
Petición atendida
Datos Usuario
6
Determinar privilegios de uso
7
Llenado de Forma de Registro
Alumno
Clave Usuario
Datos nuevo Alumno
Acceso Exitoso
Petición atendida
Nivel Admon.
9
Función administración
Datos Usuario
8
Función Usuario
Petición atendida

| Proceso | 1. Determinar tipo de petición. |
|---|---|
| Descripción | Determina el tipo de servicio que proporcionará SARC en base a la petición realizada por el usuario, a saber: atender función especial; petición de datos de un curso, y petición de apartado. |
| Restricciones | Ninguna. |
| Flujo datos entrada | Petición de servicio. |
| Flujo Datos Salida | Petición de apartado; función especial; petición de datos de cursos. |
| Almacén de datos | Ninguno. |

| Proceso | 5. Desplegar información de cursos. |
|---|---|
| Descripción | Formatea la información de los diferentes cursos que se impartirán y los despliega al usuario. |
| Restricciones | Los datos proporcionados serán generales; no deberá desplegarse ninguna información restringida: cupo, demanda, nombre de alumnos inscritos, etc. |
| Flujo datos entrada | Información procesada de cursos. |
| Flujo Datos Salida | Petición atendida. |
| Almacén de datos | Ninguno. |

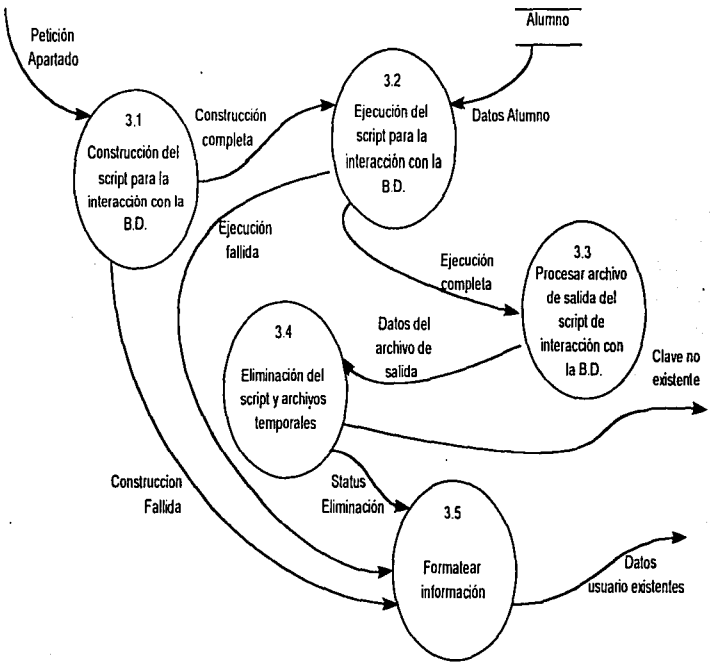
| | |
|---|---|
| **Proceso** | **6. Determinar privilegios de uso.** |
| **Descripción** | Establece en base a la clave del usuario insertada, el nivel de restricción que se aplicará. Para SARC sólo existen dos: usuario y administrador. |
| **Restricciones** | Sólo deberán existir dos niveles de seguridad: usuario y administrador. |
| **Flujo datos entrada** | Datos del usuario (proporcionados de forma directa); clave del usuario (proveniente de la forma de registro a SARC). |
| **Flujo Datos Salida** | Datos del usuario (para el nivel usuario); indicación de privilegios de administración (para el nivel administrador). |
| **Almacén de datos** | Ninguno. |

| | |
|---|---|
| **Proceso** | **7. Llenado de forma de registro.** |
| **Descripción** | En caso de no haberse ubicado una clave de usuario, o bien, por voluntad, se podrá llenar una forma de registro para hacer uso de los servicios de SARC. |
| **Restricciones** | La forma será llenada libremente por el usuario. La clave de acceso será asignada por el sistema (facilita el control de usuarios). |
| **Flujo datos entrada** | Indicación de clave no existente; ésta puede darse ya sea a petición del usuario, o bien, por la inexistencia de la clave introducida. |
| **Flujo Datos Salida** | Clave del usuario (asignada por el sistema); datos del nuevo alumno hacia el contenedor (tabla) Alumno. |
| **Almacén de datos** | INSERT Alumno (nombre, cve_alumno, compania, ciudad, telefono, email, url). |


Refinamiento de DFD 2. Preparar reporte de cursos
Cuota
Cursos
Datos
Cuotas
Datos
Cursos
Petición datos
de curso
2.2
Ejecución del
script para la
interacción con la
B.D.
Grupo
Datos Grupos
Requisitos
Datos
Requisitos
2.1
Construcción del
script para la
interacción con la
B.D.
Construcción
completa
Ejecución
fallida
Ejecución
completa
2.3
Procesar archivo
de salida del
script de
interacción con
la B.D.
2.4
Eliminación del
script y archivos
temporales
Datos del
archivo de
salida
Construccion
Fallida
Status
Eliminación
2.5
Formatear
información
Información
cursos procesada

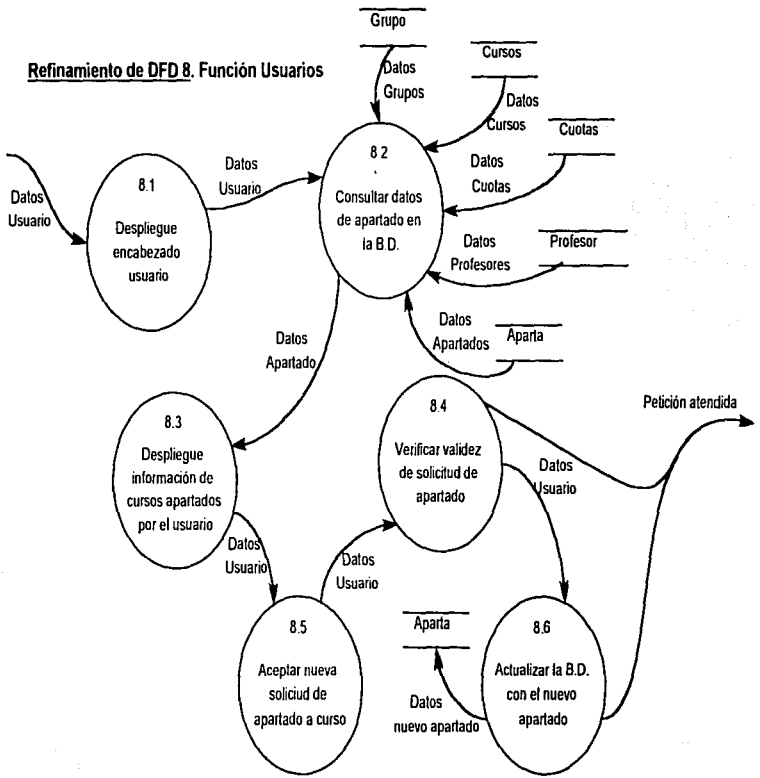
| **Proceso** | **2.1 Construcción del script para la interacción con la B.D.** |
|---|---|
| **Descripción** | Construye el archivo EXPECT que se encargará de la interacción entre el servidor WWW y la B.D. |
| **Restricciones** | Ninguna. |
| **Flujo datos entrada** | Petición de datos de los cursos. |
| **Flujo Datos Salida** | Status de la construcción del archivo, es decir: construcción exitosa; construcción fallida. |
| **Almacén de datos** | Ninguno. |
| **Términos locales** | Contenido del archivo (Corriendo desde el servidor WWW):<br>Inicio<br>Realiza conexión a B.D.<br>Espera Login<br>Manda Login<br>Espera password<br>Manda password<br>Espera Prompt<br>Manda instrucción (produce archivo de salida X)<br>Espera Prompt<br>Abre sesión ftp (a servidor WWW)<br>Espera Login (ftp)<br>Manda Login (ftp)<br>Espera password (ftp)<br>Manda password (ftp)<br>Espera prompt (ftp)<br>Coloca archivo de salida X en servidor WWW<br>Espera prompt (ftp)<br>Realiza desconexión (ftp)<br>Espera prompt<br>Realiza desconexión<br>Fin. |

**Refinamiento de DFD 3. Obtener clave usuario**



| **Proceso** | **3.3 Procesar archivo de salida del script de interacción con B.D.** |
|---|---|
| **Descripción** | Hace uso del archivo colocado por el script EXPECT en el servidor WWW; la información contenida en éste es recuperada en base a la correcta lectura de sus campos, cuyo formato corresponde a renglones y columnas, obedeciendo al modelo relacional de Base de Datos. Es un archivo plano. |
| **Restricciones** | Ninguna. |
| **Flujo datos entrada** | Bandera de construcción completa del script EXPECT. |
| **Flujo Datos Salida** | Status de la ejecución: completa ó fallida. |
| **Almacén de datos** | Ninguno |

| **Proceso** | **3.4 Eliminación del script y archivos temporales.** |
|---|---|
| **Descripción** | Elimina el script EXPECT así como todos los demás archivos depositados por éste en el servidor WWW (provenientes de la B.D.). |
| **Restricciones** | El borrado deberá ser puntual, es decir, borrará exclusivamente los archivos empleados en el presente flujo de datos; los demás deberán quedar totalmente intactos. |
| **Flujo datos entrada** | Datos del archivos de salida. |
| **Flujo Datos Salida** | Status eliminación (exitosa ó fallida). |
| **Almacén de datos** | Ninguno. |


Refinamiento de DFD 8. Función Usuarios
Grupo
Datos Grupos
Cursos
Datos Cursos
Cuotas
Datos Cuotas
Profesor
Datos Profesores
Aparta
Datos Apartados
Datos Usuario
8.1
Despliegue encabezado usuario
Datos Usuario
8.2
Consultar datos de apartado en la B.D.
Datos Apartado
8.3
Despliegue información de cursos apartados por el usuario
Datos Usuario
8.4
Verificar validez de solicitud de apartado
Petición atendida
Datos Usuario
Datos Usuario
8.5
Aceptar nueva soliciud de apartado a curso
Aparta
Datos nuevo apartado
8.6
Actualizar la B.D. con el nuevo apartado

| **Proceso** | **8.2 Consultar datos de apartado en la B.D.** |
|---|---|
| **Descripción** | Obtiene de la B.D. la información de los apartados realizados por el alumno, así como la información completa de cursos no aparatados, para posibilitar esta acción al usuario. |
| **Restricciones** | El alumno deberá recibir información exclusivamente de los cursos que NO ha apartado. |
| **Flujo datos entrada** | Datos del usuario; datos grupos; datos cursos; datos cuotas; datos profesores; datos apartados. |
| **Flujo Datos Salida** | Datos completos de cursos apartados y no apartados por el usuario. |
| **Almacén de datos** | READ Grupo (cve_grupo, aula, cupo).<br>READ Curso (cve_curso, nombre, horas, cuota).<br>READ Profesor (cve_profesor, nombre).<br>READ Aparta (alumno, curso, grupo).<br>READ Cuota (cve_cuota, cantidad). |

| **Proceso** | **8.4 Verificar validez de solicitud de apartado.** |
|---|---|
| **Descripción** | Verifica la validez de la solicitud de apartado; es decir, verificar que el usuario aparte un curso que NO haya apartado anteriormente. |
| **Restricciones** | A pesar de haberse validado con anterioridad, es necesario asegurarse que el usuario NO aparte cursos que ya haya apartado. |
| **Flujo datos entrada** | Datos del usuario. |
| **Flujo Datos Salida** | Datos del usuario con aprobación de apartado; solicitud atendida (para usuarios sin aprobación de apartado). |
| **Almacén de datos** | Ninguno. |

## Diseño de la Base de Datos.

**Definición de Base de Datos.**

Una Base de Datos (B.D.) es un conjunto de datos persistentes, sin redundancias perjudiciales o innecesarias, cuya finalidad es la de servir a una ó mas aplicaciones de la mejor manera posible. Entre sus características más importantes se presentan:

- Los datos son almacenados de modo que resulten independientes de los programas de aplicación que los utilizan.
- Se emplean métodos bien definidos para incluir nuevos datos y para modificar ó extraer los ya existentes.
- Es posible tener acceso a los datos en tiempo real, haciendo uso de la infraestructura de comunicaciones actual.

Las operaciones realizadas sobre una base de datos son controladas por un sistema denominado DBMS (DataBase Management System; Sistema Manejador de Base de Datos, por su traducción del inglés). Este sistema se encarga de hacer la traducción entre la perspectiva global de los datos en la B.D. y la perspectiva local esperada por cada programa de aplicación. Un DBMS es un conjunto de rutinas, funciones, métodos para accesar áreas de trabajo, almacenamiento y control requeridos para el manejo de información bajo el concepto de Base de Datos.

**Modelo Relacional de Bases de Datos.**

**Modelo.** Tratando de encontrar una representación más accesible de lo que es una B.D. se han desarrollado *modelos* que ilustran la lógica de su comportamiento, así como las restricciones y relaciones que existen entre sus datos. Un modelo es la representación que define los aspectos importantes acerca de la información que se necesita tener, y las relaciones entre dicha información; representa a la realidad tomando como base la forma en que los datos son almacenados en la computadora y cómo son mantenidas las relaciones entre ellos[6].

En el modelo relacional de B.D., la información se maneja como un conjunto de relaciones (tablas) que contienen atributos (columnas); su objetivo central es la elaboración de un diagrama denominado de **Entidad-Relación**, el cual representa los requerimientos de información de una organización y es independiente del hardware o software empleado en la implementación. Ver figura V.10.

Los componentes del Modelo Entidad-Relación son:

[6] Una buena bibliografía para estudiar con detenimiento los tipos de modelos de datos la podemos encontrar en: **KHOSHAFIAN,** Setrag et.al., "A Guide to Developing Client/Server SQL Applications", Editorial Morgan Kaufmann, Estados Unidos, 1996, Primera edición

- **Entidades.** Una entidad es un aspecto importante acerca del cual se necesita tener o conocer información; se identifica como una clase o categoría de cosas. Por ejemplo, durante el análisis efectuado se han distinguido las siguientes entidades: CURSOS, ALUMNOS, GRUPOS, CUOTAS, PROFESOR, etc.

- **Atributos.** Los atributos son rasgos o elementos que caracterizan y describen a una entidad para calificar, identificar, clasificar, cuantificar o expresar su estado. Por ejemplo, La entidad ALUMNOS se compone de los siguientes atributos: clave del alumno, nombre, dirección, teléfono, ciudad de residencia, etc.

- **Relaciones.** Una relación representa la asociación entre dos entidades o entre una entidad con sí misma. Por ejemplo, la relación entre las entidades PROFESOR y CURSO es: *cada curso puede ser impartido por uno y solamente un profesor; cada profesor puede ser asignado a uno o más cursos.*

La simbología empleada en el modelo E-R es la siguiente:


**Figura V.10.** Simbología empleada en un diagrama E-R.

**Tipos de relaciones.** Existen tres tipos de relaciones en un modelo E-R:

- **Relaciones muchos a uno (M:1).** Esta relación tiene el grado de ***uno o más*** en una dirección y el grado de ***uno y solamente uno*** en la otra. Por ejemplo, la relación entre las entidades CURSOS y CUOTAS: un curso debe tener ***una y solamente*** una cuota; una cuota puede ser asignada a ***uno o más*** cursos.

- **Relaciones muchos a muchos (M:M).** Esta relación tiene el grado de ***uno ó más*** en ambas direcciones. Por ejemplo, la relación entre las entidades ALUMNOS y CURSOS: un alumno puede estar registrado en ***uno o más*** cursos; un curso puede ser tomado por ***uno o más*** alumnos.

- **Relaciones uno a uno (1:1).** Esta relación tiene un grado de ***uno y solamente uno*** en ambas direcciones; son muy raras, y en ocasiones pueden ser la misma entidad. Nuestro modelo E-R no posee ninguna relación de este tipo.

El modelo relacional de Bases de Datos se ocupa de tres aspectos importantes: la estructura de la B.D., la integridad de la misma y su manipulación.

**Estructura de una Base de Datos Relacional.** Una Base de Datos relacional es aquella que es percibida por el usuario como una colección de relaciones o de tablas de dos dimensiones. Por ejemplo, la "tabla" ALUMNO contiene los siguientes datos:

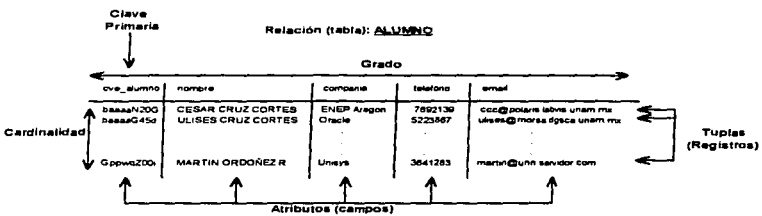



**Figura V.11.** Estructura de una "tabla" en el Modelo Relacional de Base de Datos.

En la figura anterior podemos observar el uso de una terminología propia de las Bases de Datos relacionales:

- Una **relación** corresponde a una tabla.
- Una **tupla** es una fila ó registro de una tabla.
- Un **atributo** es una columna o campo de una tabla.
- La **cardinalidad** de una tabla es el número de tuplas o filas.
- El **grado** de una relación está representado por el número de atributos ó columnas.
- Un **dominio** es una colección de valores, de los cuales uno o más atributos obtienen sus valores reales.
- Una **llave primaria** es un atributo ó grupo de atributos que identifican de manera ÚNICA a cada tupla en una tabla. Cada tabla debe tener una llave primaria.
- Una **llave foránea** es una columna o combinación de columnas en una tabla, que hacen referencia a una llave primaria en otra tabla. Por ejemplo, la tabla APARTA tiene como llave primaria a los atributos alumno, curso y grupo; el atributo alumno corresponde a la llave primaria de la tabla ALUMNO, por lo que dicha columna, es una llave foránea para APARTA.

**Integridad de los datos en una B.D. relacional.** Se refiere a la exactitud y consistencia de los datos[7]. Existen varias "reglas" para asegurar la integridad de datos en una B.D., éstos son:

| Tipo de regla | Descripción |
|---|---|
| **Integridad de Entidades** | Ninguna parte de la Llave Primaria puede ser NULA (vacía, carente de valor) |
| **Integridad Referencial** | Una llave foránea debe coincidir con un valor de una Llave Primaria. |
| **Integridad de Columnas** | Una columna debe contener sólo valores consistentes con el formato de tipo de dato definido para la columna. |
| **Integridad definida por el usuario** | Los datos almacenados en la B.D. deben cumplir con las reglas del negocio. |

**Tabla V.1.** Reglas para asegurar la integridad de datos en una Base de Datos relacional.

**Formas Normales.** La normalización es un proceso que minimiza la redundancia de los datos, evitando así problemas de integridad. Un dato sin normalizar debe considerarse redundante. La normalización ayuda a identificar entidades, relaciones y tablas faltantes en un modelo E-R. Existen tres Formas Normales:

***Primera Forma Normal.*** Los registros de las tablas no deben incluir grupos repetidos. Tomando como ejemplo una tabla de SARC en su estado inicial, empleamos la entidad ALUMNO, con los siguientes atributos (la llave primaria está en negritas):

| Cve_alumno | Nombre | Compañía | Telefono | Curso | Grupo | Cuota |
|---|---|---|---|---|---|---|
| **baaaG34N** | César Cruz | UNAM | 7692139 | S.O. UNIX | 001 | $1100.00 |
| | | | | Adm. UNIX | 002 | $1800.00 |
| | | | | DOS Básico | 003 | $780.00 |
| **bccc328H** | Ulises Cruz | Oracle | 5223867 | Adm. UNIX | 002 | $1800.00 |
| | | | | "C" Básico | 004 | $1000.00 |

**Tabla V.2.** Ejemplo de la tabla ALUMNO que incluye grupos repetidos.

Como es posible ver, los registros de la tabla incluyen grupos repetidos, así que los volvemos atómicos:

| Cve_alumno | Nombre | Compañía | Telefono | Curso | Grupo | Cuota |
|---|---|---|---|---|---|---|
| **baaaG34N** | César Cruz | UNAM | 7692139 | S.O. UNIX | 001 | $1100.00 |
| **baaaG34N** | César Cruz | UNAM | 7692139 | Adm. UNIX | 002 | $1800.00 |
| **baaaG34N** | César Cruz | UNAM | 7692139 | DOS Básico | 003 | $780.00 |
| **bccc328H** | Ulises Cruz | Oracle | 5223867 | Adm. UNIX | 002 | $1800.00 |
| **bccc328H** | Ulises Cruz | Oracle | 5223867 | "C" Básico | 004 | $1000.00 |

**Tabla V.3.** Ejemplo de la tabla ALUMNO, integrado por grupos atómicos.

La tabla anterior cumple con la Primera Forma Normal.

***Segunda Forma Normal.*** La tabla debe estar en Primera Forma Normal. Cada columna que no es llave debe ser dependiente de la llave primaria como un todo.

[7] Un dato es consistente cuando al existir varias copias del mismo registro TODAS ellas contienen la misma información; es decir, al consultar cualquiera de esos registros, se obtendrán los mismos datos.

En nuestro ejemplo, la entidad NO está en segunda forma normal, pues grupo, cuota y curso no dependen de la clave del alumno, así que se hace lo siguiente:

- Se determina cuáles columnas no llave no dependen de la llave primaria completa de la tabla. En nuestro ejemplo: grupo, cuota y curso.
- Se remueven esas columnas de la tabla base.
- Se crea una nueva tabla con esas columnas y la(s) columna(s) de la llave primaria de la cual dependen.

El resultado de aplicar los pasos anteriores sería (eliminando los grupos repetidos que se presentan en la tabla ALUMNO al remover las columnas que originaron la nueva tabla APARTA):

| Cve_alumno | Nombre | Compañía | Telefono |
|---|---|---|---|
| baaaG34N | César Cruz | UNAM | 7692139 |
| bccc328H | Ulises Cruz | Oracle | 5223867 |

**Tabla V.4.** Ejemplo de la tabla ALUMNO, todas sus columnas son dependientes de la llave primaria Cve_alumno, no presenta grupos repetidos.

| Cve_alumno | Cve_curso | Curso | Grupo | Cuota |
|---|---|---|---|---|
| baaaG34N | 001 | S.O. UNIX | 001 | $1100.00 |
| baaaG34N | 002 | Adm. UNIX | 002 | $1800.00 |
| baaaG34N | 003 | DOS Básico | 003 | $780.00 |
| bccc328H | 002 | Adm. UNIX | 002 | $1800.00 |
| bccc328H | 004 | "C" Básico | 004 | $1000.00 |

**Tabla V.5.** Ejemplo de la nueva tabla APARTA, derivada de la tabla ALUMNO.

La tablas anteriores cumplen con la Segunda Forma Normal.

**Tercera Forma Normal.** La tabla debe estar en Segunda Forma Normal. Una columna que no es llave primaria no debe ser funcionalmente dependiente de otra columna no llave. En nuestro ejemplo, la tabla ALUMNO cumple con esta Forma Normal, no así la entidad APARTA; debe hacerse lo siguiente:

- Determinar qué columnas son dependientes de otra columna no llave, en nuestro ejemplo, cuota y curso son dependientes de Cve_curso, pero no de grupo..
- Remover esas columnas de la tabla base.
- Crear una segunda tabla con esas columnas y con la columna no llave de la cual son dependientes.

El resultado de los pasos anteriores sería:

| Cve_alumno | Cve_curso | Grupo |
|---|---|---|
| baaaG34N | 001 | 001 |
| baaaG34N | 002 | 002 |
| baaaG34N | 003 | 003 |
| bcccJ28H | 002 | 002 |
| bcccJ28H | 004 | 004 |

**Tabla V.6.** Ejemplo de la tabla APARTA, cumple con la 3FN.

| Cve_curso | Curso | Cuota |
|---|---|---|
| 001 | S.O. UNIX | $1100.00 |
| 002 | Adm. UNIX | $1800.00 |
| 003 | DOS Básico | $780.00 |
| 004 | "C" Básico | $1000.00 |

**Tabla V.7.** Ejemplo de la nueva tabla CURSOS, derivada de la tabla APARTA.

Ahora, todas las tablas involucradas en nuestro ejemplo se encuentran **normalizadas**.

Siguiendo el mismo proceso antes descrito, todas las entidades involucradas de nuestro sistema fueron normalizadas, pasando de un estado inicial (sin normalizar) a un estado final (ya normalizado), a partir del cual se desarrolló el modelo E-R siguiente:

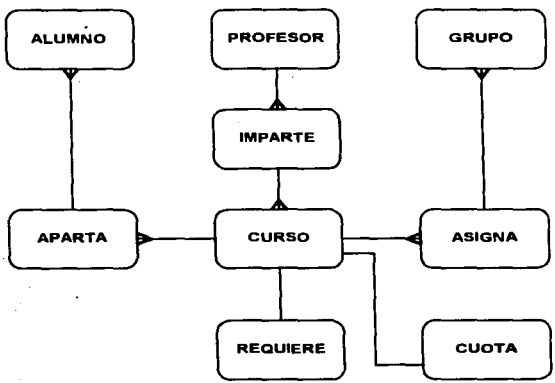



**Figura V.12a.** Modelo E-R de SARC.

A continuación presento los atributos de las entidades comprendidas en el modelo E-R de SARC. Las llaves primarias de cada entidad se encuentran escritas con **negritas:**

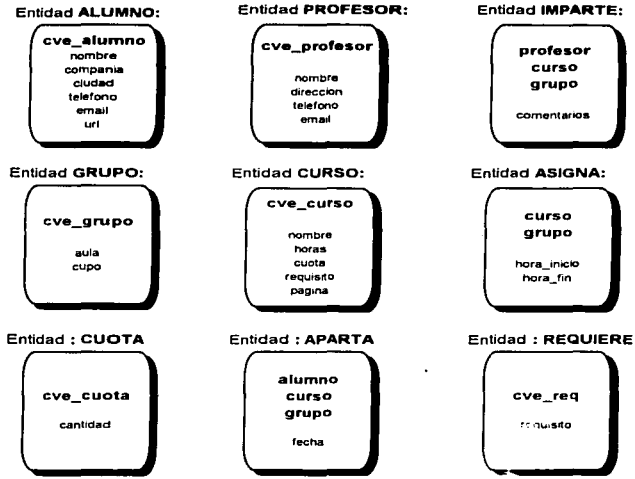



**Figura V.12b.** Definición de Atributos del modelo E-R de SARC

Este modelo es en el que habremos de basarnos para la implementación de nuestra Base de Datos.

**Manipulación de la Base de Datos.** Para la manipulación de los datos en cualquier B.D. relacional es necesario el uso de un lenguaje que permita entre otras cosas obtener información existente, incrementarla o eliminarla. El nombre de este lenguaje en el Diseño Relacional de B.D. es SQL (Structured Query Language; Lenguaje Estructurado de Consultas, de su traducción del inglés).

Las instrucciones SQL se apegan a un estándar; sin embargo, llegan a variar de la implementación de una compañía a otra. El SQL que emplearemos para la

manipulación de datos corresponde a All-Base de Hewlett-Packard, cuyas instrucciones principales son resumidas en el Apéndice A de este trabajo de tesis.

## Diseño de la interfaz del usuario (HTML).

La metodología de diseño que habrá de emplearse para la parte de la interfaz del usuario será **top-down** (pues conocemos a rasgos generales qué queremos del sistema).

En la interfaz del usuario se empleará una página base a partir de la cual se podrá navegar por SARC de acuerdo a las selecciones del usuario. Los documentos HTML desplegados –sean archivos HTML ó páginas generadas por los CGIs– conservarán un formato que den la impresión de consistencia y uniformidad en todo el sistema; para ello, como la programación se llevará a cabo en lenguaje C, se emplearán librerías que contengan **código reutilizable** por todos los programas que compongan a la aplicación.

SARC no contará con demasiadas páginas (en el módulo de usuario, no así en el de administración), pues se ha determinado que el exceso de ellas contribuiría a dificultar una tarea que se desea resulte sencilla y amena; además, la presencia de demasiados documentos HTML en el lado del usuario le daría elementos a éste para intentar corromper el sistema. Un mapa que comprende la totalidad de los módulos a desarrollarse es presentado a continuación:

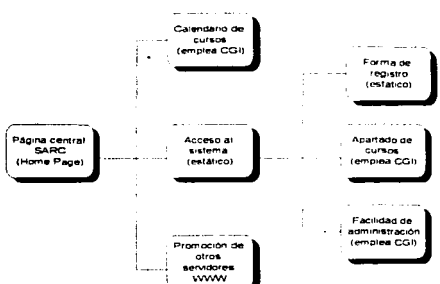



**Figura V.13.** Mapa de los módulos comprendidos en SARC.

## V.4 Construcción.

Esta etapa en el desarrollo de SARC, se dividirá en cuatro fases:

- Creación de la Base de Datos.
- Programación de las páginas HTML.
- Programación de las compuertas CGI.
- Programación de los programas SQL que manipulan los datos en la B.D.

### Creación de la Base de Datos.

Se ha creado un script para el sistema operativo UNIX que permite mediante un pequeño programa para All-Base de Hewlett Packard crear toda la infraestructura de la Base de Datos que habrá de emplearse en nuestra aplicación. Se han incluido comentarios (en la medida de lo posible) con la intención de facilitar el mantenimiento de SARC y/o su aprendizaje. Los códigos de esta etapa puede consultarse el apéndice C.

### Programación de las páginas HTML.

Para la programación de las páginas HTML se ha seguido un formato específico que le da a la totalidad de ellas una apariencia uniforme y con sentido de pertenencia a algo más grande. Se han incluido comentarios para facilitar su estudio. Los códigos de las páginas comprendidas en SARC NO se incluyen en el presente trabajo de tesis, pues únicamente contribuirían a hacerlo más grande de lo que ya es; si se desea obtener dichos códigos, puede hacerse a través del *paginador* empleado, mediante la opción "Save as", con lo que será posible guardar en la computadora cliente los documentos HTML deseados.

### Programación de las compuertas CGI.

Para la codificación de los CGIs se ha seguido un esquema modular muy interesante: todos los archivos **".c"** que integran la aplicación hacen uso de un solo archivo donde se concentra la totalidad de variables comunes a todos ellos (**"general.h"**), esto permite migrar la aplicación a cualquier Sistema Operativo UNIX, y a cualquier servidor HTTP y/o DBMS All-Base, ya que su programación es paramétrica y

fácil de entender. Se han incluido además, comentarios y una identación y marginación uniforme que hacen el estudio de estos códigos tarea fácil para cualquier programador con experiencia media en el lenguaje C. Los códigos fuente de TODOS los CGI de la aplicación han sido incluidos en el apéndice C.

### Programación de los scripts SQL.

Esta parte de la codificación son pequeños programas que son ejecutados por All-Base, cuyo contenido consiste en simples instrucciones que manipulan la información de la Base de Datos de SARC. Cualquier duda acerca de la sintaxis y función de All-Base, puede ser aclarada consultando el Apéndice A. Los códigos fuentes han sido incluidos en el Apéndice C de este trabajo.

## Implantación del Sistema.

La implantación del sistema representa el esfuerzo realizado para reunir las condiciones necesarias y colocar TODOS los componentes de la aplicación en las localidades adecuadas con la finalidad de poder ser ejecutada sin contratiempo alguno. Para la ejecución de SARC deberán reunirse las siguientes condiciones:

- SARC se ubica en el URL :
  **http://indy.aragon.unam.mx/~cruzcc/htdocs/SARC_general.html**

  Obviamente, las demás páginas que lo conforman poseen URLs distintos; sin embargo, el arriba expuesto es la página central o home page de la aplicación.

- El directorio /usr/people/estud/tesis/cruzcc/public_html/htdocs de la máquina indy.aragon.unam.mx, deberá contener TODOS los documentos HTML que componen la aplicación.

- El directorio /usr/local/etc/httpd/cgi-bin/cruzcc de indy.aragon.unam.mx contendrá exclusivamente los programas ejecutables de las compuertas CGI programadas.

- Deberá existir en /usr/people/estud/tesis/cruzcc (en indy.aragon.unam.mx) un directorio llamado publico, que deberá tener privilegios totales (lectura-escritura-ejecución).

- Los scripts SQL, y los de creación de la Base de Datos deberán encontrarse en el directorio /usr/people/estud/tesis1/bases de hp-730.aragon.unam.mx, este directorio deberá tener otorgados privilegios totales (lectura-escritura-ejecución).

- El directorio /usr/people/estud/tesis/cruzcc/public_html/images de la máquina indy.aragon.unam.mx deberá contener TODOS los gráficos que son invocados en cada uno de los documentos HTML de SARC.

Una vez reunidas las condiciones anteriores será posible emplear SARC sin problema alguno. El Apéndice B proporciona una guía del usuario que facilita en gran medida la utilización de la aplicación.

Por lo que respecta a la promoción de SARC, he contactado a los administradores de los servidores web http://informatica.aragon.unam.mx y http://morsa.dgsca.unam.mx, con los que he acordado la inclusión del URL de SARC con la finalidad de que el público que los visita tenga manera de accesarlo y hacer uso de él; obviamente el servidor http://indy.aragon.unam.mx (donde se encuentra SARC) incluirá ligas que promuevan a los servidores antes mencionados.

## Perspectivas de desarrollo.

SARC es un esfuerzo por demostrar la utilidad de HTML y CGI en el desarrollo de aplicaciones para Internet; desde mi punto de vista es útil e innovador, además hace uso de recursos actualmente subutilizados en ENEP Aragón.

Posiblemente una mejora inmediata que se puede hacer es unir los servidores web y de base de datos mediante algún truco de administración UNIX (como el levantamiento de un servicio NFS entre las dos máquinas), esto con la finalidad de reducir tiempos de respuesta que –aunque no son muy elevados– son siempre importantes para mantener la atención de las personas que hacen uso de él.

Es posible mejorarlo en muchos aspectos: apariencia, tecnología empleada, estructura de la información que ofrece, etc.; siempre obedeciendo a las necesidades de los usuarios y a las posibilidades y alcances de nuestra escuela. Sin embargo, actualmente se encuentra en un estado 100% funcional, haciendo uso de la misma tecnología que emplean empresas muy importantes a nivel mundial para ofrecer servicios semejantes.

SARC proporciona la modularidad necesaria en sus códigos para ser modificado; además, ofrece en la presente publicación una guía detallada de la manera en que fue ideado y desarrollado, bastará con que cualquier persona tome este libro para mejorar la aplicación o bien implantarla en algún otro servidor. Su mejora está limitada únicamente por la imaginación y capacidad de las personas que se animen a estudiarla.

# Conclusiones

Internet es el fenómeno tecnológico que mayor crecimiento y aceptación ha tenido en la historia de la humanidad, pues en apenas 30 años de existencia se ha convertido en uno de los principales medios de compartición e intercambio de información en el mundo. A pesar de que la razón de su creación haya sido el interés militar del gobierno de los E.U., actualmente existe como un nuevo medio de comunicación, más eficiente que los actuales en ciertos aspectos, pues es interactivo y permite la intervención de millones de personas alrededor de TODO el mundo en cuestión de segundos, sin fronteras ni restricciones.

La tecnología que da vida a la red global (el ruteo, el direccionamiento IP y la familia de protocolos TCP/IP) no es complicada, sinceramente es muy simple y fácil de entender; sin embargo, lo que llega a aportarle cierto grado de dificultad es la gran cantidad de equipos de cómputo de diferentes tipos que se comunican a través de ella.

La gran popularidad alcanzada por Internet se debe al conjunto de servicios que proporciona: e-mail, ftp, telnet, gopher, etc; pero hay uno de ellos que le ha dado un poder de penetración tal en la población mundial que ha llegado a ser confundido con Internet mismo (a pesar de ser sólo un servicio mas): World Wide Web, WWW. Este servicio permite agregar a los documentos que lo forman (comúnmente llamados páginas HTML), una increíble capacidad de comunicación sensorial, pues permiten la inclusión de audio, vídeo, gráficos y texto (entre otras cosas), convirtiéndolo en un nuevo medio de comunicación con serias ventajas sobre los demás medios, pues es masivo, da libertad de expresión, es barato y cualquier persona con acceso a él puede exponer la idea que le venga en mente.

La capacidad de comunicación existente en WWW ocasiona que la generación profesional de documentos HTML (que es su unidad funcional) sea un asunto delicado, pues es necesario seguir una metodología definida para su desarrollo; así, será posible tener la certeza de crear documentos con el grado de penetración deseada en la audiencia, prevista desde una etapa conceptual.

Actualmente, existe un número inimaginable de páginas web al alcance de cualquier persona con acceso a Internet; sin embargo, sólo un grupo de ellas posee elementos que las hacen sobresalir de las demás: cuentan con contenidos dinámicos; es decir, realizando una sola invocación, es posible obtener documentos que varíen en contenido de acuerdo a selecciones del usuario; ésto se logra haciendo uso de la tecnología CGI.

Un CGI es un programa escrito en cualquier lenguaje de programación (en éste trabajo se concluye que el más indicado para su desarrollo es el lenguaje C), que funciona a manera de una compuerta (interfase) entre un documento HTML y una aplicación ajena a WWW; es un programa ejecutable cuya función es la de formatear y/o traducir cualquier "salida" generada por la aplicación ajena al web, adecuándola a la sintaxis correspondiente a un documento HTML, logrando así la comunicación (transparente al usuario) entre la tecnología WWW y cualquier otra que se desee integrar a Internet. De esta manera es posible encontrar en el Web" sistemas de apartado de viajes, reservaciones hoteleras, pago de servicios, etc, todos ellos haciendo uso de HTML y CGI.

Considero que el presente trabajo de tesis cumple plenamente con los objetivos generales expuestos en un inicio, pues tras identificar, conocer y dominar los diferentes detalles comprendidos en el lenguaje HTML y la tecnología CGI, proporciona las bases suficientes para realizar la integración de una Base de Datos a WWW, materializando un Sistema de Apartado Remoto a Cursos (SARC), el cual estuvo inspirado en el tipo de aplicaciones que es posible encontrar en Internet y que hacen uso de las tecnologías mencionadas.

Además, a pesar de que este trabajo se hubiese encontrado en una etapa de desarrollo, contiene información que ha servido de apoyo bibliográfico al desarrollo de otros trabajos de investigación similares, síntoma inequívoco de la riqueza y validez de la información que contiene en sus páginas.

Desde mi punto de vista, la tecnología CGI ha llegado al máximo de su crecimiento; como todas las cosas, posee ventajas y desventajas, pero de manera general es útil y así lo demuestran la gran cantidad de aplicaciones en Internet desarrolladas con ella. Considero que la tecnología CGI no será desplazada tan rápidamente como lo auguran ciertas personas; creo sinceramente que las nuevas tecnologías que han aparecido (como JavaScript, Java, Visual J++, ActiveX, etc.) se apoyarán en CGI para posicionarse en la preferencia de los usuarios de Internet, y así dar a conocer las grandes ventajas que ofrecen. En unos cuanto años será posible observar –sin duda alguna– una infraestructura más sólida para el desarrollo de aplicaciones en Internet, posiblemente sin la presencia física de compuertas ó CGIs, pero seguramente derivada de éstas.

La validez e importancia del presente trabajo de tesis es una realidad, pues aporta a la comunidad informática un documento útil para un claro entendimiento de los orígenes, evolución y alcance de la red global; así como una guía detallada del desarrollo de una aplicación real, totalmente funcional, ubicada dentro del marco de la tecnología HTML-CGI e Internet.

Durante el desarrollo de esta obra me fue posible comprender las razones que han hecho de Internet el fenómeno tecnológico, social y de comunicación más importante en la historia de la humanidad. Esta tecnología en conjunto continuará creciendo, por lo que conocer los conceptos básicos de su funcionamiento resulta vital para el profesional de computación que desee –en un futuro– implementar soluciones en este ámbito, obviamente haciendo uso de los conocimientos que posea para aplicarlos, o bien, para asimilar de una manera más rápida las nuevas estrategias y tecnologías que

seguramente seguirán apareciendo, a una velocidad tan vertiginosa como el mismo crecimiento de la red.

Finalmente, es importante destacar que el desarrollo de aplicaciones en Internet no es todavía algo habitual entre la comunidad informática mundial, pues existen ciertos aspectos que impiden su crecimiento franco: las limitantes de velocidad y conexión hacia Internet por parte del usuario final; el limitado ancho de banda existente actualmente; los enormes huecos de seguridad existentes en las transacciones de Internet, etc. Sin embargo, es oportuno señalar que la tecnología en informática y telecomunicaciones ha crecido a pasos agigantados, y seguramente resolverá de manera satisfactoria estos obstáculos con prontitud. Nuestra responsabilidad como ingenieros es mantenernos actualizados para poder actuar consecuentemente a las tendencias del mundo informático, para en base a ellas ofrecer nuestra capacidad para el desarrollo de soluciones que puedan poner en alto nuestro grado de estudios, nuestra escuela, y –principalmente– a nuestro país.

**César Cruz Cortés.**

# A

# Referencia Rápida: EXPECT e ISQL(All-Base)

El objetivo principal de este apéndice es proporcionar una guía completa y concisa, que permita al lector entender, sin dificultad alguna, los scripts (programas) incluidos en el Apéndice C: Códigos Fuentes de SARC. Esta Referencia Rápida" incluye una visión general de dos de las herramientas –muy novedosas por cierto– empleadas en la construcción de la aplicación demostrativa del presente trabajo de tesis.

## EXPECT.

Expect[1] es un programa cuyo objetivo es controlar y automatizar aplicaciones interactivas, permite –mediante sentencias sencillas de envio/espera– eliminar la intervención de un usuario para la operación de alguna aplicación interactiva. Por ejemplo, un script de Expect permitiría eliminar la acción de un usuario para transmitir mediante ftp un conjunto de archivos; realizar ejecuciones de aplicaciones remotas a través de telnet;" o bien, mandar a ejecución un programa que requiera parámetros proporcionados necesariamente desde el teclado, entre otras cosas.

Imaginemos que tenemos una aplicación que monitorea la integridad del sistema de archivos de un disco duro en un sistema operativo UNIX. Si se llegara a presentar alguna anormalidad tal que afectara el sistema de archivos, ésta sería detectada por la aplicación de monitoreo, así que una solución inmediata al problema sería la ejecución del comando UNIX **fsck**, pero ... ¡Necesitamos a alguien que ejecute el programa, pues éste solicita al usuario un conjunto de parámetros y opciones que necesariamente deben ser insertadas desde el teclado! ¿Llamaríamos al administrador del equipo para la ejecución del comando? ¿Cuánto tiempo tardaría en llegar al lugar en caso de encontrarse alejado de éste? ¿Qué pasaría si el sistema de archivos con problemas es de misión crítica en el negocio?. La solución sería una de las siguientes:

[1] Expect fue creado por Don Libes, investigador del Instituto Nacional de Estándares y Tecnología de los Estados Unidos (NIST; National Institute of Standards and Technology, por su traducción al inglés) en 1990.

- Contar con varias personas que se dedicaran a operar de tiempo completo la aplicación de monitoreo (las 24 horas del dia, los 365 días del año), la cual podría no detectar falla alguna en años. Horas hombre totalmente desperdiciadas.

- Un programa que interactuara de manera eficiente entre la aplicación de monitoreo y el comando pertinente para la corrección del problema (en el ejemplo, fsck). Esta capacidad es proporcionada eficientemente por **Expect**.

No es necesario confrontar las ventajas y desventajas de ambas opciones, obviamente la mejor de ellas es la segunda: el uso de un programa de apoyo (Expect) que automatice la interacción existente en la ejecución del comando mencionado.

Expect es empleado en un gran número de empresas internacionales para resolver problemas semejantes al expuesto anteriormente, entre ellas se encuentran Silicon Graphics, 3Com, Hewlett-Packard, IBM, Xerox, Sun, Tektronix, Data General, AT&T, entre otras. Su penetración en la industria ha sido rápida y muy bien vista, esta rapidez de adopción probablemente se deba a que Expect es totalmente gratuito, de libre distribución[2].

La instalación de Expect queda totalmente fuera del alcance de este apéndice; sin embargo, la distribución hecha por el NIST no sólo incluye el programa, sino también los códigos fuente, ejemplos, y una guía extensa que expone los procedimientos necesarios para su instalación y puesta en marcha.

## Estructura básica de un script EXPECT.

Los scripts de Expect son archivos planos comunes, se componen únicamente de dos partes:

- Una línea de encabezado que hace referencia a la "uta física" donde se encuentra instalado el programa; dicho directorio será el que Expect tomará como base para la búsqueda de archivos auxiliares, útiles en la ejecución de cualquier script.

- El cuerpo del programa.

La filosofía que sigue cualquier script Expect en la ejecución de sus instrucciones, obedece –de forma general– al siguiente algoritmo:

1. **Ejecuta** el programa sobre el que se actuará.

[2] Puede encontrarse la más reciente versión de Expect en **ftp://ftp.cme.nist.gov**. Don Libes y el NIST no restringen el uso del programa a persona ó institución alguna, la única observación especial para su uso, es el deslindamiento de responsabilidades que el NIST y Don Libes hacen en caso de presentarse cualquier daño ó problema derivado del uso de la versión original de Expect.

2. **Espera** alguna cadena de caracteres que indique la consecución de la instrucción anterior.

3. **Manda** una cadena de caracteres en respuesta a la recibida en el punto 2.

4. En caso de existir más instrucciones que se deseen incluir en el script Expect, pasar al punto 2; de lo contrario, pasar al punto 5.

5. **Espera** alguna cadena de caracteres que indique la consecución de la instrucción anterior.

6. Fin del programa.

En el algoritmo anterior es posible apreciar la naturaleza envío/espera de Expect; es decir; una vez ejecutado el comando ó aplicación deseado, Expect cae en una tarea repetitiva de envío de una cadena de caracteres (que puede ser un comando), y espera de otra cadena de caracteres; una vez recibida esta respuesta, Expect actuará en consecuencia, mandando una cadena adecuada y esperando una respuesta predefinida.

## Comandos básicos de EXPECT.

Expect es un lenguaje basado en scripts muy robusto; posee un número regular de comandos, y varias sentencias de control que le dan la solidez que lo ha hecho famoso internacionalmente. Exponer la totalidad de las instrucciones que lo componen, se encuentra fuera del alcance de este apéndice[3]; sin embargo, a continuación se describen sus tres instrucciones principales, las cuales en conjunto, pueden controlar sin problema alguno la interacción con un programa, automatizando su función.

### Comando spawn (engendrar).

Este comando permite la ejecución ó invocación de otro programa. Un programa corriendo en memoria se conoce como proceso; el comando spawn permite "engendrar" un proceso hijo que ejecute un programa ó aplicación dados. Su sintaxis es:

**spawn** <programa> { argumentos }

donde:

| | |
|---|---|
| <programa> | es el nombre del programa ó aplicación a ejecutar, |
| { argumentos } | puede o no presentarse, corresponde al conjunto de argumentos propios del programa ejecutado. |

[3] Una referencia completa de EXPECT puede encontrarse en: LIBES, Don, "Exploring Expect", Editorial O'Reilly & Associates, Estados Unidos, 1995, 1a. edición.

Por ejemplo, en la sentencia:

**spawn** telnet 132.248.145.5

el comando spawn "engendra" un proceso hijo que ejecutará el comando telnet, cuyo argumento es una dirección IP (132.248.145.5).

**Comando expect (espera).**

Como su nombre lo indica, este comando espera la presencia de una cadena de caracteres predefinida; cuando ésta se presenta, el flujo del programa pasa a la siguiente instrucción. Su sintaxis es:

**expect** "<cadena>"

donde:

<cadena> Es una cadena de caracteres cualquiera.

Por default, esta instrucción espera la cadena de caracteres citada de la entrada estándar (el teclado); pero si aún existe algún proceso hijo "engendrado" por alguna ejecución anterior del comando spawn, expect esperará dicha cadena a partir del flujo de datos producido por dicho proceso.

Cabe la posibilidad de que el comando expect –debido a un error en la planeación del script– espere una cadena que nunca se presentará; por ejemplo, "password" en lugar de "Password", "Login" en lugar de "login", etc. Cuando esto ocurra, el comando expect nunca pasará de tal línea, por lo que es recomendable establecer un tiempo máximo de espera antes de tomar una acción alterna; esto se realiza con la directiva **set timeout**, que posee la siguiente sintaxis:

**set timeout** <segundos>

donde:

<segundos> es cualquier número entero que representa los segundos que se aguardarán antes de tomar una acción alterna, en caso de no recibirse alguna cadena contemplada.

Por ejemplo, en el script:

```
set timeout 30
        spawn telnet 132.248.145.5
        expect {
                timeout {exit}
                "ogin:"
                }
```

es posible apreciar que en caso de que el proceso que ejecuta el comando telnet (a través de spawn) no regresara la cadena "ogin:" en 30 segundos, la acción alterna a tomar salir inmediatamente del programa (con el comando **exit**).

**Comando send (envía).**

Este comando envia una cadena de caracteres cualquiera, por default, a la salida estándar (el monitor), aunque en caso de existir todavía algún proceso "hijo" engendrado por el comando spawn, dicha cadena será enviada al flujo de entrada del proceso engendrado. Su sintaxis es:

**send** "<cadena>"

donde:

<cadena> Es una cadena de caracteres cualquiera.

Es importante destacar que en caso de requerirse la ejecución de algún comando ó programa con send, el equivalente a la tecla "Enter", será la sentencia de escape "\r", así, la línea:

**send** "fcsk \r"

Equivale a escribir desde el teclado la cadena **fcsk** y un "Enter" para que dicho comando sea ejecutado.

A continuación se expone un script Expect que automatiza una sesión telnet, cuyo propósito es conocer la hora que posee un servidor remoto:

```
#!/usr/disco2/people/ulises/cesarin/expect-5.19/expect -f

set timeout 30
spawn telnet 132.248.145.3
expect {
        timeout {exit}
        "ogin:"
        }
send "ccc"
expect "assword"
send "mipassword\r"
expect "TERM"
send "vt100\r"
expect "ccc"
send "date\r"
expect "ccc"
send "exit\r"
exit
```

Analicemos línea a línea el script anterior:

- Primeramente es posible apreciar el encabezado del script; es decir, la ruta física del servidor en donde se encuentra el programa Expect: #!/usr/disco2/people/ulises/cesarin/expect-5.19/expect -f

- En seguida aparece la directiva set timeout 30, la cual nos indica que el tiempo máximo de espera de una cadena de caracteres por parte del comando expect será de 30 segundos.

- La línea spawn telnet 132.248.145.3, ejecuta telnet a través del comando spawn, a la dirección IP especificada.

- La primer directiva expect, esperará la cadena "ogin:", pues se está considerando que la maquina 132.248.145.3 solicitará un login para accesarla. No se escribe la cadena "login" completa, para evitar que una variación en la capitalización de la primera letra evite la ejecución del script. En caso de pasar 30 segundos sin recibir esta cadena, el programa es abortado inmediatamente.

- Una vez recibida la cadena "ogin:", el script manda –con el comando send– el login correspondiente, es decir la cadena "tcc", finalizándola con la secuencia de escape "\r", que equivale a oprimir la tecla Enter.

- Esta secuencia de espera-envío se repite para "assword:" "mipassword\r", y "TERM" "vt100\r". Habiendo pasado ésto, nos encontramos ya en una sesión remota telnet en el host 132.248.145.3.

- El siguiente comando expect espera la cadena "tcc", que representa el prompt de la cuenta UNIX accesada; al recibirlo, sabremos que ya es posible la introducción de cualquier comando, el cual se ejecutará en el host remoto. El comando a ejecutar es "date" (que es invocado con el comando send, finalizado con la secuencia de escape "\r").

- Una vez más, la cadena "tcc" indicará el prompt de la cuenta UNIX, y por ende, la finalización del comando ejecutado y la posibilidad de introducir nuevas instrucciones.

- Finalmente, aparecen dos cadenas "exit", una de ellas complementada con "\r"; la primera para finalizar la sesión telnet abierta, y la segunda para finalizar el script Expect que se está ejecutando.

La ejecución del script anterior aparece directamente en la pantalla: se desplegarán todas las cadenas de los comandos "send", así como los mensajes generados por las diferentes aplicaciones ejecutadas en él. Es responsabilidad del desarrollador canalizar tales salidas en flujos que convengan a la aplicación que hace uso de Expect. Por ejemplo, para eliminar el despliegue antes citado, bastará con redireccionarlo a un archivo (`archivo.expect > salida`); el tratamiento que se le dé a dichos flujos de datos, así como a los resultados generados en las ejecuciones de los diferentes comandos invocados, serán responsabilidad total del desarrollador, en

obediencia directa a las disposiciones establecidas durante la fase de análisis y diseño de la aplicación.

Es importante destacar que los comandos antes expuestos NO son todos los que proporciona Expect; sin embargo, con los aquí explicados basta y sobra para automatizar procesos interactivos, es más, en la aplicación desarrollada en el presente trabajo de tesis, son todos los comandos que se emplean. En caso de necesitar una información más profunda acerca de Expect, es recomendable consultar la bibliografía antes citada.

# ISQL (All-Base).

ISQL es un acrónimo que significa Interactive Structured Query Language (Lengauje de Búsquedas Estructuradas e Interactivas, por su traducción al español). Este programa es proporcionado por TODAS las bases de datos en el mercado (Sybase, Oracle, Informix, **All-Base**, etc.) como parte de su producto. Un isql de Sybase NO puede explotar información de una B.D. Oracle, así como un isql Oracle no puede consultar una B.D. All-Base; es decir, cada programa isql está especializado para explotar bases de datos especificas. Este apéndice expone información acerca del programa ISQL de All-Base[4], que fue la Base de Datos empleada para el desarrollo de la aplicación demostrativa del presente trabajo de tesis.

Con ISQL, es posible definir y accesar datos en un ambiente de Base de Datos relacional All-Base: permite la introducción de comandos diversos para crear y mantener ambientes de Bases de Datos; instalar módulos, e incluso ejecutar conjuntos de instrucciones contenidos en archivos planos (a manera de scripts).

## Trabajando con ISQL (All-Base).

### Invocación de ISQL.

Para invocar ISQL desde el sistema operativo[5], bastará con teclear desde el prompt del sistema:

```
$ isql
```

Inmediatamente después, aparecerá un nuevo prompt (`isql=>`) que nos indica que hemos ingresado al programa.

[4] All-Base es el ***DBMS*** implementado por Hewlett-Packard, no tiene gran presencia en el mercado (si es que lo comparamos con los productos de Sybase, Oracle ó Informix), pero es utilizado en un gran número de empresas privadas, gobierno, y universidades de todo el mundo.

[5] El sistema operativo en que se encuentra el programa ISQL empleado es UNIX (HP-UX de Hewlett-Packard).

**Salida de ISQL.**

Para salir del programa, será necesario teclear:

```
isql=> end;
```

Al ejecutarse la instrucción anterior, abandonamos el programa y nos encontramos nuevamente en la línea de comandos UNIX (lo que es posible apreciar gracias a la presencia del prompt de sistema : $).

Nótese que al final de la línea ejecutada se tecleó además un símbolo de punto y coma (;), lo que indica a ISQL la finalización de la instrucción y la orden de ejecutar inmediatamente el comando. En caso de no teclearse dicho símbolo, al oprimir Enter no se ejecutará la instrucción, sino que se abrirá un nuevo renglón que complemente el comando de la ó las líneas anteriores. Por ejemplo, los siguientes renglones corresponden a una sola instrucción:

```
isql=> select A.nombre,C.nombre
> from Alumno A,Curso C,Aparta AP
> where A.cve_alumno = 'CUCC730624'
>    and A.cve_alumno = AP.alumno
>    and AP.curso = C.cve_curso;
```

El texto escrito durante la ejecución de ISQL, no es sensitivo; es decir, no hace distinción alguna entre mayúsculas y minúsculas.

**Iniciando una sesión DBE.**

El DBMS All-Base requiere (a diferencia de otros productos) la invocación de un espacio reservado de recursos del sistema (memoria, disco, etc.), llamado DBE (DataBase Environment; Ambiente de Base de Datos, por su traducción al español). El DBE debe invocarse (ser abierto) una sola vez, antes de la introducción de cualquier comando.

Una vez abierto un DBE, es posible crear en él Bases de Datos, tablas, vistas, etc., o bien, realizar la manipulación de información presente en éste. Es imposible realizar alguna afectación de datos si no se ha abierto anteriormente un DBE.

La sintaxis para la invocación de un DBE es:

```
isql=> connect to '<nombre DBE>';
```

donde:

`<nombre DBE>` Es el nombre de un DBE existente.

Por ejemplo, la sentencia:

```
isql=> connect to 'SARC';
```

invoca un DBE llamado SARC. Después de esta operación es posible crear ó eliminar cualquier elemento ó información existente en 'SARC'. En caso de que el DBE referenciado no exista, es necesario crearlo, para lo cual se emplea la sentencia:

```
isql=> start dbe 'nuevo_dbe' new;
```

donde es creado un nuevo DBE llamado "nuevo_dbe", el cual queda abierto una vez creado; es decir, no será necesario emplear la instrucción "connect to ...".

**Uso de comandos SQL en un DBE All-Base.**

Obviamente, la cantidad de comandos que es posible emplear al hacer uso del producto de Base de Datos de HP (All-Base) es muy grande; es más, exponerlo representa un trabajo exhaustivo, meritorio por sí mismo de un tema de tesis. Además, la intención de este apéndice es exponer las características más generales que hacen diferente a All-Base con respecto a los DBMS más comunes en el mercado; cualquier otra explicación (como significado de los queries, uso de SQL, conceptos de integridad referencial, etc.), se encuentran completamente fuera del alcance de este anexo[6].

All-Base, al igual que los demás DBMS posee comandos para la explotación y manipulación de información contenida en cada DBE, apegándose al estándar SQL. Resulta exactamente lo mismo construir un "query" en All-Base ó en cualquier otro DBMS, pues su significado es consistente no importando el producto utilizado; por ejemplo, en All-Base es posible emplear:

- Select.
- Update.
- Insert.
- Delete.

Su uso obedece al estándar SQL; sus sintaxis NO serán expuestas en este apéndice.

**Creación de scripts con sentencias ISQL (All-Base).**

ISQL (All-Base) permite agrupar un conjunto de instrucciones válidas en un archivo plano, a manera de script; de esta manera es posible ejecutar mediante una sola llamada (la del nombre del archivo) una a una las sentencias implicadas. Los comandos siguientes son el contenido del archivo **prueba.sql**:

[6] Para mayor información acerca de All-Base puede consultarse: **AUTORES VARIOS**, "AllBase/ISQL Reference Manual", Editorial Hewlett-Pachard Press, Estados Unidos, 1991, Quinta edición.

```
connect to 'SARC';
select * from alumno
where cve_alumno = 'CUCC730624';
commit work;
exit;
```

Para ejecutarlo, bastará con teclear desde la línea de comandos:

```
$ isql < prueba.sql
```

o bien, desde el prompt isql:

```
isql=> start prueba.sql;
```

La salida producida por ambas instrucciones será desplegada en la pantalla; sin embargo, para que dicha salida sea formateada y guardada en un archivo plano, el script deberá contener las siguientes instrucciones:

```
connect to 'SIRC';
unload to external salida.sal
from "select * from alumno
where cve_alumno = 'CUCC730624'"
desc 10 50 30 20 20 30 50;
ROLLBACK WORK RELEASE;
EXIT;
```

Es posible apreciar que el archivo plano en el que se guardará la salida del comando es **salida.sal**; la instrucción a ejecutar se encuentra entre comillas (""); existe una serie de números antecedidos de la palabra **desc**, éstos números representan la longitud de los campos en que se guardarán –en el archivo plano– cada una de las columnas regresadas por el comando.

**Parámetros en los comandos SQL.**

All-Base posee una cualidad muy interesante: permite parámetros en la ejecución de sus comandos; por ejemplo:

```
isql=> select &1 from &2;
```

En esta sentencia, no se determina el nombre del campo que se desea ver, ni la tabla que se consultará para obtener información; realmente sus nombres podrían ser cualquiera (siempre y cuando fueran válidos); así, su ejecución en isql sería la siguiente:

```
isql=> select &1 from &2;
Value for parameter &1> *;
Value for parameter &2> Alumno;
```

Si se deseara incluir el comando anterior (parametrizado) en un script ISQL, tendríamos lo siguiente:

```
connect to 'SIRC';
unload to external salida.sal
from "select &1 from &2"
desc 10 50 30 20 20 30 50;
ROLLBACK WORK RELEASE;
EXIT;
```

Para poder ejecutarlo, es necesario realizar su invocación desde el prompt de isql; si se considera que el script se llama prueba.sql, tendríamos lo siguiente:

```
isql=> start prueba.sql(*,alumno);
```

Hasta el momento se ha expuesto sólo una pequeña parte de ISQL (All-Base), ésta herramienta contempla muchos aspectos extras, los cuales sería difícil exponer en un apéndice. En caso de desear profundizar más en el estudio de este DBMS, es recomendable consultar la bibliografía citada.

# B

# Manual de Usuario (SARC)

El presente apéndice tiene el propósito de ofrecer al usuario de SARC una herramienta escrita que le auxilie a manipular la aplicación sin problema alguno; ésto NO significa que sea difícil de manejar, más bien se intenta ofrecer una documentación completa del sistema: en el capítulo 5 se presentó la sección técnica de SARC, éste apéndice representa la sección de operación del mismo.

Al igual que todos los sites en WWW, SARC posee una página principal ó Home Page a partir del cual es posible ejecutar cualquiera de las opciones que ofrece. Su URL es:

**http://indy.aragon.unam.mx/~cruzcc/htdocs/SARC_general.html**

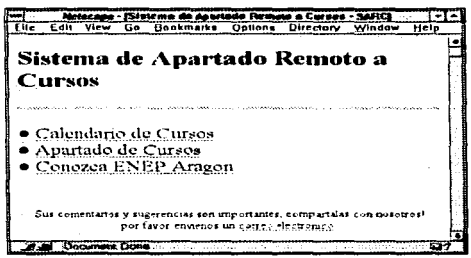



**Figura B.1.** Home Page de SARC.

El Home Page de SARC cuenta con tres ligas de hipertexto:

- **Calendario de Cursos**. Nos expone el calendario de cursos para el periodo actual. Esta hiperliga es expuesta a detalle en la siguiente sección.
- **Apartado de Cursos**. Representa la entrada a la aplicación SARC.

- **Conozca ENEP Aragón.** Es una liga hipertexto que nos lleva al servidor que contiene la información general de ENEP Aragón; es una liga externa, totalmente ajena a SARC, por lo que no será explicada en el presente apéndice.

## Calendario de Cursos.

Está página es llamada desde el Home Page de SARC, nos proporciona la información correspondiente a los cursos que han sido programados para el periodo actual. La vigencia de la información que nos presenta está totalmente garantizada, pues ésta es consultada en tiempo real desde la Base de Datos de la aplicación.

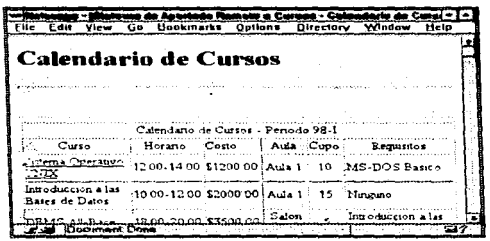



**Figura B.2.** Calendario de Cursos.

En esta página la información esta claramente organizada en una tabla de contenidos. Algunos cursos de la tabla pueden ser hiperligas, las cuales, en caso de presentarse, nos conducirán a una nueva página HTML (estática) con una explicación detallada del curso en cuestión.

Cabe destacar que TODAS las páginas HTML que componen a SARC incluyen en su parte inferior, ligas hipertexto que permiten (sin importar el lugar en que uno se encuentre) dirigirse al Home Page de la aplicación; además, es posible emplear en cualquier momento los recursos propios del navegador que se esté utilizando para desplazarse hacia la página anterior (o la siguiente, según sea el caso).

## Apartado de Cursos.

Esta opción es invocada desde el Home Page de SARC; representa la entrada formal al Sistema de Apartado Remoto a Cursos. Cualquier persona que intente hacer uso de la aplicación, deberá contar con una palabra clave (password), así que la primera acción de SARC, es solicitarla:

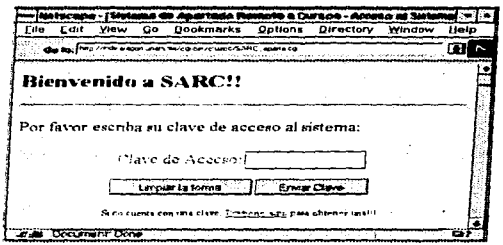



**Figura B.3.** Solicitud de la clave de acceso.

Es posible que se presenten usuarios ajenos (que no cuenten con una palabra clave válida); si éstos intentaran accesar la aplicación, la carencia de un clave correcta evitará su irrupción a SARC:

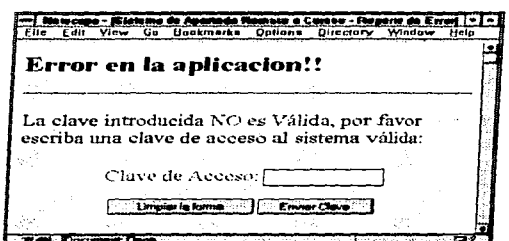



**Figura B.4.** Mensaje de error producido por la introducción de una clave de acceso NO válida.

En caso de no contar con alguna clave de acceso válida para hacer uso de las facilidades de SARC, es posible adquirir una a través del llenado de una forma de registro:

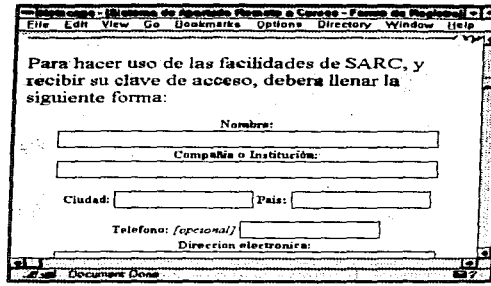



**Figura B.4.** Forma de registro para la obtención de una clave de acceso a SARC

Esta forma de registro es invocada desde la página de acceso al sistema, basta hacer click" sobre la hiperliga pertinente para situarnos en ella. Una vez que se ha llenado esta forma de registro, y tras haberla mandado al servidor, SARC asigna automáticamente una clave única, ésta deberá ser guardada confidencialmente, pues representa la llave de acceso a las facilidades de apartado de cursos de ENEP Aragón.

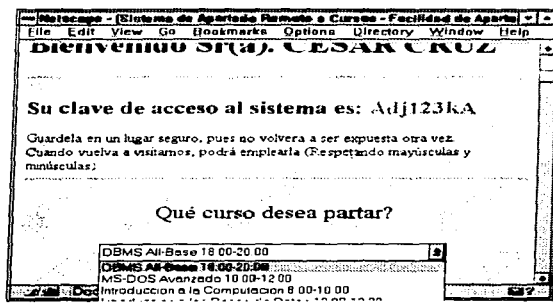



**Figura B.5.** Página desplegada al usuario, donde se informa de la clave de acceso recién creada y asignada.

En el documento HTML anterior es posible apreciar una sección donde se informa al usuario acerca de su clave de acceso. Es importante tener en cuenta que ésta será la única vez que aparecerá, así que es necesario anotarla en un lugar seguro. A partir del siguiente acceso, será posible introducir la clave asignada –al momento en que se solicite– para hacer uso del sistema de apartado.

En esta página es posible realizar el apartado de cursos, basta con posicionarnos en la lista desplegable (ubicada debajo de la leyenda Qué curso desea apartar?), donde aparecen TODOS los cursos disponibles. No es necesario volver a visitar el Calendario de Cursos para efectuar un apartado, pues aquí se ofrece una información general de cada uno de ellos (de los que se encuentren disponibles).

Una vez seleccionado un curso, basta con oprimir el botón con la leyenda 'Apartar AHORA!' y listo!!! se ha realizado el apartado deseado. Lo que resta será acudir a la Coordinación del Centro de Cómputo de ENEP Aragón (dentro de las 24 horas siguientes) para formalizar la inscripción al curso.

Imaginemos que se desea apartar otro curso. Ya que se haya introducido la clave de acceso al sistema correctamente, se entrará a la página descrita en el párrafo anterior. Para apartar otro curso se sigue el mismo procedimiento antes descrito. Es imposible realizar el apartado repetido de un curso pues en la lista desplegable sólo aparecerán aquellos cursos vigentes en los que el usuario actual NO haya hecho ya un apartado anterior. El número máximo de apartados es dos.

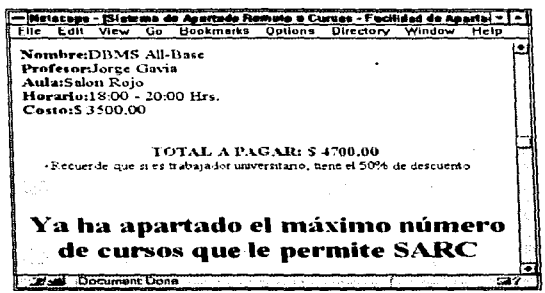



**Figura B.6.** Mensaje desplegado al haber apartado ya dos cursos.

Cuando se haya alcanzado el número máximo posible de apartados (2), la lista desplegable ya no aparecerá; en su lugar, será posible ver una leyenda que indica al usuario la situación en la que se encuentra, así como el monto que deberá pagar al inscribirse en los cursos que ya ha apartado.

En caso de que se presentara algún error súbito en la aplicación (posiblemente producido por alguna falla en las comunicaciones, o en algún servidor ya sea el de WWW ó el de la Base de Datos), se presentará un mensaje de error al usuario. Cuando esto suceda, no hay mucho qué hacer, pues la aplicación deja de trabajar inmediatamente (hasta que se haya normalizado la situación); lo único recomendable en estos casos es reportar la falla al administrador de SARC (es posible comunicar la anormalidad a través de correo electrónico: TODAS las páginas de la aplicación cuentan con la hiperliga necesaria para entrar en contacto con la administración).

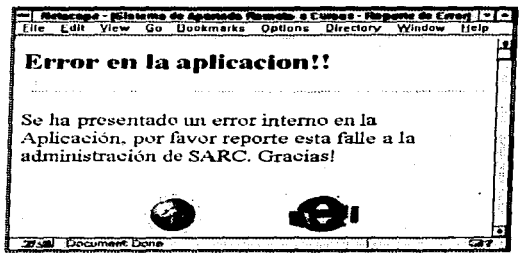



**Figura B.7.** Mensaje de error producido por SARC

Existen algunos mensajes que no necesariamente son fatales, es decir, no representan una falla mayor, por eso es útil leer con atención el texto del mensaje proporcionado pues en ocasiones sólo se tratan de advertencias, así que la ejecución podría continuar sin problema.

Las páginas expuestas hasta este momento constituyen una parte de SARC; para ser precisos, sólo conforman la **parte del usuario** de la aplicación (que no representa ni un 30% de SARC); el resto corresponde a la parte de administración de la aplicación, donde se realiza –entre otras cosas– el mantenimiento necesario a los catálogos de la Base de Datos comprendidos por el sistema de apartado.

Se ha considerado prudente no exponer las páginas que conforman la parte de administración de SARC, pues de hacerlo, éste documento sería una guía ilustrada para conocer detalles de la aplicación que pudieran ayudar a violar su seguridad. La información completa de dicha sección será entregada exclusivamente a las personas que vayan a cumplir las funciones de administración (y obviamente a los revisores del presente trabajo de tesis).

# C

# Códigos fuentes de SARC

Debido a su extensión, los códigos fuentes de SARC no han sido incluídos de manera impresa en esta publicación; sin embargo, todos ellos han sido reunidos en el diskette (formato 3.5", IBM compatible) que se anexa al presente trabajo de tesis.

La estructura de directorios que presenta el diskette de códigos fuentes es la siguiente:

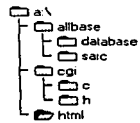



**Figura C.1.** Árbol de direcotrios del diskette de códigos fuentes de SARC.

Como es posible apreciar en la figura anterior, existen tres directorios principales:

- **allbase.** Contiene los códigos correspondientes a la Base de Datos All-Base. En el directorio **database** es posible encontrar aquellos programas isql empleados en la creación de la Base de Datos. En el direcotorio **sarc** se encuentran todos los scripts isql empleados por SARC al momento de su ejecución.

- **cgi.** Contiene todos los programas compuertas en lenguaje C. En **c** es posible encontrar códigos de programas; mientras en **h** se localizan los archivos de cabecera empleados en el desarrollo de SARC.

- **html.** Contiene todos los documentos HTML estáticos (páginas) empleados en la aplicación.

# D

# Configuración del servidor HTTP para la ejecución de CGI

Obviamente, el primer paso para ejecutar compuertas (CGI) es contar con un servidor HTTP; es posible obtener uno para plataformas UNIX de manera gratuita en el URL:

**http://hoohoo.ncsa.uiuc.edu/docs/Overview.html**

Este servidor es posible instalarlo en cualquier equipo que cuente con el sistema operativo UNIX; en caso de requerir que el servidor WWW sea visto en Internet, es necesario contar con una conexión (de preferencia dedicada) a la red global a través del protocolo TCP/IP[1]. Cualquier máquina que presente las características citadas, puede ser un servidor Web sin importar su poder de cómputo; obviamente, la velocidad de respuesta (atención a las peticiones de los clientes) dependerá en una relación directa del equipo de cómputo donde se esté ejecutando el demonio httpd.

## Instalación.

El servidor HTTP de NCSA puede encontrarse de dos maneras distintas[2]:

- **Precompilado.** En caso de contar con sistemas operativos UNIX tales como: Irix (Silicon Graphics), Solaris (Sun), HP-UX (Hewlett-Packard), etc., es posible obtener versiones precompiladas de servidores WWW (acordes al sistema operativo que se posea). Estas versiones ahorran el trabajo de compilar los programas fuentes del servidor; además, en caso de no contar con un compilador de lenguaje C en el sistema UNIX, ésta es la única opción disponible.

- **Programas fuente.** Es posible obtener los códigos fuente del servidor HTTP; en caso de inclinarse por esta opción, será necesario realizar la compilación (en

[1] Es importante mencionar que para instalar y utilizar un servidor WWW como el que aquí se expone, NO es indispensable la conexión a una red de computadoras, y mucho menos a Internet; una vez instalado el demonio httpd, es posible hacer que éste corra exclusivamente en la máquina en que ha sido instalado (de manera aislada), o bien en una red con ó sin acceso a Internet.

[2] Pueden obtenerse actualizaciones del servidor WWW de NCSA –para cualquier plataforma de cómputo– en **ftp://ftp.ncsa.uiuc.edu/Web/httpd/**.

lenguaje C) acorde al sistema operativo UNIX con que se cuente. Esta compilación es realizada a través de un archivo make; el usuario no debe preocuparse por ninguno de los intrincados detalles que representan la infinidad de opciones del compilador de lenguaje C (cc). Si se elige esta opción y no se modifica ningún valor de las variables de los códigos fuente, después de la compilación se obtendrá un programa ejecutable con las características estándar asignadas por NCSA (rutas de directorios, nombres de archivos, etc.).

Los procedimientos de instalación del servidor web de NCSA se encuentran fuera del alcance de este apéndice, pues su intención real es exponer los pasos para la configuración del servidor en el rangión de CGIs. Cualquier información extra acerca de las características generales de configuración e instalación pueden encontrarse en **ftp://ftp.ncsa.uiuc.edu/Web/http...nix/ncsa_httpd/...rent/httpd_docs.tar.Z**.

Una vez obtenido el programa ejecutable del ...idor WWW (por cualquiera de las dos opciones comentadas anteriormente), bastará con invocarlo desde la línea de comandos del sistema operativo UNIX (la ruta estándar de su ubicación dentro del sistema es: /usr/local/etc/httpd/, su nombre es ***httpd*** ). Inmediatamente después de su ejecución, el programa genera un archivo llamado **httpd.id** (en el directorio /usr/local/etc/httpd/logs/ ), el cual contiene el número de proceso padre del programa; en caso de necesitar "eliminar" la ejecución del demonio, bastará con "matar" el proceso padre mencionado (con el comando UNIX **kill**).

## Contenido del directorio .../httpd/.

El contenido del directorio /usr/local/etc/httpd/ variará dependiendo de la organización deseada por el administrador del servidor WWW; sin embargo, de manera estándar posee un solo archivo y seis directorios distintos:

- El **archivo** corresponde al programa ejecutable del servidor web, llamado **httpd**.
- El directorio **cgi-bin** es el lugar estándar propuesto por NCSA para contener los scripts ó compuertas CGI que son ejecutadas por el servidor.
- El directorio **cgi-src** es el propuesto por NCSA para contener los códigos fuente de las compuertas ó CGIs ejecutadas en el servidor.
- El directorio **conf** contiene los cuatro archivos de configuración del servidor WWW, éstos serán expuestos en la siguiente sección.
- El directorio **icons** es propuesto por NCSA para contener los gráficos incluidos en los distintos documentos HTML del servidor web.

- El directorio **logs** contiene varios archivos donde se registran: los mensajes de error que se presenten durante la ejecución del servidor; una bitácora de los accesos realizados al servidor web; una referencia al proceso padre de la ejecución actual del demonio httpd.

- El directorio **support** es propuesto por NCSA para contener utilerías empleadas por el demonio httpd para cumplir algunas de sus funciones; por ejemplo, por default existe en este directorio la utilería **htpasswd**, empleada para la creación de archivos con referencias login/password, útiles para el servidor al momento de efectuar tareas de seguridad, expuestas a mayor detalle en secciones posteriores de este anexo.

Todos estos directorios –como ya se comentó– son el estándar propuesto por NCSA; cualquiera de ellos puede cambiar en nombre y/o localización en el sistema (según las preferencias del administrador), siempre y cuando tal modificación sea registrada en los archivos de configuración del servidor, tema central de nuestra siguiente sección.

## Configuración del servidor WWW.

El directorio en el que se encuentran los archivos de configuración del servidor WWW es **/usr/local/etc/httpd/conf/** (según el estándar propuesto por NCSA). Cuando el demonio httpd es invocado, éste lee los archivos de configuración del directorio citado, haciendo una copia de ellos en memoria (duradera en tanto continúe la ejecución actual del servidor WWW). Cuando se desee cambiar alguna propiedad del servidor web podrán editarse sin problema alguno sus archivos de configuración (que son archivos planos); sin embargo, para que tales cambios surtan efecto, será necesario acabar con la ejecución actual (si es que hay alguna) y reiniciarla con las propiedades recién modificadas.

### Archivo httpd.conf.

Este archivo representa el punto de partida para la configuración de un servidor WWW. A continuación se exponen las principales directivas empleadas para definir la información básica del servidor:

- **ServerType**. Indica el modo en el que se ejecutará el servidor WWW. Existen dos modos: **standalone** (donde el servidor web atiende él mismo las peticiones de los clientes conforme éstas se presentan), e **`inetd`** (donde cualquier petición efectuada al servidor es comenzada y finalizada por un programa demonio llamado `inetd`). Por default, el modo es standalone.

Sintaxis: ServerType {standalone | inetd}
Ejemplo: SerVerType standalone

- **Port**. Indica el número de puerto que empleará el demonio httpd para recibir/atender las peticiones realizadas por el cliente. Por default, es el puerto 80.
  Sintaxis: Port *<número>*
  Ejemplo: Port 80

- **TimeOut**. Especifica el tiempo (en segundos) que el servidor esperará la llegada de *paquetes de información* antes de abortar la comunicación con el cliente. Por default, son 30 segundos.
  Sintaxis: TimeOut *<segundos>*
  Ejemplo: TimeOut 30

- **User**. Especifica el usuario dueño de los procesos hijo generados (por el demonio httpd) para atender las peticiones realizadas por el cliente. Por default es el usuario -1 (nobody).
  Sintaxis: User *<nombre de usuario>*
  Ejemplo: User nobody

- **Group**. Especifica el grupo dueño de los procesos hijo generados (por el demonio httpd) para atender las peticiones realizadas por el cliente. Por default no tiene valor.
  Sintaxis: Group *<nombre de grupo>*
  Ejemplo: Group webadmin

- **PidFile**. Especifica el nombre (y ruta) del archivo donde se guardará la referencia del proceso padre que ejecuta el demonio httpd. Por default es el archivo httpd.id, en el directorio logs.
  Sintaxis: PidFile *<archivo>*
  Ejemplo: PidFile logs/httpd.pid

- **ServerName**. Especifica el nombre (y dominio) correspondiente al servidor WWW.
  Sintaxis: ServerName *<dominio>*
  Ejemplo: ServerName www.xyz.com

- **ServerAdmin**. Permite especificar un nombre o dirección electrónica que se desplegará en la máquina cliente al ocurrir un error en el servidor.
  Sintaxis: ServerAdmin *<dirección ó nombre>*
  Ejemplo: ServerAdmin webmaster@www.xyz.com

Existen más directivas en este archivo; sin embargo, las aquí expuestas –según mi criterio– son las más importantes. Dentro del archivo httpd.conf, es posible encontrar a manera de comentarios una breve explicación de la función de cada directiva; así que, para manipular las aquí no contempladas, bastará con leer brevemente las

observaciones asentadas en dichos comentarios para poder modificar su valor al gusto del administrador del servidor WWW.

## Archivo mime.types.

Este archivo contiene los encabezados MIME disponibles[3], que auxilian al servidor a saber qué aplicación deberá ligar al encabezado recibido, cuando sea éste el que está aceptando la respuesta de otro servidor WWW.

Es recomendable no alterar los tipos estándar incluidos en este archivo; en caso de hacerlo, es prudente guardar una referencia del cambio (tal vez como comentario) para poder regresarlo a su estado original en caso de no efectuarse la acción tal y como era deseada.

## Archivo srm.conf.

Este archivo contiene una guía ó mapa de recursos del servidor (Server Resource Map; por su traducción al inglés); es decir, contiene directivas que indican la localización de recursos útiles al servidor para extender sus capacidades. **En este archivo se contiene la referencia necesaria para habilitar la ejecución, así como establecer el lugar base de los programas compuerta ó CGI.** A continuación se exponen las principales directivas comprendidas en este archivo:

- **DocumentRoot**. Especifica la ruta en la que por default estarán contenidos los documentos HTML que el servidor manda al cliente tras alguna petición realizada. Por default es /usr/local/etc/httpd/htdocs.
  Sintaxis: DocumentRoot *<directorio>*
  Ejemplo: DocumentRoot /usr/local/etc/httpd/htdocs
  Observaciones: Cualquier cliente que realice alguna petición al servidor cuya directiva DocumentRoot sea la arriba expuesta (y de dominio www.xyz.com), al referenciar http://www.xyz.com, estará viendo los documentos HTML contenidos en /usr/local/etc/httpd/htdocs/, la cual será la ruta más alta (en el árbol de directorios) a la que cualquier cliente tenga acceso; es decir, en caso de que éste intentara accesar un directorio superior al especificado (mediante .. ), no sería posible, pues DocumetRoot lo evitaría totalmente.

- **Alias**. Es posible atender peticiones de cualquier cliente a partir de documentos contenidos en un directorio distinto al especificado en DocumentRoot, empleado la directiva Alias.
  Sintaxis: Alias *<ruta virtual> <ruta física>*

[3] Para mayor información, véase la sección IV.3 del presente trabajo de tesis: Encabezados MIME en el ambiente CGI.

Ejemplo: Alias /tesis /usr/home1/cesar/trabajos/tesis
Observaciones: Cualquier cliente que realice alguna petición al URL http://www.xyz.com/tesis/abc.html, realmente estará visualizando el documento abc.html ubicado en el directorio /usr/home1/cesar/trabajos/tesis.

- **ScriptAlias. Especifica el directorio en el que podrán contenerse archivos "ejecutables", que puedan regresar alguna respuesta predefinida (por el programador) a todo cliente que los invoque; es decir, un programa compuerta ó CGI. Por default, esta directiva no posee ningún valor, así que en caso de no especificar alguno válido, el servidor con ésta directiva faltante, NO podrá hacer uso de CGIs.**
  Sintaxis: ScriptAlias *<ruta virtual> <ruta física>*
  Ejemplo: ScriptAlias /cgi-bin /usr/local/etc/httpd/cgi-bin
  Observaciones: Al igual que en la directiva Alias, cualquier cliente referenciando el URL http://www.xyz.com/cgi-bin/a.cgi, estará invocando no a un documento HTML estático (como hasta el momento se ha visto), sino al programa a.cgi, que tras ejecutarse, enviará una cadena de respuesta al cliente que lo invocó; nótese que la ruta real en que se encuentra el programa compuerta ó CGI citado es /usr/local/etc/httpd/cgi-bin.

Existen más directivas en este archivo; sin embargo, las aquí expuestas son las más importantes, pues de alguna manera motivaron la creación del presente apéndice. Dentro del archivo srm.conf, es posible encontrar a manera de comentarios una breve explicación de la función de cada directiva; así que, para manipular las aquí no contempladas, bastará con leer brevemente las observaciones asentadas en dichos párrafos para poder modificar su valor al gusto del administrador del servidor WWW.

## Archivo access.conf.

Éste es el último de los cuatro archivos de configuración con que cuenta el servidor web de NCSA (y por estándar, cualquier otro para plataformas UNIX). Contiene todas las directivas relacionadas con cuestiones de permisos y seguridad en el servidor web. Las directivas más importantes que comprende son:

- **Directory**. Esta directiva otorga una serie de propiedades de acceso a un directorio determinado. En el archivo access.conf pueden existir tantas directivas directory como sean necesarias.
  Sintaxis: <Directory *<ruta física>* . . . *opciones* . . . </Directory>
  Ejemplo: <Directory /usr/local/etc/httpd/htdocs/SARC>
  Options None
  </Directory>

- **Limit**. Esta directiva es una opción de **directory** (antes expuesta), restringe el acceso a los archivos existentes en el directorio indicado. Por default, se encuentra abierto a cualquier dirección y no solicita password alguno.
  Sintaxis: Limit { GET | PUT | POST } . . . > . . . *opciones* . . . </Limit>
  Ejemplo: Véase al final del punto **Order**.

- **Deny**. Esta directiva es una opción de **Limit**. Indica el nombre de los hosts, o direcciones IP para los que se encuentra restringido el acceso al directorio especificado. Por default, ningún directorio tiene algún tipo de restricción.
  Sintaxis: Deny from { *<host(s)>* | all }
  Ejemplo: Deny from 132.248.145.5
  Deny from morsa.dgsca.unam.mx
  Deny from all

- **Allow**. Esta directiva es una opción de **Limit**. Indica el nombre de los hosts ó direcciones IP para los que se permite el acceso al directorio especificado. Por default, el acceso es permitido a cualquier host.
  Sintaxis: Allow from { *<host(s)>* | all }
  Ejemplo: Allow from 132.248.145.3
  Allow from polaris.labvis.unam.mx
  Allow from all

- **Order**. Esta directiva es una opción de **Limit**. Indica el orden en que las directivas **Deny** y **Allow** será evaluadas. Por default la evaluación es Deny - Allow.
  Sintaxis: Order { deny,allow | allow,deny }
  Ejemplo:
  ```
  <Directory /home1/users/cesar>
      <Limit POST>
      order deny,allow
      deny from all
      allow from 132.248.160.4
      </Limit>
  </Directory>
  ```

- **AuthUserFile**. Esta directiva es una opción de **Directory**. Especifica el nombre (y ruta) del archivo que contiene la relación de login/password permitida, generada con la utilidad **htpasswd** en el directorio /usr/local/etc/httpd/support[4].
  Sintaxis: AuthUserFile *<ruta física>*
  Ejemplo: AuthUserFile /usr/local/etc/httpd/grants/permisos.htpasswd

- **AuthName**. Esta directiva es una opción de **Directory**. Especifica el nombre del ó los usuarios a quienes es permitido el acceso al directorio especificado.
  Sintaxis: AuthName *<nombre>*
  Ejemplo: Authname pandilla nerds cesarin

[4] Para mayor información sobre la utilidad **htpasswd**, consultar la documentación proporcionada por NCSA en la dirección **ftp://ftp.ncsa.uiuc.edu/Web/httpd/Unix/ncsa_httpd/current/httpd_docs.tar.Z**

- **Ejemplo general de Directory.** A continuación se expone un ejemplo que comprende las opciones de **Directory** explicadas anteriormente:

```
<Directory /usr/local/etc/httpd/htdocs/tesis/cesar>
    AuthName cesarin
    AuthUserFile /usr/local/etc/httpd/passwords.httpd
    <Limit GET>
        Order deny,allow
        Deny from all
        Allow from 132.248.145.2
    </Limit>
</Directory>
```

Existen más directivas en este archivo; sin embargo, las aquí expuestas son –desde mi punto de vista– las más importantes. Dentro del archivo access.conf, es posible encontrar a manera de comentarios una breve explicación de la función de cada directiva; así que, para manipular las aquí no contempladas, bastará con leer brevemente las observaciones asentadas en dichos párrafos para poder modificar su valor al gusto del administrador del servidor WWW.

# Glosario

**ancho de banda** Capacidad máxima de transmisión en un enlace de comunicación, usualmente se mide en bits por segundo (bps).

**ANSI C** Versión de lenguaje C proporcionado por el American National Standards Institute, organismo de los E.U., encargado de establecer diversos estándares en ese país.

**API** Application Programming Interface. Interfaz estándar utilizada en el desarrollo de programas fuentes de aplicaciones, que asegura la compatibilidad de aplicaciones en plataformas que soportan API, incluyendo plataformas de diferentes arquitecturas. Esto significa que sólo una versión de una aplicación API necesita ser generada, y posteriormente el desarrollador sólo recompila el programa fuente en cada una de las plataformas en la cual la aplicación va a ejecutarse.

**archivo plano** Arreglo simple bidimensional de elementos dato. Comúnmente empleado para almacenar la información descriptiva necesaria acerca de los datos que se hallan en una localización dentro del archivo mismo.

**broadcast** Mensaje que al ser emitido se recibe por todos los nodos de la red actual.

**Buffer** Segmento reservado de memoria que se emplea para almacenar datos mientras se procesan; también es conocido como pequeño banco de memoria usado para ciertos fines.

**CSMA/CD** Carrier Sense Multiple Access/Collision Detection. Protocolo de comunicaciones de acceso al medio. Empleado por la topología Ethernet. Cuando un dispositivo trata de mandar un mensaje a través de la red, verifica si el canal (el medio) está libre, si no lo está, espera una cantidad aleatoria de tiempo antes de intentarlo nuevamente. Si la red se encuentra libre y dos dispositivos tratan de ganar acceso exactamente al mismo tiempo, ambos se retractan para evitar una colisión y luego cada uno espera cierta cantidad aleatoria de tiempo antes de reintentarlo.

**compilador** Programa que sirve para traducir un lenguaje de programación simbólico más alto a un lenguaje de máquina, comprensible para el procesador. El programa traducido no será ejecutado sino hasta que la traducción realizada se haya efectuado correctamente para la totalidad del programa en cuestión.

| | |
|---|---|
| **conexión dedicada** | Acción que consiste en unir dos o más elementos o componentes, de manera que exista un canal de comunicación permanente y único entre estos. |
| **datagrama** | Mensaje enviado en una red de conmutación de paquetes. |
| **daemon** | Programa o aplicación que se encuentra generalmente en "modo de espera" (listo para ejecutarse), de forma tal que cuando se cumplan ciertas condiciones o determinado evento programado, realice las funciones para las que fue creado. |
| **entrada estándar** | Medio estándar por el cual un programa puede recibir información; típicamente es el teclado, aunque puede variar. |
| **Ethernet** | Red de computadoras dispuesta en forma de bus lineal. Es la topología de red más empleada en la actualidad. Emplea el protocolo CSMA/CD. |
| **emulación** | Proceso por el cual alguna computadora puede correr programas que no estén escritos específicamente para esta, debido a que simulan las operaciones de otra computadora. |
| **Frame Relay** | Protocolo de conmutación de paquetes de alta velocidad. Es más adecuado para la transferencia de datos e imágenes que para voz. |
| **FDDI** | Fiber Data Distributed Interface. Red de computadoras dispuestas en forma de doble anillo redundante; comúnmente empleando fibra óptica como medio de transmisión. |
| **GUI** | Graphical User Interface. Interfaz de usuario basada en gráficos, incorporando íconos, menús desplegables, mouse, etc. |
| **hardware** | Circuitos electrónicos y dispositivos electromecánicos que constituyen un sistema de computación. |
| **hipermedia** | Extensión del concepto de hipertexto para la inclusión del manejo de información en forma de imágenes, gráficas, audio o video. |
| **hipertexto** | Termino usado para referirse a un sistema no lineal de búsqueda y recuperación de información, que actúa mediante texto resaltado. |
| **host** | Computadora que trabaja dentro de una red, específicamente un transmisor o receptor de información, desde el punto de vista de la red global de comunicación Internet. |
| **inetd** | Internet Services Daemon. Es el programa demonio principal del conjunto de protocolos TCP/IP; es responsable de supervisar TODAS las operaciones de red efectuadas. Realiza su trabajo fiscalizando un conjunto de programas demonio, cada uno especializado en una tarea específica. |
| **ISO** | International Organization for Standarization (Organización Internacional de Estándares). |
| **ícono** | Pequeña representación pictórica de un objeto en pantalla a algún otro medio de almacenamiento de información, utilizada en interfaces gráficas. |

| | |
|---|---|
| **indexar** | Colocar un elemento en un lugar específico dentro de un arreglo o archivo de datos, o bien, preparar una lista ordenada de referencias a los contenidos de un conjunto mayor de datos |
| **información en línea** | Información disponible a través de una conexión activa, cuya respuesta se presenta inmediatamente acorde a una petición formulada. |
| **interfaz** | Entidad que da una apariencia distinta de la que originalmente tiene un elemento cualquiera. |
| **interfase** | Entidad que permite el correcto funcionamiento entre dos elementos distintos. Es un punto de unión para la interacción entre dos fases distintas. |
| **interoperabilidad** | Capacidad de un sistema para comunicarse y/o interactuar con otro. |
| **Intranet** | Red de uso privado (de cualquier magnitud) que emplea la misma tecnología de Internet a través del conjunto de protocolos TCP/IP, el ruteo independiente y la conmutación de paquetes. |
| **intérprete** | Programa que ejecuta un lenguaje de alto nivel, este recorre cada línea del código del programa fuente (escrito en alto nivel), trasformándolo en código máquina que la computadora entiende. Cada vez que el programa es llamado, traduce las instrucciones y de inmediato las ejecuta, línea por línea. |
| **loopback** | Mensaje emitido por una computadora a si misma, registrando el envío de datos sin generar tráfico de información en la red actual. |
| **motor de búsqueda** | Programa de aplicación capaz de efectuar búsquedas simples y complejas ya sea en bases de datos distribuidas o concentradas en uno o varios servidores conectados a Internet a través de la tecnología de CGIs. |
| **NFS** | Network File System (Sistema de Archivos en Red), permite a sistemas de archivos localizados en computadoras distantes, compartir información aparentando a los usuarios que los sistemas de archivos remotos se encuentran ubicados localmente, como un directorio más del sistema. |
| **NCSA** | National Center for Supercomputing Applications, Centro Nacional (de Estados Unidos) para desarrollo de Aplicaciones en Supercómputo. Creador de Mosaic. |
| **nodo** | Computadora o terminal que proporciona un punto de entrada/salida de datos en una red de computadoras, la cual también puede procesar información. |
| **pixel** | Nombre que se le da a cada punto-elemento sobre la pantalla, en una exhibición por barrido en un tubo de rayos catódicos (monitor). |
| **plug-in** | Programa de aplicación que sirve para extender las capacidades de despliegue a través del visualizador de Netscape. Los Plug-ins son empleados comúnmente para poder accesar a ciertos archivos de audio, video, tercera dimensión o realidad virtual entre otros. |
| **Portabilidad** | Propiedad del software que le permite ser empleado en diferentes arquitecturas de computadora. |

| | |
|---|---|
| **protocolo** | Conjunto de reglas y normas que gobiernan el formato y significado de los paquetes o mensajes, que se utilizan al establecer una comunicación entre dos o más computadoras. |
| **red de computadoras** | Es un conjunto de computadoras autónomas que se encuentran interconectadas entre sí, estableciendo un medio de comunicación eficiente para compartir recursos y distribuir tareas. |
| **Red UNAM** | Red de área amplia, distribuida principalmente en territorio mexicano, bajo la supervisión de la UNAM a través de su Dirección de Telecomunicaciones Digitales. Su función principal es proporcionar un medio de comunicación electrónica eficiente entre los diversos campus que la integran. |
| **ruta absoluta** | Es un camino dentro del árbol de directorios; su descripción se realiza desde el inicio del árbol de directorios; es decir, la raíz del mismo (en UNIX es /...; en DOS es \...; etc.). |
| **ruta virtual** | Es un camino dentro del árbol de directorios en el que el eje central en torno el cual gira la descripción de la locación deseada, es el directorio actual. |
| **salida estándar** | Medio estándar por el cual un programa puede dar información al usuario; típicamente es el monitor, aunque puede variar. |
| **shareware** | Software de distribución pública, pero no de uso gratuito. El autor o distribuidor establece un periodo de prueba, después del cual solicita una cuota de recuperación. |
| **SQL** | Structured Query Language. Lenguaje utilizado para llevar a cabo el procesamiento de datos y transacciones en una base de datos de tipo relacional. |
| **sistema operativo** | Conjunto de programas y rutinas que administran las múltiples funciones y tareas que desempeña una computadora, además proporciona ayuda a los programas y a los usuarios de esta, con funciones de apoyo, incrementando la utilidad de hardware. |
| **software** | Serie de instrucciones y rutinas (programas) que realizan una tarea en particular, comúnmente llamada programa de software. |
| **token passing** | Protocolo de comunicaciones de acceso al medio que utiliza un cuadro de repetición continua (el token). Empleado por la topología `Token Ring`. Cuando una computadora desea enviar un mensaje, espera un token vacío; cuando encuentra uno, lo llena con la dirección destino y el mensaje deseado. Para transmitir nuevamente, deberá esperar otro token vacío; en este protocolo sólo se transmite un token a la vez en el medio físico de la red. |
| **Token Ring** | Red de computadoras dispuestas en forma de anillo. Hace uso del `protocolo token passing`. |
| **variables de ambiente** | Son variables que contienen valores consistentes al ejecutar cualquier aplicación, importantes para el sistema operativo o una aplicación en particular. |

# Referencias

## Bibliografía:


- **FRISCH,** Aeleen, "Essential System Administration", Editorial O'Reilly & Associates, Estados Unidos, 1992, Primera edición.
- **M. CHANDLER,** David, "Running a Perfect Web Site", Editorial QUE, Estados Unidos, 1995, Primera edición.
- **J. YEAGER,** Nancy, "Web Server Technology", Editorial Morgan Kaufmann, Estados Unidos, 1996, Primera edición.
- **CHEONG,** Fah-Chun, "Internet Agents", Editorial New Rider Publishing, Estados Unidos, 1996, Primera edición.
- **CHELLIS,** James, "The CNE-4 Study Guide", Editorial Network Press, Estados Unidos, 1996, Primera edición.
- **DECEMBER,** John, "HTML & CGI Unleashed", Editorial Sams Net, Estados Unidos, 1995, Primera edición.
- **AUTORES VARIOS,** "Java Unleashed", Editorial Sams Net, Estados Unidos, 1996, Primera edición.
- **LIBES,** Don, "Exploring Expect", Editorial O'Reilly & Associates, Estados Unidos, 1995, Primera edición.
- **E. WEINMAN,** William (Traductor José Luis Gutiérrez), "El libro de CGI", Editorial Prentice Hall Hispanoamericana, México, 1996, Primera edición.

- **AUTORES VARIOS,** "Software Engineering HandBook", Editorial McGraw-Hill, Estados Unidos, 1986, Tercera edición.
- **KHOSHAFIAN,** Setrag, "A Guide to Developing Client/Server SQL Applications", Editorial Morgan Kaufmann, Estados Unidos, 1992, Primera edición.
- **AUTORES VARIOS,** "AllBase/ISQL Reference Manual", Editorial Hewlett Packard Inc., Estados Unidos, 1991, Primera edición.


## URLs:


- **The WWW Consortium (W3C).**
  http://www.w3.org/pub/WWW
- **The Internet Society**
  http://www.isoc.org/home.html
- **A history of Internet Protocols at CERN**
  http://wwwcn.cern.ch/pdp/ns/ben/TCPHIST.html
- **The Web Designer**
  http://web.canlink.com/webdesign
- **The Common Gateway Interface**
  http://hoohoo.ncsa.uiuc.edu/cgi