T
004.6
L732
c.2

Escuela Superior
Politécnica del Litoral
Facultad de Ingeniería en Computación

TESIS DE GRADO

Previa a la Obtención del Titulo de:

INGENIERO EN COMPUTACION

Estudio e Implementación de un Sistema de Información Interactivo para la ESPOL basado en el World Wide Web

Presentada por:

Servio F. Lima Reina

* * *

Guayaquil - Ecuador

Año 1996

A mis padres, por su incondicional ayuda en todo momento. A Telconet y Cesercomp, dos empresas que me brindaron un gran apoyo logistico.

A mis padres

A mis hermanos

A Dalila

ING. CARLOS VILLAFUERTE
Decano FIEC ESPOL

ING. GUIDO GALLEDO ROSSI
Director de Tesis

Dr. Enrique Peláez
Miembro Tribunal de Tesis

Ing. Carlos Monsalve
Miembro Tribunal de Tesis

ESPOL
ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL
BIBLIOTECA
CENTRAL

"La responsabilidad por los hechos, ideas y doctrinas expuestos en esta tesis, me corresponden exclusivamente; y, el patrimonio intelectual de la misma, a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL".

(Reglamento de Exámenes y Títulos profesionales de la ESPOL)

Nombre y Firma del Autor

RESUMEN

El World Wide Web es una gran telaraña de sistemas de informacion dispersos por todo el mundo y conectados a la Internet. Su crecimiento se debe a la gran aceptacion que ha tenido en todas las regiones del planeta hacia las cuales llega la Internet y a la tecnologia sobre la que se basa, la cual es cambiante e innovadora. Estos sistemas de informacion tienen dos caracteristicas que hasta antes de 1990 (aiio de creación del primer programa servidor web), no eran eficientemente explotadas en la Internet: se trata de su capacidad de publicar informacion hipermedia (texto al que se le han aiiadido caracteristicas de hipertexto y multimedia: imagenes, sonidos, etc) y enlazar informacion u "hojas de hipertexto" pertenecientes a cualquier rincon del planeta que posea un programa servidor web conectado a la Internet, a traves de los llamados hiperenlaces.

Existen personas de todo el mundo que, dia a dia trabajan en el mejoramiento de las tecnologias existentes en el Web. El presente trabajo constituye una contribución mas al mejor uso de tales tecnologias y es una demostracion de lo que se puede realizar a traves de la investigación en la Internet. Se ha desarrollado un Sistema de Información para la ESPOL, siguiendo los estándares definidos por algunas publicaciones y personas o instituciones con una gran experiencia del tema. Pero eso no es todo; además de ceiiirnos

a un proceso riguroso de desarrollo de servidores web que va desde la planificacion, pasando por el analisis, diseño, implementación, promoción e innovación del servidor web, se ha puesto enfasis en los siguientes aspectos:

a. Búsqueda de las mejores herramientaspara sewidores web existentes tales como:

Máquinas de Busqueda que faciliten al usuario del servidor de la ESPOL hallar la informacion de una manera rapida y eficiente.

Servidores Proxy que mejoren la velocidad de acceso a las páginas de hipertexto por parte de los usuarios de la ESPOL hacia la Internet.

Programas estadisticos, que faciliten al administrador del servidor, conocer diferentes parametros de comportamiento de los usuarios que visitan el servidor web.

b. Esquemas de seguridad para el envio de información.

Debido a que en servidores pertenecientes a instituciones como la ESPOL, en algh momento se deberá manejar informacion confidencial (por ejemplo: envio de clave para acceso a informacion administrativa, notas, etc), se hizo un estudio de un esquema de seguridad para servidores web conocido como Secure Socket Layer. Mediante este sistema la informacion que envia un usuario a traves de su visor de hipertexto (browser) viaja de manera encriptada (codificada para que no pueda ser interpretada por algh interceptor de la informacion) hacia el servidor web. Este esquema de seguridad permite definir una clave diferente por cada conexión entre el cliente y el servidor de web, lo cual lo hace sumamente dificil de romper. .

ESCUELA SUPERIOR POLITECNICA DEL LITORAL
BIBLIOTECA
CENTRAL

c. *Desarrollo de aplicacionespara sewidores web.*

Basadas en lo que se conoce como los CGI (Common Gateway Interfase), los cuales permiten ejecutar las aplicaciones del lado del servidor web, como por ejemplo:

1. Acceso a la base de datos de usuarios del Sistema Operativo en el cual corre el servidor web en la ESPOL, para la validacion de un usuario.

2. Utilización de herramientas de programacion del Sistema Operativo en el cual corre el servidor web de la ESPOL, como son los sockets (caminos de comunicacion entre un cliente o browser y un programa servidor web), para la validacion de los hiperenlaces (por ejemplo, validacion de enlaces de la forma: http://www.cnn.com)

3. Empleo de los utilitarios propios del Sistema Operativo donde corre el servidor web de la ESPOL, como son: el Crontab, para ejecucion de tareas en forma periodica y el Sendmail para ejecucion de programas a traves del correo electrónico

De esta manera, el presente trabajo pretende dar un camino a seguir para el desarrollo o mejoras a futuros servidores web orientado a Universidades, sin importar la tecnologia a usarse (como por ejemplo: Java, ActiveX, CGIs u otras) , puesto que en la mayoria de los casos, habrá que tomar en cuenta los aspectos mencionados previamente.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

Índice de Figuras

Capitulo I

Capitulo II

Capitulo III

Capitulo IV

Capítulo V

Capítulo VI

ESPOL
ESCUELA SUPERIOR POLITECNICA DEL LITORAL
BIBLIOTECA CENTRAL

Índice de Tablas

Existen otras interrogantes relacionadas con el desarrollo de aplicaciones para servidores web: ¿Cuál es el mejor lenguaje en el que se pueden desarrollar aplicaciones para la plataforma existente en la ESPOL? ; ¿Cuáles son las consideraciones de seguridad a tomarse en tales aplicaciones?; ¿Son necesarios tales consideraciones?; ¿Es recomendable utilizar herramientas previamente desarrolladas en la Internet a crear una aplicacion con nuestros medios, que realice la misma tarea? Todas estas interrogantes son aclaradas en la presente Tesis, sentando un precedente para el desarrollo de servidores web de "calidad".

Se definieron además una serie de objetivos para el servidor web de la ESPOL, que directa o indirectamente ayudarán a aclarar las interrogantes planteadas. Estos son:

1. Desarrollar un sistema de información que de a conocer la Universidad y nuestro País a la comunidad de Internet, a traves de la implementación de páginas de facil consulta y acceso, con metodos de busqueda y mapas sensitivos que faciliten el uso del sistema por parte del usuario final.

2. Desarrollar un medio por el cual los miembros y no miembros de la ESPOL puedan integrarse al mundo del World Wide Web a traves de herramientas como: La generación de homepages, el libro de invitados o Guestbook, la estructuracion

del directorio de personas y la recopilacion de avisos de interes para el alumnado, todo esto de manera automática.

3. Explorar el web como interfase con el usuario final, integradora de los diferentes servicios de Internet, a traves de la adicion al servidor web del correo electronico, transferencia de archivos o FTP y Gopher.

4. Estudiar las diferentes herramientas de los sistemas de información basados en el World Wide Web tales como las maquinas de busqueda y los servidores Proxy y escoger e implementar las que presenten el mejor rendimiento y la máxima eficiencia, adaptándolas a la realidad de hardware y software de la ESPOL.

5. Desarrollar herramientas de administración del Sistema de Información que permitan controlar y monitorear diferentes parametros del mismo, tales como, el chequeo de hiperenlaces y las estadísticas de acceso al servidor.

Con estos objetivos en mente, procederemos a la explicación, en las siguientes páginas, del servidor web de la ESPOL.

CAPITULO I

LOS BROWSERS O NAVEGADORES DE INTERNET

1.1. Los browsers como clientes en el modelo Cliente/Servidor

En la arquitectura cliente/servidor, los navegadores *o* browsers son *tan* importantes como los servidores web. Para el usuario final, el browser sera una de las interfases de entrada al mundo del World Wide Web (tambien puede accesarse a los servidores web via correo electronico) que le pennitira visualizar homepages e interactuar con las aplicaciones del servidor. Existe todo tipo de browsers: Graficos, textuales, con caracteristicas limitadas *o* caracteristicas extendidas, entre otros. La selección de uno de tales browsers, dependera del hardware y software que tuviese el usuario final.

Cada vez mas, los browsers se hacen mas complejos en su arquitectura, sobrepasando en muchos casos, la complejidad de los programas servidores web. Al principio fueron simple lectores de HTML (Lenguaje utilizado para dar formato a los documentos mostrados por el servidor web). Hoy en dia, son capaces de soportar las mas variadas caracteristicas, como: Frames (elementos HTML permiten separar la información mostrada en el browser, a traves de marcos o ventanas), elementos extendidos de HTML (que añaden mas caracteristicas al lenguaje HTML), imágenes en movimiento, lenguajes de programacion como CGIs (aplicaciones que se ejecutan del lado del servidor web), Java (lenguaje multiplataforma creado por Sun Microsystems), Javascript (sentencias de Java incrustadas en el código HTML), Vbscripts (sentencias de Visual Basic incrustadas en el código HTML), entre otros (Ver apéndice A para una explicación mas detallada de estos terminos).

La arquitectura del World Wide Web posee todos los requisitos necesarios para ser catalogada como arquitectura Cliente/Servidor. Cliente/Servidor es una arquitectura de uso universal que se basa en la existencia de dos elementos: un lado cliente que es el que por lo general realiza los requerimientos, y un lado servidor, que los atiende. Una de las razones de la existencia de Cliente/Servidor es que contribuye a balancear la carga de procesamiento que antigiiamente soportaban los servidores solamente, pasandola al lado

del cliente, esto es, los clientes son un poco mas "inteligentes" y capaces de realizar tareas sin necesidad de recurrir al servidor.

Pero, ¿dónde encajan los browsers en la arquitectura Cliente/Servidor? Pues esta es una pregunta cuya respuesta solo puede ser entendida viendo al WWW como un todo integrador de los programas servidores web por un lado, la red Internet y los navegadores por otro. Para entender mejor toda esta terminología, nos guiaremos en base al diagrama de la figura 1.1 que esquematiza una macrovision de lo que es el modelo Cliente/Servidor del WWW

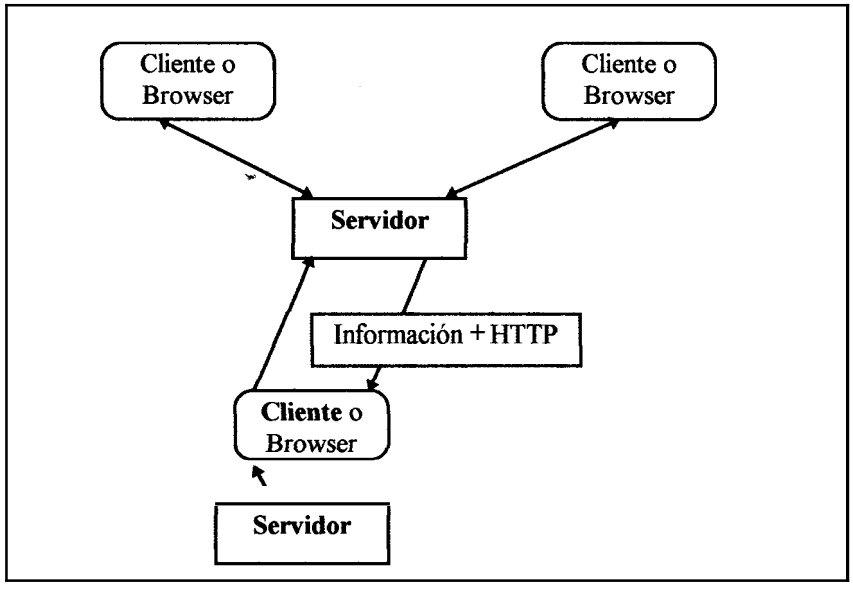


Figura 1.1.
Comunicación Cliente/Servidor *en* el WWW

El modelo Cliente/Servidor del WWW se compone de diferentes partes: Por un lado están los usuarios finales usando los **browsers** o **navegadores** para accesar a cierta informacion , y por otro están los **programas servidores web,** quienes ofrecen tal informacion. Enlazando unos con otros se encuentra el **protocolo** HTTP, que es el lenguaje con el que los clientes y servidores se entienden entre si a traves de una o varias redes TCP/IP. El usuario final simplemente ve una interface agradable que es el *lado del* cliente, y para el es transparente la existencia del protocolo HTTP. Además, solo conocera que existe un servidor web al tener que poner la dirección de tal servidor.

Cuando un browser inicia un requerimiento a un servidor, ocurre lo siguiente: El requerimiento es creado usando el protocolo HTTP. Este requerimiento llegara hasta el servidor (el cual puede estar fisicamente en la misma maquina que el cliente o en una maquina diferente en cualquier red), el cual lo interpretara y cumplirá con la acción indicada, que puede ser el enviar de regreso una hoja de hipertexto o el ejecutar un programa CGI, entre otros.

1.2. Funcionamiento de los browsers

Gracias a los browsers es que podemos accesar a la informacion que los servidores web nos brindan. Cada vez mas crece la complejidad de tales browsers, permitiendoles accesar a informacion textual, visual y audible. Poseen su propio interpretador de

HTML, que es el lenguaje en el que están confeccionadas las hojas de Hipertexto. Son capaces de ejecutar aplicaciones (tales como las que están hechas en Java, Javascript, etc.), y cada vez más es notoria su independencia de los servidores web de la red. Pero no todos los browsers tienen la capacidad de presentar de igual manera la información . Lo que un grupo de browsers puede interpretar correctamente, para otro grupo de browsers es incomprensible. Esto es debido a que, en la actualidad, algunos browsers tienen sus propios elementos de lenguaje HTML, llamados "extendidos", que no son soportados por el resto de browsers. Un ejemplo de esto es el elemento de HTML llamado MARQUEE que sirve para presentar un texto en movimiento en el Microsoft Internet Explorer (browser comercial). Dicho elemento HTML es solamente interpretado por este browser, mas no por Netscape Navigator (browser comercial) u otros browsers existentes. Para complicar el asunto, para un mismo browser, Qgamos Netscape, existen diferentes versiones que aparecen en el tiempo, que soportan un conjunto de elementos HTML superior en número, al que soportaba una version previa del mismo browser. Para aliviar todas estas incompatibilidades, existe una organización llamada el W3 C o WWW Consortium que es la que define el estándar HTML vigente, sobre el cual se debe tender a desarrollar las hojas de hipertexto de servidor web, de tal manera que pueda ser visto por la mayor cantidad de público posible. En la actualidad el estándar HTML vigente es HTML 2.0.

Pero, pasemos ahora a analizar el funcionamiento de los browsers, de tal manera que entendamos el funcionamiento de los mismos. La siguiente figura muestra como es posible que un browser realice multiples funciones a la vez.

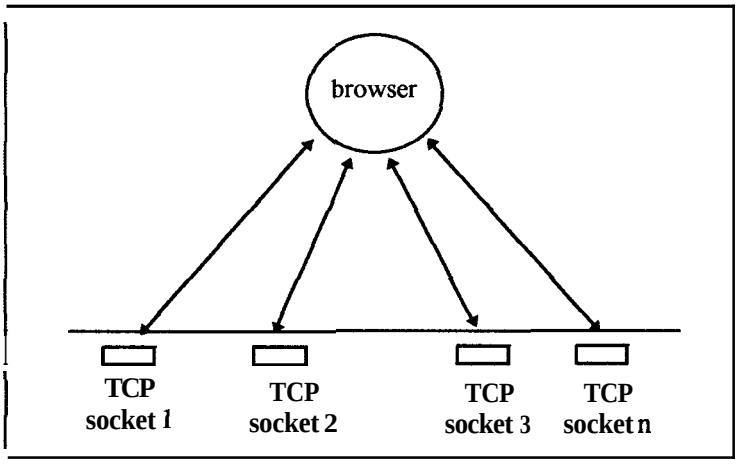


Figura 1.2.
Funcionamiento de los Browsers o Navegadores
Fuente: Commer; Internetworking with TCP/IP, vol III

En la grafica 1.2 observamos un tipico ejemplo de un browser multitarea, el cual , abre un proceso por cada lazo de comunicacion con el programa servidor web. Es necesario conocer algunos conceptos de TCP/IP antes de explicar el grafico en mención.

Lo primero que debemos conocer es el hecho de que TCP/IP no lo constituye un solo protocolo, sino un conjunto de protocolos que definen las bases para el funcionamiento de una red TCP/IP y que son independientes de la topología con la que se trabaje (sea esta una red Ethernet, Token ring, FDDI, etc). Por un lado está el protocolo TCP

En un mismo computador pueden existir multiples procesos o aplicaciones ejecutándose simultáneamente (FTP, TELNET, GOPHER y/o sus respectivos servidores). Cada aplicacion debe tener un identificador único que permita la comunicacion entre procesos que pueden estar en la misma máquina o en maquinas diferentes. A este indentificador o direccion única de procesos se lo conoce con el nombre de "puerto". El número de puertos o de identificadores que puede tener un mismo computador es del orden de 2^{16} (esta es una restricción de TCP/IP). Una vez conocidos el puerto de la aplicacion y la direccion IP del cornputador, podemos establecer un camino de comunicacion entre otro puerto y computador en la misma red o en redes Qferentes, a traves de lo que se conoce como "sockets". Los sockets entonces, son caminos de comunicacion entre procesos que funcionan bajo el esquema armado por TCP e IP. Existe una analogia entre lo que son los sockets y el protocolo TCP, ya que mientra los sockets se encargan de comunicar procesos de un computador con otro, TCP se encarga de enlazar confiablemente tales computadores.

A nivel de procesos existe una diferenciacion de los mismos, dependiendo de las tareas que ejecuten. Aquellos procesos encargados de realizar un requerimiento son conocidos como "clientes" mientras que los procesos que atienden tales requerimientos son conocidos como "servidores". Un mismo proceso puede actuar tanto como cliente como

servidor puesto que puede atender y requerir al mismo tiempo. Este esquema constituye lo que se conoce como arquitectura Cliente/Servidor orientada a TCP/IP.

Tomando en consideración toda la teoría mencionada, procederemos a explicar el gráfico de la figura 1.2:

La comunicacion cliente/servidor está representada por TCP/socket1 , TCP/socket2, TCP/socketn. Un socket o camino de comunicacion se crea por cada tarea que realice el browser, como por ejemplo, la recepción de imagenes o ejecucion de comandos SSI (ver sección 3.6.1).

La comunicacion cliente/servidor a nivel de Web se realiza a traves del puerto número ochenta (80) por default, pero podría especificarse otro puerto.

Existen ciertas consideraciones que hay que tomar en cuenta, en la comunicacion con el cliente o browser:

1. El browser lee la entrada desde el servidor a cualquier tasa que este genere tales entradas.

2. El procesamiento local continuara aim si el servidor se demora. Mas aim, el browser continuara leyendo comandos de control aún si el servidor falla al responder.

3. Luego de recibir la informacion completa en el lado del browser, el servidor cierra la conexion. Cabe anotar que el browser puede abortar la conexion, causando que se interrumpa el flujo de informacion a traves del socket. A pesar de esto, del lado del servidor, aim quedara un proceso activo (el proceso que atendia al browser), hasta que haya transcurrido un cierto tiempo (tirneout) definido del lado del servidor (en los archivos de configuración del servidor).

1.3. Comunicación con el servidor: Mensajes MIME

MIME (Siglas en ingles de Multipart Internet Mail Extensions) es un estándar de manejo de tipos de datos, comúnmente usado en el correo electrónico para la transferencia de todo tipo de informacion no textual. MIME fue adoptado por los programas servidores web como estándar para envio de informacion no textual entre el cliente (browser) y el programa servidor web. MIME reconoce el tipo de archivo solicitado, gracias a un encabezado adicionado al archivo, relacionado con una extension. De esta manera, los clientes y servidores web diferencian a una hoja de

hipertexto (extension .htm o .html), de un archivo de imagenes (.gif, .jpg) y pueden discernir que acción tomar en base al tipo de archivo dado.

Si hablamos de ejecucion de programas CGIs (mas adelante se darán detalles sobre los CGIs), MIME ocupa un papel preponderante en su elaboración. Todo resultado que arrojase el programa CGI debe ir encabezado por un mensaje de tip0 MIME en el cual se especifique que tipo de información se está entregando.

Los encabezados de tipo MIME son de la forma: tipo/subtipo, donde tipo es el tipo de archivo entregado y subtipo es una subclasificacion dentro del tipo de archivo dado. Por ejemplo,

para denotar un archivo texto, definimos un encabezado: **text/plain**

para denotar un archivo de hipertexto, definimos un encabezado: **text/html**

para denotar un archivo grafico, definimos un encabezado: **image**

para denotar un archivo comprimido para DOS, definimos un encabezado: **application/zip**

Tanto el cliente(browser) como el programa servidor web poseen una tabla en la que se especifican la mayoría de los formatos MIME existentes.

1.4. El browser como interpretador de HTML

Si un browser requirio una simple hoja de hipertexto, esta sera enviada de regreso por el servidor, encabezándola con datos de tipo MIME (los cuales le ayudarán al browser a reconocer que tipo de información está recibiendo).

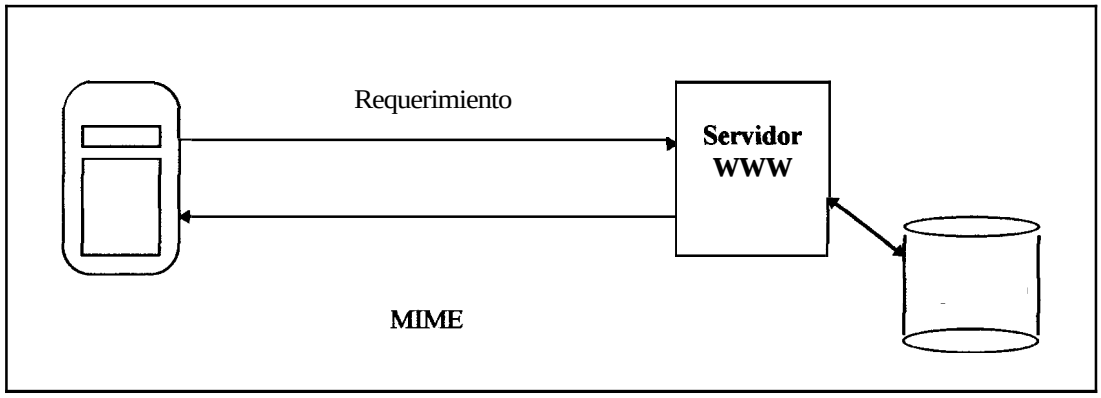


Figura 1.3.
Requerimiento de una simple hoja de Hipertexto

Los browsers poseen la capacidad de interpretar las hojas de hipertexto. Gracias al encabezado que envía el servidor (MIME text/html) los browsers reconocen los elementos HTML propios de la hoja y los publican con el formato adecuado.

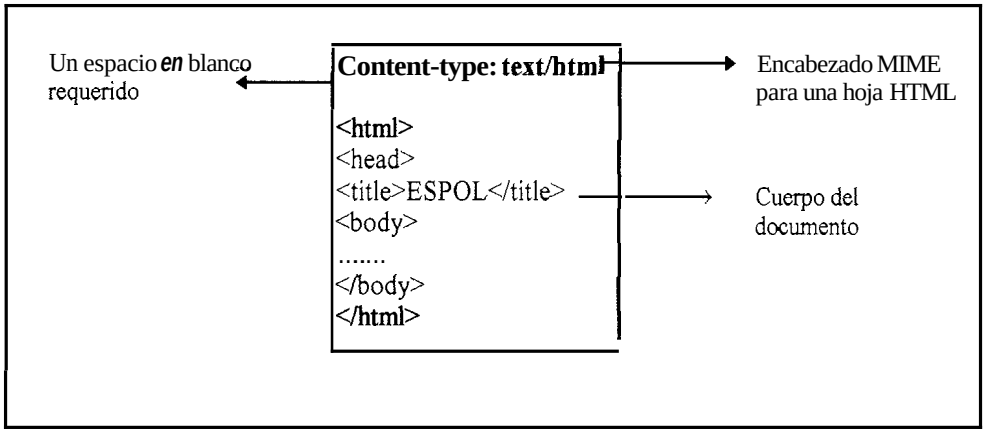


Figura **1.4.**
Hoja de Hipertexto usando MIME

1.5. Interactividad con el servidor: CGIs

Las hojas de hipertexto proveen facilidades de interactividad que pueden activar en el servidor la ejecucion de un programa CGI.

Si un browser requirio la ejecucion de un programa CGI, el servidor esperara hasta que el programa termine su ejecucion y entregue sus resultados, para enviar tales resultados de regreso al browser, los cuales serán encabezados por datos de tipo **MIME** (como mencionamos anteriormente) para su correcta interpretación. De esta manera, el cliente y el programa servidor web se entienden entre si, y gracias a las actualizaciones del protocolo HTTP (como por ejemplo HTTP-NG Next Generation) y de la aparicion de nuevos tipos **MIME** *o* fonnatos en el mundo de Internet, es que cada vez mas se va enriqueciendo y mejorando la variedad de resultados que un usuario final puede obtener de un servidor web.

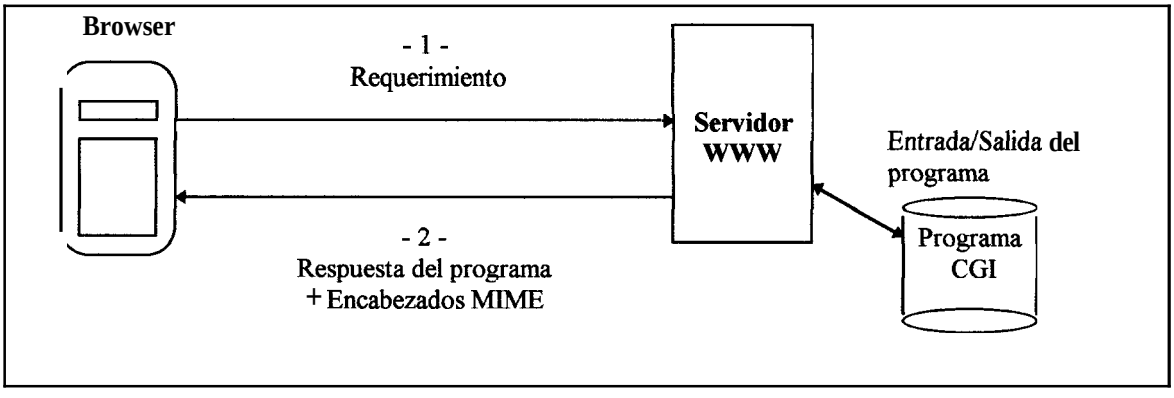


Figura I.5.
Requerimiento de ejecucion de un programa CGI

Ahora bien, hemos hablado de como se realiza la comunicacion entre Cliente/Servidor, para envío y recepción de información, pero no hemos entrado en detalles tales como, el contenido de la informacion. El contenido de la infonnacion que recibe un browser puede variar desde una simple hoja de texto, hasta una muy elaborada hoja de hipertexto, que puede poseer caracteristicas como graficos, frames, Java, Javascript, entre otras propiedades. En la mayoría de los casos, el browser recibira una hoja de hipertexto, la cual podra contener texto, imagenes, sonido, etc. en fin, una gama de propiedades que se conocen con el nombre de ***hipermedia.*** En otros casos, el browser podra recibir un archivo binario, como por ejemplo una grafica ***o*** un programa, y la acción que ejecute el browser dependera de terceras aplicaciones en caso de que el browser no las pueda presentar (esto es lo que se conoce como los ***helper applications,*** los cuales son un conjunto de apliciones invocadas al momento de tratar de leer un formato de archivo no soportado por el browser).

CAPITULO II

LOS PROGRAMAS SERVIDORES WEB

2.1. Los programas servidores web en el modelo Cliente/Servidor

Los programas servidores web son aquellos que atienden los requerimientos de los clientes, por lo cual, su creación requiere de conocimientos de programacion en ambientes de multiples usuarios y multiples tareas con el objeto de hacer un uso eficiente de los recursos del Sistema Operativo que utilizan.

Afortunadamente, existen implementaciones de programas servidores de web que son de libre distribución y lo único que hay que aprender de ellos, es cómo configurarlos de la mejor manera posible.

Los programas servidores web, generalmente corren en platafonnas multitarea, como Unix, NT, OS/2, AS/400, entre otros. Su funcion es ser el receptor de todos los requerimientos de los browsers, por lo que maneja un repositorio de archivos (hojas de hipertexto o archivos binarios como las imagenes) y/o un conjunto de programas o CGIs, a la espera de ser solicitados. Todo programa servidor web entiende los mensajes HTTP enviados por el cliente (browser) que preestablecidamente se reciben en el puerto TCP número 80 (ver explicación de figura 1.2). En Unix existen dos maneras de ejecutar un programa servidor web: Como parte del superproceso servidor (o superdemonio) conocido como INETD o en modo Standalone, lo cual significa que es invocado a traves de uno de los scripts de inicializacion de la maquina. Generalmente se prefiere el segundo modo, de tal manera que no sobrecargue más al ya sobrecargado inetd (Inetd es un superproceso servidor de Unix del cual dependen otros procesos como telnet, ftp, finger, etc).

ESPOL 1958 ESCUELA SUPERIOR POLITECNICA DEL LITORAL BIBLIOTECA CENTRAL

2.2. Funcionamiento de los programas servidores web

La manera en la que fueron programados los primeros servidores web, era mucho mas simple que las actuales tendencias de programacion de los mismos. Los primeros servidores creaban un proceso por cada requerimiento de información que los usuarios hacían con su browser a los cuales se los llama procesos esclavos. Esto, además de

consumir memoria, hacia un tanto lento el tiempo de respuesta de un servidor, por la demanda que exigía el crear cada vez, un nuevo proceso. Para evitar esto, se crearon dos nuevas técnicas: *Las conexionespersistentes* y la *pre-inicializacidn de procesos.*

Mediante *las conexiones perszstentes,* un programa servidor web mantiene solo un camino de comunicacion con el browser a traves del cual envia la mayor cantidad de información posible (fig. 2.1, lado derecho). Las imagenes incrustadas en el texto o los comandos SSI (ver sección 3.6.1) ya no ocasionan la creación de caminos de comunicacion (sockets) adicionales. Pero, a pesar de que el programa servidor web tiende a mantener siempre una conexion persistente con el browser, existen browsers como Netscape que permiten controlar el "número de conexiones" con el programa servidor web, con la finalidad de acelerar el tiempo de carga de una pagina de hipertexto. Esto puede llegar a consumir grandes cantidades de memoria en el sistema del lado del servidor web. Por tal motivo, algunos programas servidores web pueden fijar un maximo de procesos creados a la vez. De esta manera *no se satura la memoria del sistema y se evita una posible caida del mismo.*

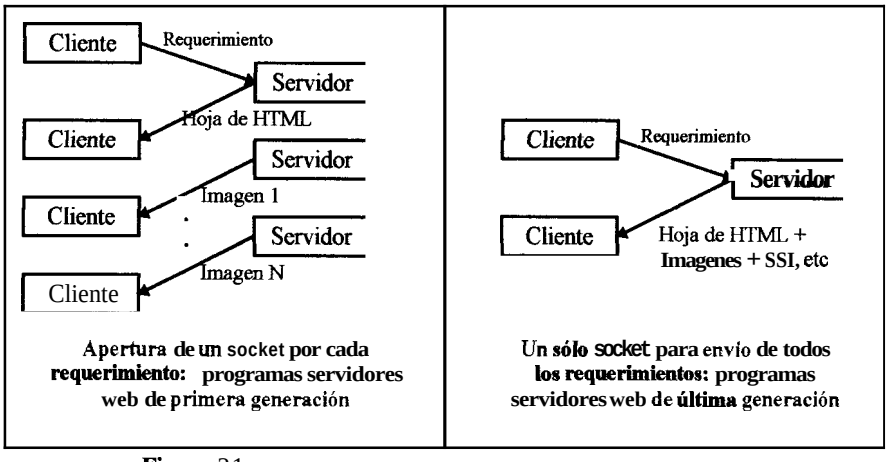


Figura 2.1
Programas servidores web cony sin conexion persistente

Mediante la *pre-inicialización* de *procesos* se predefine o preubica un número determinado de procesos justo en el momento del arranque del servidor principal o proceso maestro (fig. 2.2.). De esta manera, cuando un requerimiento del usuario final llega, el proceso que lo atendera, ya ha sido creado en memoria, por lo que no existirá un retraso por crear un nuevo proceso, sino que la respuesta del servidor viajara inmediatamente de regreso.

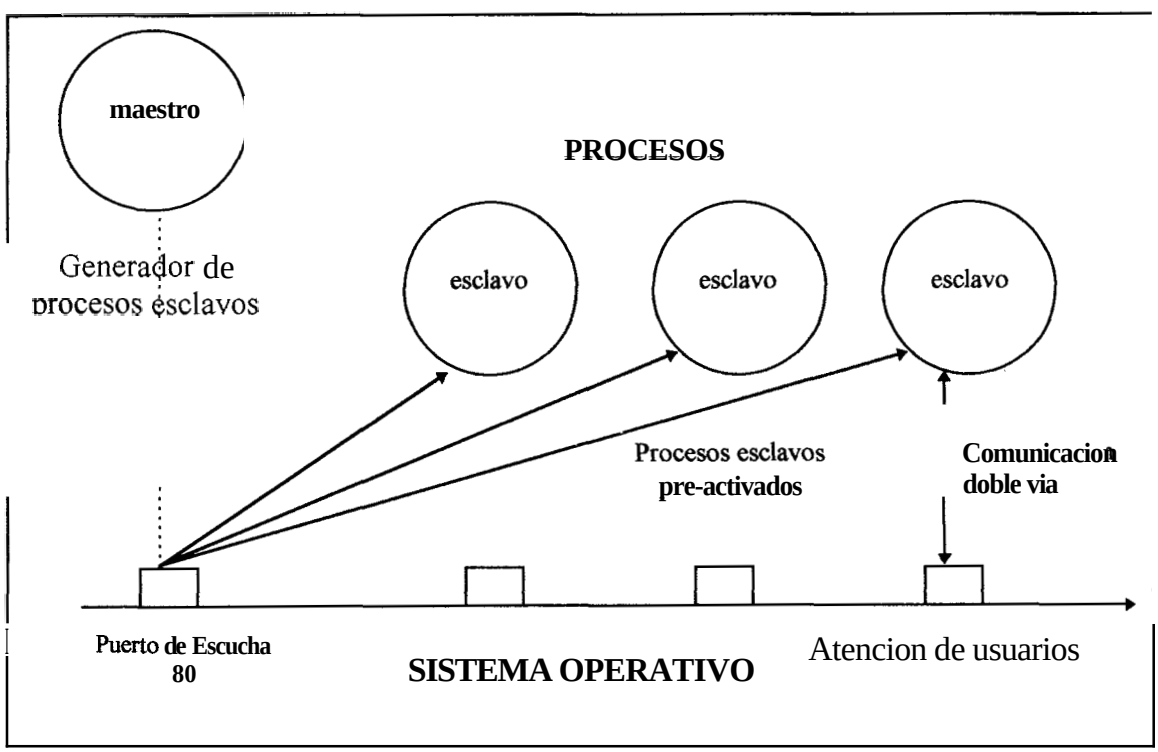


Figura 2.2
Funcionamiento de un programa servidor web
Fuente:Commer; Internetworking with TCP/IP, vol III

2.3. HTTP como protocolo de comunicacion cliente/servidor

Cuando hace muchos años atras, alguien deseaba dar a conocer a la comunidad de Internet información textual y gráfica, solo tenía como recurso acudir a los USENET o grupos de discusion , hacer una copia de sus archivos a otros sitios de FTP o utilizar el GOPHER (Sistema de información basado en menues).

Ninguno de estos medios, solos o en combinación, daban la posibilidad de colaborar a nivel mundial con un ambiente de hipermedia abundante en recursos.

Pero esto cambio al empezar los 90, cuando 'Tim Berners-Lee implement6 el protocolo **HTTP** (HiperText Transfer Protocol) en el CERN (Centro Europeo de Investigaciones Nucleares, Suiza) el cual permitía que los recursos de hipermedia disponibles localmente, se pudieran dar a conocer mundialmente.

HTTP, cuya version actual es la 1.0, posee algunas caracteristicas importantes, que se detallan a continuación:

---

[1] **Tim Berners Lee inventó el WWW a finales de** 1990 **mientras trabajaba en el CERN en Génova, Suiza. El escribio el primer programa servidor web en una plataforma** NeXT. **Es graduado en la Universidad de Oxford, Inglaterra. Actualmente es director del** W3Consortium. Email: timbl@w3.org

1. Un esquema de direccionamiento comprensible:

HTTP posee un formato de direcciones fácil de entender, aprender y recordar. El formato es el siguiente:

http://nombre-máquina:número-puerto/path/archivo.html

URL (siglas en ingles que corresponden a Universal Resource Locator), es el nombre oficial dado a este esquema de direcciones HTTP.

Expliquemos un poco la estructura de un URL:

La primera parte del URL es el protocolo que se está utilizando para comunicarse, esto es, el HTTP , es por esto que se escribe: **http://** .

Dependiendo de la aplicacion que se desee ejecutar en el browser, se pueden poner otros nombres de protocolos, tales como:

ftp:// (accesa a sitios de FTP anonimo), **gopher://** (Accesa a sitios de GOPHER), **telnet://** (accesa a sitios via telnet; requiere un Helper application), **wais://** (accesa a sitios de wais o Wide Area Information System), **news:** (accesa a sitios de News, o noticias), **https://** (Accesa a sitios con seguridad activada: Secure Socket Layer), entre otros.

La segunda parte del URL es el nombre del servidor al que se va a accesar. En este caso, es representado por **nombre-máquina**. El nombre tiene que encontrarse registrado a nivel de DNS (Base de datos de nombres de maquinas

que tiene alcance mundial, y que se encuentra repartida jerarquicamente, en los llamados "Servidores de DNS") o puede ser la direccion IP (direccion numérica del servidor) del servidor de web al cual se desea accesar.

La tercera parte del URL es el puerto en el cual el programa servidor web esta atendiendo los requerimientos. Casi todos los programas servidores web atienden los requerimientos en el puerto 80 (esto es un estándar preestablecido por IANA - Internet Assigned Numbers Authority- coordinador central para la asignacion de valores únicos a los parámetros de los protocolos usados en la Internet). Sin embargo, pueden existir ciertos servidores que atiendan en otros puertos. Además, el especificar otro puerto, sirve en casos en que se desee realizar pruebas con un programa servidor de web y no se tengan los permisos de sistema operativo suficientes para atender los requerimientos en el puerto 80 (Los puertos que van desde 0 hasta 1024 solo pueden ser asignados por el administrador del sistema operativo).

La cuarta parte del URL es el path o **ruta** en la cual se puede hallar el archivo en el servidor. Esta ruta es referencial, esto quiere decir, que el directorio especificado no se basa en el directorio raiz del sistema (En Unix es /) , sino en el

directorio raiz del servidor de web (definido en los archivos de configuración del servidor).

Finalmente, se encuentra el archivo al cual se está accesando. En este caso es representado por **archivo.html.** La extension de este archivo indica que es una Hoja de Hipertexto, pero puede accesarse tambien a otros tipos de archivos, tales como: imagenes, sonidos, programas, etc.

2. Una representación extensible y abierta de tipos de datos:

Cuando un cliente envía una transacción al servidor, Se añade un encabezado que es compatible con el **Standard Internet E-mail Specification (RFC822),** debido a que incluye campos como From (que especifica desde donde se hace el requerimiento). Cuando el servidor HTTP transmite la información de regreso al cliente, este incluye un encabezado tipo **MIME (Multipart Internet Mail Extension),** para decirle browser, que tipo de dato sigue al encabezado.

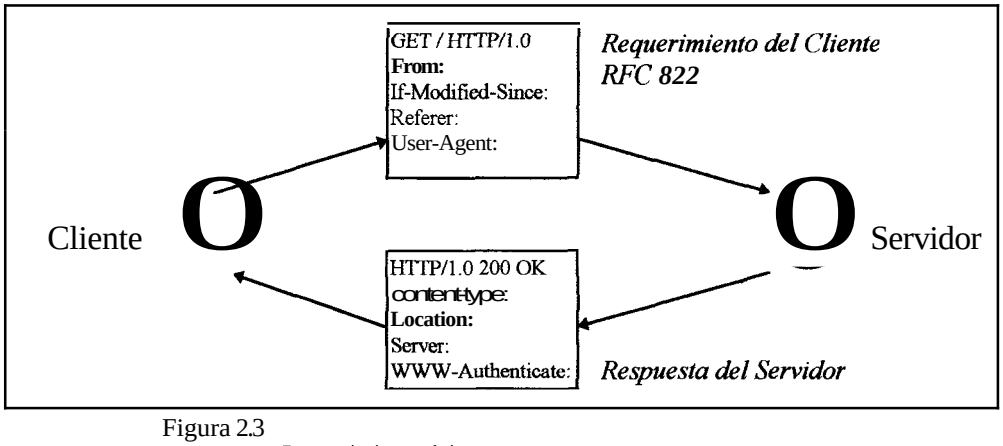


Figura 2.3
Formato del Requerimiento del Browser y Respuesta del Servidor
Fuente: Tim ***Bemers-Lee, RFC 1945 HTTP 1.0***

El servidor no necesita tener la capacidad de interpretar un tipo de dato; el servidor le puede pasar los datos de regreso al cliente, y la interpretación depende de si el cliente posee la utilidad apropiada (visor de imagenes, visor de películas), correspondiente a ese tipo de datos.

3. HTTP es Stateless

Stateless significa que, luego que el servidor ha respondido al requerimiento del cliente, la conexion entre cliente/servidor **es cerrada.** Stateless tambien significa, desde el punto de vista del desarrollador del programa servidor web, que no **"hay memoria"** en el servidor acerca de los datos de los clientes. Es decir, no se mantiene una bitácora (archivos log) del estado del enlace en un momento dado.

4. Es Rapido

El objetivo de Tim Berners-Lee fue lograr un ciclo de respuesta de hipermedia del orden de los [2]100 milisegundos. Es por esto que hizo que los programas servidores web cerraran la conexion inmediatamente despues de haber atendido

---

[2] **Ref.** HTML **& CGI Unleashed,** 1ra edición, capítulo 19, **pag** *379*

el requerimiento del cliente. Entonces, cualquier demora percibida se deberá generalmente a la congestion de la red.

5. Su desarrollo futuro es abierto.

Hoy en dia, Tim Berners-Lee encabeza el World Wide Web Consortium, o W3C, el cual provee un foro abierto para el desarrollo en diferentes areas. Adicionalmente, Netscape Communications Corporation ha desarrollado un esquema de seguridad, llamado el **"Secure Socket Layer" y** ha publicado la especificación SSL de forma abierta, es decir, disponible para el publico.

6. Sus debilidades son bien conocidas

Un ejemplo intrigante constituye el hecho de que el protocolo HTTP 1.0 a menudo tiene problemas de rendimiento del lado del servidor y en la red, ya que la tasa de transmision de los documentos de hipertexto disminuye al aumentar la carga en el servidor y en la red. [3]Simon Spero ha publicado un reporte de lo que

---

[3] Simon Spero es de origen británico. Actualmente labora en el EIT (Enterprise Integration Technologies) y ha colaborado a traves de la propuesta de su protocolo HTTP-NG en el desarrollo de comercio electrónico via Internet. Email: ses@tipper.oit.unc.edu

el W3C llama **"HTTP Next Generation" o HTTP-NG** que es una propuesta para mejorar el protocolo HTTP.

La diferencia entre el protocolo HTTP 1.0 y el protocolo HTTP-NG propuesto, es que mientras HTTP 1.0 crea una nueva conexion por cada requerimiento, HTTP-NG crea una conexion simple que puede ser usada por multiples requerimientos. De esta manera, la conexion cliente/servidor no se cierra inmediatamente (como en el caso de HTTP 1.0), sino que continua abierta durante el transcurso de la sesion (mientras el usuario final siga conectado al servidor web). Con esto, logramos una disminucion del número de procesos activos y una mayor velocidad de respuesta del lado del servidor.

2.4. Medición del rendimiento de un programa servidor we6

La metrica comúnmente empleada para medir el rendimiento de programas servidores web es el **número de conexiones por segundo** que estos pueden manejar. No hay ninguna manera estándar para medir esta cantidad.

Por ejemplo, la tabla que se muestra a continuación (Tabla I), muestra el número de conexiones que fueron atendidas por varios programas servidores web, en un período dado, con un número X de usuarios simultáneos.

Este estudio fue realizado por [4]Robert E. McGrath en el Departamento de Computación y Comunicaciones del National Center of SuperComputing Aplications (NCSA), en la Universidad de Illinois (Urbana, Champaign, EEUU). Para el efecto, utilizó un computador HP 735, 96 Mb en RAM. en una red de 100 Mbps en un anillo FDDI.

	Numero de Clientes				
Servidores	1	2	3	4	8
v1.3	17132	19576	19736	19767	19780.....
CERN 3.0	18109	22526	22730	22423	22781
v1.4, fork	20599	25159	25267	25482	25410
Netsite	26386	33677	35276	35629	36189
v1.4, pass fd	27435	36609	38364	38778	38572

Tabla I.
Requerimientos atendidos en 7 min por varios programas servidores web
Fuente: NCSA (http://hoohoo.ncsa.uiuc.edu)

---

[4] **Robert E. McGrath trabaja en el National Center for Supercomputing Applications (NCSA) de la Universidad de Illinois (Urbana, Champaign). Co-autor del libro "Web Server Technology: The Advanced Guide for World Wide Web Information Providers". Email: mcgrath@ncsa.uiuc.edu**

La primera fila representa el nhrnero de clientes simultáneos que realizan los requerimientos. La primera columna representa los diferentes programas servidores web estudiados. Los nhneros de la region mas clara, representan el número de requerimientos atendidos en 7 minutos para una carga de usuarios dada.

Existen tecnicas que sirven para mejorar las conexiones/segundo de un programa servidor web. Una de ellas es la pre-inicializacion de procesos (ver sección 2.2) o tambien llamada pre-forking (que viene de la palabra fork, que en ambiente Unix significa generar un proceso esclavo a partir de un proceso maestro).

Los dos últimos programas servidores web (v1.4,pass fd y Netsite) son servidores con pre-fork. Los nhneros reflejan que son capaces de soportar mas requerimientos que los otros tres servidores (v1.4 fork, CERN 3.0 y v1.3), que son simples servidores sin pre-fork.

En particular, el uso de fork sirve a cargas pequeñas, mientras que los programas servidores web con pre-forking aumentan la respuesta bajo cargas mas elevadas. En otras palabras, si se posee un programa servidor de web concurrido, es recomendable utilizar programas servidores que posean pre-forking o pre-inicializacion de procesos a usar uno que no lo posea.

credito, etc. Esta informacion es transmitida desde el browser hasta el programa servidor web. En teoria, una tercera persona puede interceptar esta informacion en algún punto en la red entre el browser y el servidor. Para prevenir esto, se ha creado el protocolo denominado como SSL (Secure Socket Layer), que es un protocolo implementado por Netscape Corporation **(http://home.netscape.com)** para poder realizar transacciones seguras en el Web.

SSL se basa en el uso de un "Sistema de clave publica". Antes de explicar el funcionamiento de SSL, definamos algunos terminos a utilizar:

a. Clave pdblica: como su nombre lo indica, es una clave que puede ser conocida por todos. Esta clave puede ser usada por cualquiera para encriptar un mensaje y enviarnoslo. En el caso del servidor web, el browser encripta la informacion con la clave publica del servidor y se la envía a dicho servidor.

b. Clave privada: Es una clave generada aleatoriamente a partir de la clave publica, la cual me permite desencriptar los mensajes enviados por otros, que han utilizado mi clave publica para encriptar tal mensaje. En el caso del servidor web, el servidor desencripta la informacion enviada por el browser (encriptada con la clave publica del servidor), utilizando la clave privada del servidor.

Tanto la clave publica como la privada sirven para que el programa servidor de web y el browser realicen ***la conexión inicial.*** La información que es enviada por el browser al servidor, encriptada con la clave publica de tal servidor, no son los "datos" a transmitir, sino una "tercera" clave que se conoce como "clave de sesion", con la que si se encriptaran los "datos" a transmitir (passwords, números de tarjetas de credito, etc)

c. **Clave de la sesión**: Es la clave transmitida en la conexion inicial entre dos partes y que permite encriptar los datos a ser transmitidos. Esta clave cambia cada vez que se establece una nueva conexion entre las partes comunicantes. A nivel de servidor web, la clave de sesion es la clave transmitida por el browser en forma encriptada (con la clave publica del servidor) y recibida por el servidor (y desencriptada con la clave privada del servidor), de tal manera que esta sea la clave con la que browser y servidor encriptaran los datos a ser transmitidos (passwords, números de tarjetas de crédito, etc).

El esquema de funcionamiento de los SSL, toma en consideración estos tres tipos de claves: publica, privada y de sesion. Veamos la siguiente figura:

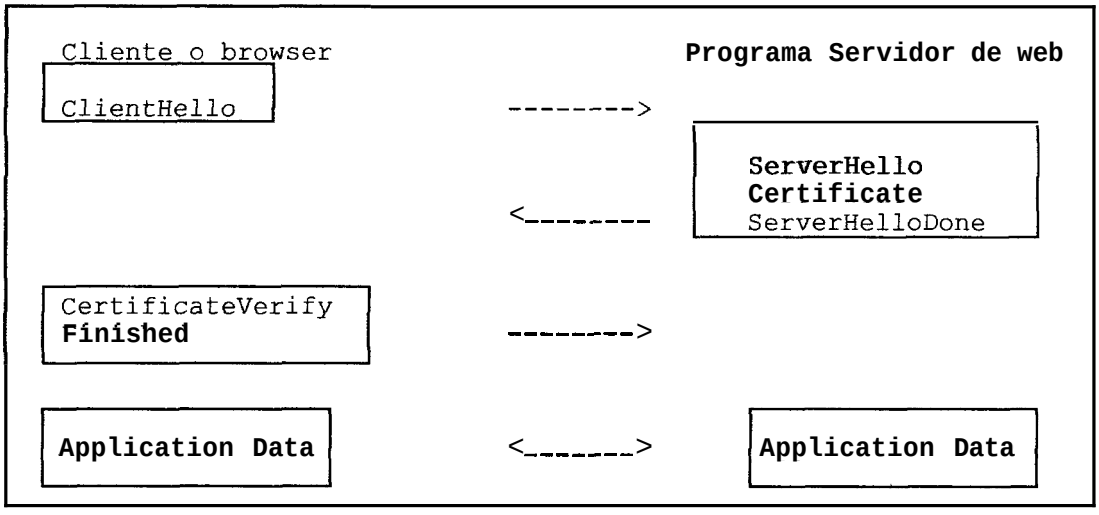


Figura 2.4
Funcionamiento del Secure Socket Layer
Fuente: Alan O. Freier, Netscape Communications

Como se ve en la figura anterior, el browser o cliente es el que inicia la conexión con un mensaje al que denominaremos **ClientHello**. Luego, el servidor responde a este mensaje enviando una respuesta a la que llamaremos **ServerHello**; además de ésto, el servidor envía lo que se conoce como su "certificado" o **Certificate** (el cual es una pieza de información encriptada que permite saber al browser que organización está accesando). Finalmente el servidor envía el mensaje **ServerHelloDone**, que indica la finalización del inicio de la sesion. El servidor quedara a la espera de la contestación del browser . El browser, por su parte, recibe y verifica (mensaje **Certificateverify** de la figura 2.4) con

un tercero (una autoridad conocida como '"Certificate Authority", que tiene en su base de datos la firma digital del servidor web), si el servidor web es quien dice que es. Una vez que el browser verifica la autenticidad del servidor web, encripta la clave de sesion con la clave pública del servidor (enviada por dicho servidor en su "certificado") y la envia al programa servidor web. El algoritmo utilizado para el encriptamiento se conoce como [6]RSA (algoritmo que utiliza una clave de sesion de 48 bytes). Luego, el browser envia un mensaje denominado Finished, el cual indica al servidor que la transmision de la clave de sesion y el mecanismo inicial de autenticacion (que lo constituyen los "certificados"), fueron exitosos. El mensaje Finished es el primer mensaje encriptado con la clave de sesion. A partir de este momento, cualquier información enviada del browser al servidor (en la figura 2.4 se la denomina "application data"), sera encriptada con la clave de la sesion.

---

[5] En la actualidad existen diversas empresas que actúan como Autoridades de Certificación, tales como Verisign (http://www.verisign.com), Thawte Consulting (http://www.thawte.com), AT&T (http://www.att.com),entre otros.

[6] RSA es un algoritmo de encriptamiento basado en un llave publica y privada, cuyo nombre es la union de las iniciales de sus autores: Ronald Rivest, Adi Shamir y Len Adleman.

2.5.2. Htpasswd NCSA

Htpasswd de NCSA (National Center of Supercomputing Applications de la Universidad de Illinois, Urbana, Champaign), es un mecanismo sencillo de protección de los directorios en los que un servidor web guarda su inforrnacion. Se basa en el uso de archivos que controlan el acceso a un directorio determinado.

En este tipo de autenticacion, la clave es pasada sin encriptar desde el cliente o browser hasta el servidor (pero esto no significa que la clave es pasada en formato texto). La clave se envía en formato uuencode (uuencode es un utilitario de Unix que permite codificar la informacion, utilizando un estándar que lleva su mismo nombre). Este mecanismo de autenticación es equivalente al usado en las conexiones Telnet (que también utiliza esta codificación).

Para proteger un directorio con el mecanismo htpasswd, se realiza lo siguiente:

a. Se crea un archivo llamado .htaccess en el directorio que se desea proteger. Este archivo tiene el siguiente formato:

```
AuthUserFile
/otherdir/.htpasswd
AuthGroupFile /dev/null
AuthName ByPassword
AuthType Basic

<Limit GET>
require user httpd
</Limit>
```

Figura 2.5
Archivo .htaccess usado *en* el esquema de seguridad htpasswd de NCSA
Fuente: http://hoohoo.ncsa.uiuc.edu/docs/tutorials/userhtml

En la figura 2.5 se observa que se debe especificar el directorio que va a ser protegido en la directiva **AuthUserFile**. Tambien se especifica el nombre del usuario y grupo que tiene acceso a tal directorio (directivas **Require** y **AuthGroupFile** respectivamente). Luego, la directiva **AuthName** especifica el nombre *o* título de la ventana que aparecera en el browser para pedir la clave al usuario. Finalmente, el tipo de autenticacion es definido por **AuthTypeBasic**, lo cual significa que la ipformacion sera codificada con formato uuencode.

b. Crear el archivo .htpasswd

Este archivo contendra el password asignado al usuario **httpd** (especificado en la directiva **Require** del grafico 2.5). Para generar este archivo, se debe correr una aplicacion llamada htpasswd, que viene con los archivos de distribución del programa servidor web (se la puede encontrar en el directorio /home/httpd/support) del servidor web de la ESPOL.

El comando "htpasswd se usa de la siguiente manera:

```
htpasswd -c  .htpasswd httpd
```

De este modo, se crea el archivo .htpasswd, que contendra al usuario httpd y su clave (pedida al momento de ejecutar "htpasswd).

Finalmente, se prueba entrando al directorio que se ha protegido, utilizando un browser. El browser ahora requerira un nombre de usuario y una clave para ingresar a la información del directorio.

2.6. Servidores Proxy

Los Servidores proxy actúan como un elemento intermedio entre el cliente y el servidor web. Un servidor proxy es aquel que redirigirá un requerimiento originado en un browser, hacia un programa servidor de web. De esta manera, se obtienen dos ventajas:

1. Se puede restringir a los browsers de una red que trabaje con proxys, el acceso a ciertos sitios no deseados ya que se pueden controlar los requerimientos de los browsers.

2. Se almacena en una memoria temporal del disco del servidor proxy (llamada memoria cache), los documentos pertenecientes a un servidor web remoto. De esta manera, si un usuario accesa a un documento previamente guardado en la memoria cache, este usuario lo accesara como si fuese un documento local, con la consiguiente disminucion del tiempo de respuesta para el usuario final. Un documento se guardara en memoria cache, la primera vez que sea accesado.

Aunque se introduce un retardo del requerimiento del browser, debido a la inserción del servidor de Proxy entre el browser y el servidor, esto ocurre solo la primera vez que se realiza tal requerimiento, puesto que los siguientes accesos a la misma información, serán recogidos del cache del servidor proxy.

En la figura **2.6.** se muestra la secuencia que sigue un servidor proxy para atender los requerimientos de un browser si el documento no se encuentra en cache. En la actualidad, los servidores proxy son utilizados en esquemas de seguridad para protección de las llamadas "intranets" *o* redes internas a una organización, aisladas de la Internet, pero que utilizan la tecnologia y protocolos existentes en ella. Su utilidad en las intranets radica en el hecho de que, al ser posible para un

servidor proxy hacer un requerimiento en lugar de un cliente o browser, el servidor proxy se convierte en un aislante de los clientes dentro de intranet.

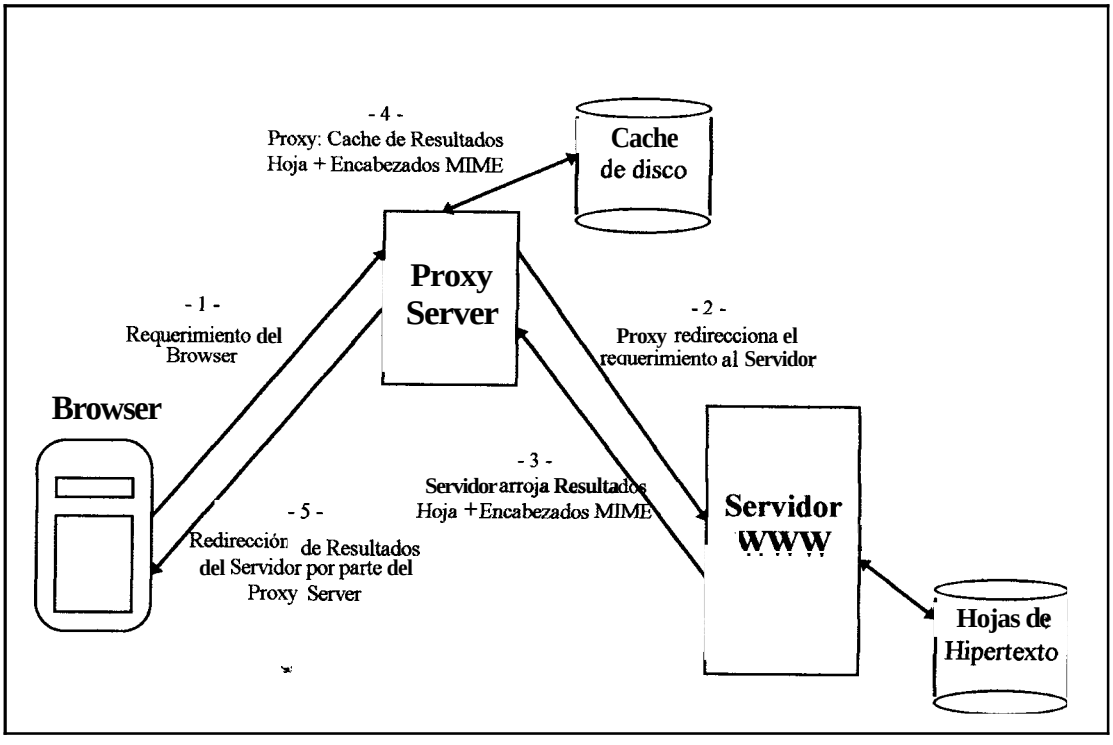


Figura 2.6.
Funcionamiento de un *Proxy Server* incluido entre el Cliente y el Servidor de Web

CAPITULO III

LOS CGIs

3.1. Ejecutando terceras aplicaciones a traves del servidor web

Los CGIs (Common Gateway Interfaces), son aquellos programas que son activados por los programas servidores de web para procesar informacion que es enviada por los browsers debido a un requerimiento hecho por un usuario final. Se diferencian de otros programas que corren en el servidor ya que son capaces de recoger la informacion, que almacenadas en variables, viaja del cliente al servidor .

Los CGIs pueden ser elaborados en cualquier lenguaje de programacion que sea capaz de leer el STDIN o STDOUT de otros procesos. El STDIN o Standard Input es el medio por el cual un programa recibe la informacion de otro que se ha ejecutado previamente. El

STDOUT es el medio por el cual un programa envía informacion a otro que se ejecutará subsiguientemente.

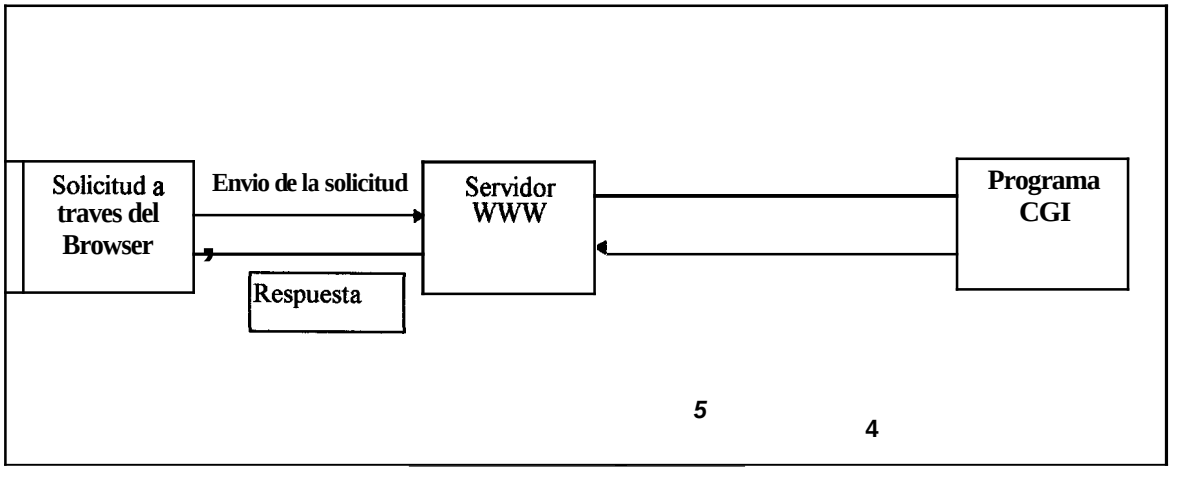


Figura 3.1
Esquema detallado de la comunicación Cliente-Servidor-CGI

Por ejemplo, cuando un usuario final llena los datos de una FORMA (conjunto de campos que un usuario final ingresa en una hoja de hipertexto) y activa la transmision (paso 1 de la figura 3.1), estos viajan primero al programa servidor web, quien a su vez los redirige hacia el CGI a traves de su STDOUT (standard output, Paso 2, fig. 3.1). El CGI recibe la informacion a traves de su **STDIN** (standard input, Paso **3,** fig. **3.1). Los** resultados de la transacción son enviados a traves del STDOUT del programa CGI **(** Paso **4,** fig. **3.1)** y recibidos por el **STDIN** del servidor web (Paso 5, fig. **3.1)** que redirigira la salida hacia el browser o usuario final (Paso 6).

Por lo general, la salida de todo programa es la ***pantalla del computador (stdout default)*** y la entrada de los mismos es ***el teclado (stdin default).*** Los CGIs aprovechan esto, haciendo que la salida o entrada de los programas provengan de otros programas.

Tanto *C* como Perl (ver sección 3.5) son los lenguajes de mayor uso en la programacion de CGIs , puesto que son rapidos de ejecutar y fáciles de entender. Perl ofrece facilidades para el manejo de archivos y bases de datos (Oracle, entre otras), además de ser una herramienta poderosa en la manipulación de archivos y el manejo de los recursos del S.O.

3.2. *Como programar los CGIs*

Existen dos métodos de programacion de CGIs: POST y GET. Estos metodos sirven para enviar informacion procedente de una FORMA en el servidor web, desde el cliente (o browser) al programa CGI (pasando por el programa servidor web), y se diferencian entre si, en la manera como envian tal informacion. Mientras el metodo POST envía la informacion a traves del STDIN del programa CGI, el metodo GET la envia a traves de variables de ambiente del sistema operativo. La figura 3.2 ilustra lo que se ha explicado. Notese que la entrada de la informacion (DATOS) es la misma; la

diferencia de los metodos radica en la forma en la que el servidor web transmite la informacion al programa CGI.

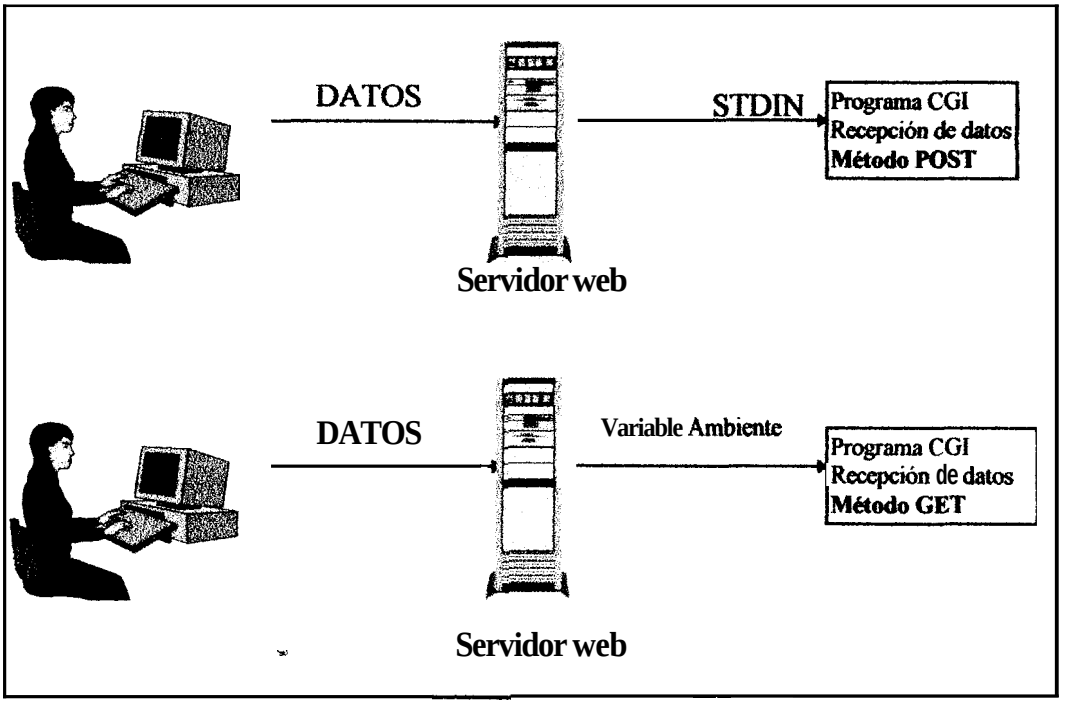


Figura 3.2.
Métodos POST y GET de envio de información al CGI

MÉTODO POST

. POST es el metodo mas empleado ya que permite recibir mediante el STDIN del programa CGI, la informacion proveniente de una forma del servidor web, sin resticciones de tamaño. La informacion que viaja a traves del metodo POST es

codificada, pues se deben substituir los caracteres que no sean numéricos o alfabeticos, por otro tipo de caracteres. Por ejemplo, los espacios en blanco se substituyen por un mas (+), mientras que caracteres especiales como el slash (/), se substituyen por su código hexadecimal, acompañado del signo de porcentaje (%2F). Esto se debe a que cuando se arma el URL con los datos ingresados, no pueden existir (en el URL), espacios en blanco o caracteres especiales, debido a que el estándar HTTP 1.0 no lo permite.

Un ejemplo de esto, es cuando buscamos informacion en las maquinas de busqueda (Altavista, por ejemplo: **http://www.altavista.digital.com).** Si deseamos obtener informacion sobre "cliente/servidor", ingresamos tales datos, y obtenemos el siguiente URL:

```
http://www.av.com/cgi-bin/query?pg=q&what=web&fmt=.&q=cliente%2Fservidor
```

Esto se interpreta de la siguiente manera:

En el directorio **"cgi-bin"** del servidor de web www.av.com, se ejecuta un programa denominado **"query",** el cual toma como parámetros a **"pg", "what", "fmt"** y **"q"**. El valor de cada uno de ellos es: **"q", "web", "."** y **"cliente%2Fservidor"** respectivamente. Para asignar a un parámetro su valor, se utiliza el signo igual **"="**. Para separar las parejas "parámetro=valor", se utiliza el signo ampersand **"&"**. De esta manera el CGI recibe la informacion, por lo cual , deberá (el CGI) tener una ***rutina de descodificación de la informacidn*** que le permita separar los datos que vienen en el

URL. Estas rutinas son hechas por el mismo programador o pueden ser obtenidas de terceros. (Por ejemplo, Thomas Boutell -http:Nwww.boutell.com- tiene un conjunto de librerías hechas en C que implementan la descodificación del URL).

El siguiente es un ejemplo de una forma de ingreso de datos en el servidor web y su codigo en HTML. Luego de esto, se presenta un ejemplo del codigo del CGI que permite leer los datos de la forma. El metodo usado es el "metodo POST".

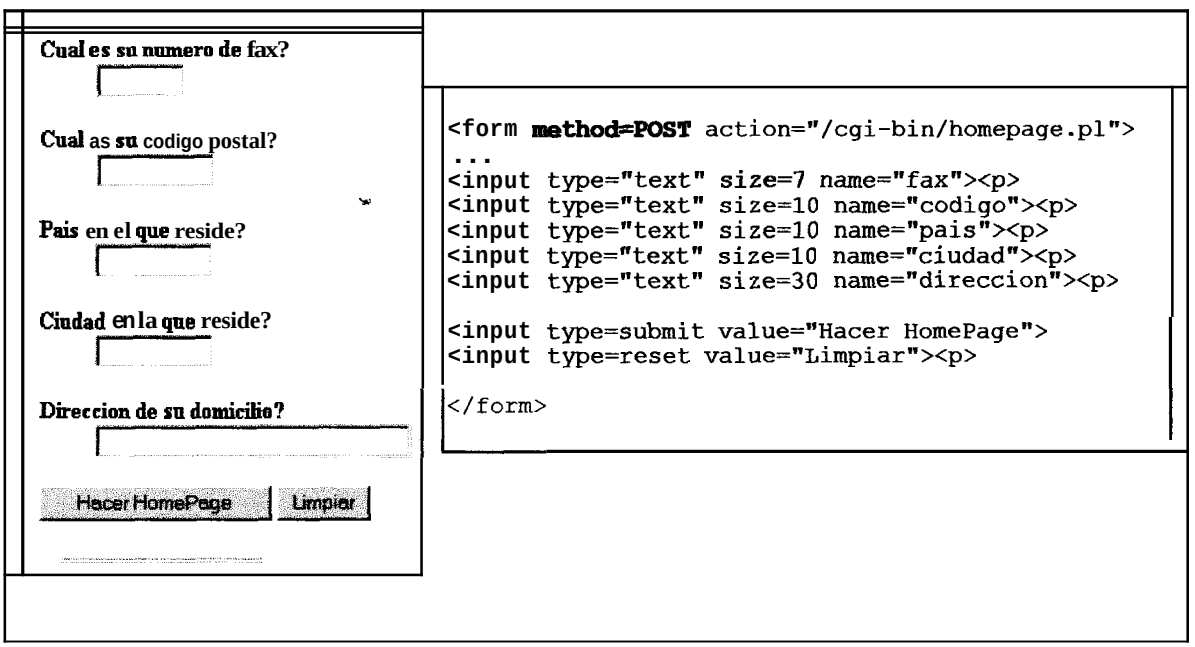


Figura 3.3.
Forma confeccionada utilizando el método POST y su código HTML
Fuente: Archivo /ESPOL/homepage.html del Web de la ESPOL

La figura 3.4. es el codigo en Perl del programa CGI que permite leer los datos ingresados en la forma anterior. Notese que los datos son recogidos a traves del STDIN del programa; La variable CONTENT–LENGTH, es una variable enviada por el programa servidor web que indica el tamaiio en bytes de los datos que se están ingresando.

```
#recepción de información a traves del STDIN
read(STDIN, $buffer, $ENV{'CONTENT LENGTH'));
#Rutina de decodificacion de la info
@pairs = split(/&/, $buffer);

     foreach $pair (@pairs){

          ($name, $value) = split(/=/, $pair);
          ...
          $FORM{$name} = $value;
     }
```

Figura 3.4.
Código en Perl de un CGI implementado con el metodo **POST**
Fuente: Archivo /cgi-bin/homepage.pl del web de la ESPOL

MÉTODO GET

El otro metodo empleado se denomina "metodo GET", el cual, tambien codifica la informacion enviada, pero tiene restricciones sobre el tamaiio de la informacion transmisible. Esta restricción se debe al hecho de que el metodo GET no utiliza el

STDIN del programa CGI para pasar los datos hacia el mismo, sino que, utiliza lo que se conoce como VARIABLES DE AMBIENTE, para tales efectos. A traves de la variable de ambiente QUERY–STRING, el programa CGI recibe la información del servidor. El inconveniente se produce, porque, esta variable tiene un tamaño finito y el número de datos que acepta es limitado en cantidad. La siguiente es una FORMA que hace una llamada usando el método GET. Se ha tomado como ejemplo la maquina de busqueda conocida como Altavista **(http://www.altavista.digital.com).** Se han tomado los fragmentos mas importantes del codigo HTML de la hoja.

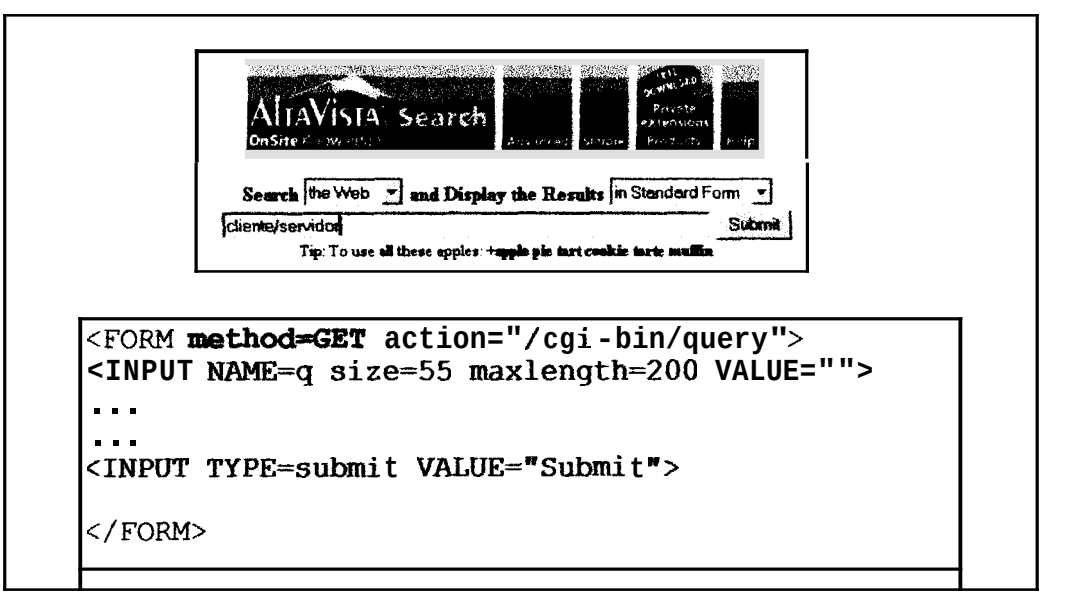


Figura 3.5.
Forma confeccionada utilizando el método GET y su código HTML
Fuente: http://www.altavista.digital.com

La figura **3.6.** es **un** extracto del codigo en Perl que nos permite tomar los datos que nos envía la hoja de la figura 3.5.

```
#Recibir campos metodo GET

 $in = $ENV{'QUERY_STRING'}; #variable de ambiente
 @pairs = split(/&/, $in);

foreach $pair (@pairs) #Rutina de decodificacion de la info
                {
                ($name, $value) = split(/=/, $pair);
                 ...
                 ...
                $FORM{$name} = $value;
                }       #Fin de rutina
```

Figura 3.6.
Código *en* Perl de un CGI implementado *con* el método GET
Fuente: Archivo /cgi-bin/header.pl del Web de la ESPOL

3.3. Consideraciones de Seguridad en los CGIs

De lo anteriormente dicho, podemos obtener algunas observaciones:

La informacion que fluye a traves de todo el mecanismo de los CGIs, es informacion textual, susceptible a ser interceptada y **leída** puesto que viaja como texto (compatibles con el conjunto de caracteres ISO 8859-1). Por tanto, los CGIs de por si, son mecanismos muy inseguros de manejo de informacion, no recomendables para realizar transacciones comerciales, en las que se maneje infonnacion confidencial.

Los CGIs dependen de la información que suministre el usuario final a traves del browser. Sin embargo, no siempre es posible saber si los datos que ha ingresado el usuario final son confiables. Hay el peligro de que un usuario final, conocedor del sistema en el que corre nuestro programa servidor web, envíe "comandos incluidos" en los datos de una forma (Utilizando los simbolos de pipe | o ampersand &, que en Unix ejecutan comandos paralelamente). Esto se puede evitar, tomando en cuenta las siguientes consideraciones:

1. No usar:

system "echo $foo"; ⟶ **Inseguro.** Permite ejecucion de proceso en paralelo

sino:

system "/bin/echo", $foo ⟶ Seguro, no permite ejecucion de un proceso paralelo

2. No usar:

$path = $ENV{'PATH'} ⟶ **Inseguro,** cualquier usuario puede cambiar el PATH

sino:

$ENV{'PATH'} = '/bin:/usr/bin'; ⟶ Seguro, se especifica el **PATH** completo

3. No usar:

open(MAIL,"/usr/lib/sendmail $recipient"); ⟶ inseguro; permite ejecucion de proceso paralelo

sino:

open(MAIL, "/usr/lib/sendmail -t"); ⟶ **seguro** ya que no permite ejecucion de proceso paralelo
print MAIL "To: $recipient\n";

En general, se debe chequear que los campos de una FORMA del servidor web no contengan los siguientes caracteres:

& ; ` ' \ " | * ? ~ < > ^( ) [ ] { } $ \n \r

Figura 3.7.
Ejemplos *en* Perl de CGIs seguros
Fuente: December, John. HTML & CGI Unleashed, 1st Edition

3.4. Tecnicas empleadas en la elaboración de CGIs

3.4.1. Crear Procedimientos genéricos para tareas comunes o repetitivas

En el momento de programar un CGI, nos daremos cuenta que existen algunos procedimientos repetitivos que se pueden separar en funciones. Veamos el siguiente ejemplo:

La figura 3.8 muestra la rutina html_header() hecha en Perl, cuya tarea es imprimir el encabezado de un documento HTML. Recibe como dato el titulo a publicarse. Esta tarea es frecuentemente utilizada cuando trabajamos con CGIs.

```
sub html_header
{
        $document–title=$_[0];
        print "Content-type:text/html\n\n";
        print "<HTML>\n";
        print "<HEAD>\n";
        print "<TITLE>$document_title</TITLE>\n";
        print "</HEAD>\n";
        print "<BODY>\n";
        print "<H1>$document_title</H1>\n";
        print "<P>\n";
}
```

Figura 3.8.
Rutina de **inicio** de documento **HTML hecha** en **Perl**
Fuente: Archivo /cgi-bin/mail5.pl del web de la ESPOL

3.4.2. Utilizar variables globales para las referencias de URL/archivo/path

Es comim mover archivos en el servidor web para realizar cambios al sistema e incorporar nuevas referencias al dominio. Si se definen variables globales para las referencias a directorios, cambiar el script para acomodarlo a una nueva localización, sera mucho mas fácil.

3.4.3. Minimizar la Entrada/Salida de Archivos cuando sea posible

Es muy común para los programadores escribir **archivos** de **configuración** que son leidos al inicio del programa. Estos archivos de configuracion le permiten cambiar rapidamente los parámetros y conducta del programa. Sin embargo, es recornendable que si se tiene un "archivo de configuración", se lo incorpore como una librería de definiciones en otro archivo separado (utilizando ##defineen lenguaje C). Esta es una solución para minimizar la Entrada/Salida hacia el archivo y reducir la cantidad de recursos necesarios para la ejecucion del script.

3.4.4. Siempre estar preparado para entradas incorrectas del usuario final

En las FORMAS no hay control sobre los datos que se pueden ingresar: pueden existir entradas maliciosas o exageradamente grandes.

Para evitar esto, el programa CGI debe validar los datos ingresados. Esto debe constituirse en una política al momento de elaborar los CGIs.

3.4.5. Implementar un bloqueo de archivo y registro

Nunca se sabe si un script (programa CGI) sera ejecutado por dos usuarios al mismo tiempo, y en tal caso, no es dificil que los datos se corrompan o tengan valores indeterminados. Para solucionar esto, existe una tecnica efectiva: Se debe hacer conocer a todos los procesos concurrentes, a traves de un archivo temporal, cuando un recurso esta siendo usado o no. Si el archivo temporal existe, el recurso está siendo usado; de otra manera, no lo esta. Con este artificio sencillo, habremos solucionado el problema, puesto que los procesos concurrentes se iran encolando y seran atendidos secuencialmente.

3.5. Uso de Perl en la elaboración de los CGIs

Perl (Practical Extraction and Report Language) es un lenguaje de programacion, optimizado para la busqueda, extracción de información, e impresion de reportes. Tambien es un buen lenguaje para tareas administrativas del S.O. Este combina, alguna de las mejores caracteristicas de C, sed, awk y sh. La sintaxis de las expresiones de Perl es muy parecida a la sintaxis de C. Es un lenguaje interpretado, puesto que sus programas no necesitan compilacion. Entre algunas de sus caracteristicas tenemos:

1. Perl no limita el tamaño de sus datos *(si se tiene memoria suficiente, Perl puede agrupar todos los datos en una cadena de caracteres simple).*

2. La recursion es de profundidad ilimitada.

3. Perl usa tecnicas sofisticadas de *Pattern Matching* para buscar gran cantidad de datos muy rapidamente.

4. Perl es sensible a mayúsculas

5. Todos los objetos no inicializados son asurnidos que empiezan con un null o valor 0 hasta que son definidos por alguna operación explicita tal como una asignacion.

6. Las tablas utilizadas por arreglos asociativos, crecen tanto como sea necesario.

Perl tiene tres tipos de datos: escalares, arreglo de escalares y arreglos asociativos de escalares.

Las variables escalares siempre empiezan con el simbolo dolar ($), aún cuando se refieren a un escalar que es parte del arreglo. Por ejemplo:

```
$days           # una variable escalar simple
$days[28]       # 29no elemento del arreglo @days
```

Los arreglos completos, son denotados por el simbolo arroba (@):

```
@days           # ($days[0], $days[1],... $days[n])
@days[3,4,5]    # igual a @days[3..5]
@days{'a','c'}  # igual a ($days{'a'},$days{'c'})
```

Los arreglos asociativos son denotados por el simbolo % y sirven para "asociar" dos valores, de tal manera que uno de ellos pueda ser obtenido en base al otro. Por ejemplo:

```
%days=("1","En","2","Feb",...)
relaciona 1 con En , 2 con Feb, etc...
```

Estos son los detalles mas importantes acerca de Perl. Para mayor información, consultar el servidor web oficial de Perl: http://www.perl.com

3.5.1 Problemas en la implementación de CGIs usando Perl

Durante el tiempo que ha durado esta Tesis, se pudieron detectar dos desventajas en la implementación de CGIs usando Perl, que se detallan a continuación:

1. Perl no soporta escrituras aleatorias en un archivo ya existente. Los datos necesariamente tienen que agregarse al final del mismo. No existe una tecnica de manejo de punteros para accesar a cierta posición dentro del archivo y adicionar allí los datos como ocurre en otros lenguajes como C, Pascal ,etc.

2. A pesar de que en los archivos de distribución de Perl se indica que existe soporte para la funcion flock( ), esto no es asi. Esta funcion permite bloquear el acceso a los archivos, lo cual es algo muy importante de realizar en un ambiente multiusuario, ya que nos permite controlar la apertura, lectura o escritura de archivos, por usuarios que los accesan simultáneamente.

3.6. *SSI y NP headers*

3.6.1. Server Side Includes (SSI)

Algunos programas servidores web, permiten a los proveedores de informacion, crear documentos que provean de manera dinámica datos sencillos a los browsers. Tal informacion puede ser la fecha actual, la fecha de la ultima modificacion de la pagina, etc, cuyos valores son cambiantes. A este tipo de informacion (que se incluye dinamicamente en un documento HTML) se le denomina los Server Side Include (o inclusiones del lado del servidor).

Los SSI trabajan de la siguiente manera:

Cuando un programa servidor web recibe un requerimiento para mostrar un archivo con directivas SSI, este debe interpretar el documento HTML junto con las directivas SSI. Solo entonces es retornado al cliente.

A continuación un ejemplo práctico de SSI:

En este ejemplo de la figura 3.9 se observa que dentro del codigo HTML, se ha incluido un comando SSI, el cual nos permitirá conocer cual es la fecha de la ultima modificacion del archivo de hipertexto. El programa servidor de web es el encargado de interpretar tal

comando y enviar la respuesta incluida en la misma posición dentro del documento de hipertexto.

```
<html>
<body>
....
....
....
Ultima modificacion <!--#echo var="LAST_MODIFIED"-->
....
....
</body>
</html>
```

Figura 3.9.
Uso de SSI *en* **un documento** HTML
Fuente: Archivo /docs/footer.html del web de la ESPOL

De los SSI se puede obtener otro tipo de información, tal como:

El nombre del documento actual *(variable DOCUMENT_NAME)*

La fecha actual *(variable DATE_LOCAL)*

La ultima fecha de modificacion y tiempo en el que el archivo actual fue modificado *(variable LAST–MODIFIED)*

El nombre del computador que hace el requerimiento *(variable REMOTE_HOST).*

El nombre del browser que utiliza el usuario final *(variable HTTP_USER_AGENT).*

Además, se pueden ejecutar comandos, cuya salida (STDOUT) se incluya en el documento de hipertexto. Un ejemplo de esto es el siguiente contador de accesos a una pagina de web, el cual permite variar el número mostrado, cada vez que alguien accesa a la hoja de hipertexto.

```
<html>
<body>
...
#Programa contador de accesos ejecutado como SSI
<!--#exec cmd="/home/httpd/cgi-bin/contador/"-->
...
...
</body>
</html>
```

Figura 3.10.
Ejecución de un programa a través de los *Server* Side Includes
Fuente: Archivo /ECUADOR/mapa.html del web de la ESPOL

En este ejemplo, el programa ejecutado es un contador de accesos, el cual es un CGI que va actualizando un número (guardado en un archivo), que se actualiza cada vez que un usuario final accesa a la pagina.

3.6.2. NP Headers(Encabezados NP)

Por norma general, la salida (STDOUT) de los programas CGIs, debe pasar siempre por el programa servidor de web, ya que este servidor debe agregar un

encabezado de tipo MIME a dicha salida, para que pueda ser correctamente interpretada por el mismo servidor. Por ejemplo, si un programa CGI nos da una salida como la siguiente:

```
Content-type: text/html
<html><body>
<h1>Bienvenidos</h1>
</body></html>
```

El programa servidor de web, agregara al encabezado "Content-type", lo siguiente:

```
HTTP/1.0 200 OK
DATE: Tue 17 Dec 1996 12:24pm
```

Esto es debido a que el estándar HTTP 1.0 así lo especifica. Para evitar que el programa servidor web agregue este encabezado al ejecutar un CGI, se utilizan los NPH o Non Parse Header scripts, que son programas CGI cuyo nombre empieza con las letras **"nph-"**, los cuales permiten que el mismo programa CGI, envíe el encabezado completo, haciendose innecesaria la pasada a traves del programa servidor web. En otras palabras, el programa CGI tendra ahora la siguiente salida:

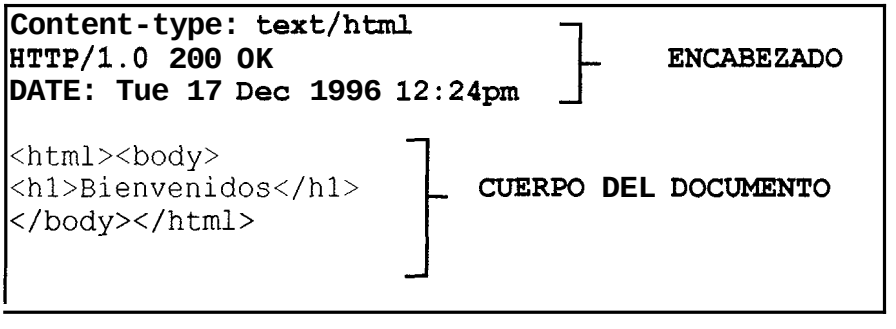


Figura 3.11.
Salida de un programa CGI utilizando No Parse Headers

CAPITULO IV

PROCESOS DE DESARROLLO DE UN SERVIDOR WEB

4.1. Etapa de Planificacion en un servidor web

La planificacion es un aspecto crucial del desarrollo del web debido a que es cuando se deben hacer decisiones que afectan el diseño, implementación y posterior promoción de un web. Por ejemplo, en ella se debe especificar:

a. Los objetivos del servidor web

b. La información que se publicara

c. Los recursos necesarios para soportar la operación y desarrollo del servidor web, entre otras especificaciones.

A continuación, haremos un estudio sobre los principios y factores de esta etapa de planificacion tomando en cuenta los aspectos mencionados anteriormente.

4.1.1. Factores considerados en la Planificación de un servidor web

Existen dos clases de factores que deben ser considerados en la planificacion de un web, estos son: los factores controlables por el planificador y los no controlables. Los factores controlables, como indica su nombre, son aquellos sobre los cuales el planificador ejerce algún control. Los no controlables, son los que escapan al control del planificador. Los factores no controlables son:

4.1.1.1. La conducta del usuario

El planificador no puede controlar la forma en la que un usuario accesara al servidor web, debido a que dicho usuario puede entrar al servidor en cualquier pagina del mismo y no necesariamente por el Homepage o pagina inicial. Lo que puede hacer el planificador es establecer un "modelo de acceso de los usuarios planificado" para que tal modelo sirva como guía en el proceso de diseño del servidor web. Los modelos que puede definir el planificador son:

a. Modelo guiado

Este modelo induce al usuario a seguir un camino o secuencia de paginas de hipertexto; por ejemplo, cuando se crea un servidor web de caracter narrativo como son aquellos que cuentan una historia o explican una serie de conceptos.

b. Modelo con múltiples alternativas

En este modelo el usuario puede escoger entre multiples opciones a seguir, pero se espera que dicho usuario pueda elegir entre tales opciones con un minimo de guia. Este modelo es aplicable para aquellos servidores web que contienen informacion que muy fiecuentemente los usuarios accesan (y por lo tanto, se han acostumbrado ya a navegar por el servidor). Tal es el caso de informacion de bases de datos en el servidor web ,entre otros.

c. **Modelo flotante**

Es aquel modelo en el que el usuario debe elegir entre multiples alternativas, pero todas ellas relacionadas solamente con la informacion de la pagina, a diferencia del modelo anterior, en el que los enlaces podían llevar a paginas que hablaran de temas diferentes. Este modelo es útil para confeccionar sitios de webs de entretenimiento o de "juego" en los que se debe explicar la tematica de cada uno de ellos.

4.1.1.2. El browser empleado por el usuario

Es otro factor sobre el cual no se tiene control. Existen browsers textuales, gráficos y que interpretan extensiones propias de HTML. Es por esto que se debe decidir en que formato colocar la informacion relevante del servidor web. Si colocamos tal informacion con imagenes, las personas que posean browsers textuales, no podrán tener alcance a la informacion. Para solucionar esto, se debe colocar el elemento HTML llamado ALT que sirve para proporcionar un texto "alternativo" a la imagen, en caso de que esta no pueda ser visualizada.

Por otro lado, hay que considerar que HTML no es un lenguaje de "publicidad", sino un lenguaje semántico. Esto quiere decir que, escapa del control del desarrollador del servidor web, la forma en que un usuario ve la hoja de hipertexto, pues esto es configurado en su propio browser. Mas bien, HTML indica la estructura del documento y ciertos parámetros de visualización. Algunos otros se configuran o cambian en el browser.

4.1.1.3. Enlaces incorrectos en el servidor web

Cuando un servidor web posee enlaces hacia sitios remotos, que realzan el contenido de una página de hipertexto, se corre el peligro de que con el tiempo, estos

enlaces no sean correctos, es decir, que algún componente del URL ya no exista, causando frustración al usuario final. Esto, en algunas ocasiones se vuelve difícil de controlar.

Pasemos a examinar **los factores sobre los que el planificador del servidor si tiene control:**

4.1.1.4. Nivel de interactividad con el usuario

Sera decision del planificador del servidor web, si se debe o no hacer un servidor web interactivo con formas, páginas de hipertexto creadas Qnamicamente, etc , y determinar donde y cuándo hacerlo, buscando hacer al sistema facil de usar y entretenido.

4.1.1.5. Elaborar "paquetes de informacibn" de manera modular

Es recomendable agrupar todas las ideas afines a un tema en un solo modulo de hojas de hipertexto, de tal manera que sea facil referenciar desde cualquier sitio del servidor web tal información.

4.1.1.6. Frecuencia de actualización de las páginas del servidor web.

La frecuencia de actualización depende del tipo de información que estamos publicando. Sin embargo, es bueno regularmente, por lo menos cambiar la forma de las hojas de hipertexto que generalmente no cambian en contenido.

4.1.1.7. Ubicacidn de los enlaces hacia sitios remotos al servidor web.

Se debe planificar el lugar en el que irán ubicados los enlaces hacia sitios remotos dentro de un servidor web. La **profundidad** del servidor web es una caracteristica del servidor por la cual, los enlaces hacia sitios remotos se encuentran ubicados a una detenninada Qstancia (medida en hojas de hipertexto), del homepage u hoja principal del servidor. Por ejemplo, el servidor web de la ESPOL es de profundidad dos, ya que a partir de las hojas que se encuentran **a** dos o mas hojas de distancia del Homepage, se ha permitido la inclusion de hiperenlaces hacia sitios remotos en la Internet.

4.1.2 Técnicas de planificación de un servidor web

En esta sección se estudian las tecnicas de planificacion que se deben seguir si se desea detenninar los siguientes puntos:

4.1.2.1. Información sobre la audiencia

Un planificador debe establecer cual sera el publico objetivo del web y definir información crítica acerca del mismo. Esto significa que el planificador debe prever para quién desarrollara el web especificamente y cuales son las caracteristicas que deberh poseer los futuros usuarios del servidor web. Para esto, es recomendable escribir en una frase quienes serh el publico objetivo del web, como por ejemplo: *"colegiales que están interesados en ingresar a la Universidad";* pero además se deben establecer las caracteristicas de los "colegiales" a los que sera dirigido el web, por ejemplo:

Colegiales:

Proximos a graduarse

Especializacion Fisico-matematicas

Gran deseo de investrgacion, etc.

Una vez establecidos el publico objetivo y sus caracteristicas, procederemos al siguiente punto: el propósito del web.

4.1.2.2. Propósito del web

La definición del propósito de un web, ayuda en Qferentes circunstancias:

- Ayuda a definir que información recoger y mantener acerca del publico objetivo.
- Influye en la forma en la que se presenta el web.

ESCUELA SUPERIOR POLITECNICA DEL LITORAL
BIBLIOTECA CENTRAL

- Los analistas lo necesitan para evaluar si un web está operando efectivamente.
- Los usuarios finales requieren saber el proposito del web, para responderse a la pregunta: ;Para que fue creado este servidor web?

Este es un ejemplo de como debe establecerse el propósito del web:
"El propósito de este servidor es proveer acceso a un amplio rango de información de y sobre Japón, con el objetivo de crear un entendimiento profundo sobre la sociedad Japonesa, política, industria y lo mas importante, la gente Japonesa" - Tomado del Homepage del Centro para las Comunicaciones Globales (http://www.glocom.ac.jp/index.html)".

En este establecimiento del propósito, se pueden diferenciar cuatro partes importantes:

a. **El area sobre la que se desarrolla el web:** Información sobre Japon.

b. **La audiencia:** Todos los interesados en conocer acerca de este país.

c. El **nivel de detalle de la información presentada:** "la sociedad Japonesa, politica, industria y lo mas importante, la gente Japonesa".

d. **El beneficio esperado por el usuario:** "entendimiento profundo" del Japon.

Estos son cuatro factores que siempre deberán considerarse al momento de establecer el propósito de un web.

El siguiente elemento a considerar en la planificacion es el establecimiento de los objetivos.

4.1.2.3. Establecimiento de los objetivos

Una vez que el desarrollador del web ha planeado el propcisito del web, quien es el público objetivo y cuáles son sus caracteristicas, el siguiente paso es combinar toda esta información para llegar al establecimiento de los objetivos del web. De esta manera, un objetivo es más específico y más detallado que el propósito del web (visto en el numeral anterior). Existen dos diferencias bien marcadas entre el establecimiento de un objetivo y el establecimiento de un propcisito:

a. La primera diferencia radica en el hecho de que a pesar de que el propósito sea siempre el mismo, los objetivos pueden variar, a medida que se vaya recibiendo mas informacion sobre la audiencia y el dominio de la informacion (que sera visto en el siguiente numeral).

b. La segunda diferencia es el hecho de que mientras el propósito de un web nos indica "Esto es lo que vamos a hacer", los objetivos nos indican "Esta es la informacion que lo hará".

Tomaremos una frase del ejemplo del numeral anterior para explicar un poco mas lo que significa el establecimiento de los objetivos del web:

"proveer acceso a un amplio rango de informacion de y sobre Japón". Este proposito puede ser implementado con una variedad de objetivos especificos. Los objetivos pueden ser: "Mostrar informacion cultural, geografica y climatologica sobre Japón" o "Publicar selecciones de publicaciones Japonesas via Internet".

4.1.2.4. Información de dominio

La informacion del dominio se refiere a informacion y conocimiento sobre el área objetivo del web e incluye tanto a la informacion que sera presentada a los usuarios finales, como a la informacion y conocimiento que los desarrolladores del web necesitan tener para hacer una buena labor. A continuación se dan tres pasos a seguir para poder planificar la informacion del dominio de la informacion:

a. El planificador debe definir que informacion es necesaria que conozca el desarrollador del web y que informacion sera dada a los usuarios finales.

Por ejemplo, tomando nuevamente el ejemplo sobre el web acerca de Japón, nos deberíamos preguntar: ¿existen bases de datos a las cuales puedan tener acceso tanto el desarrollador como el usuario?; ¿Cuáles son los conocimientos que deberá poseer el desarrollador acerca de Japón para poder hacer elecciones sobre el contenido de la informacion y su organización?, entre otras.

b. Se debe establecer como se podra conseguir la informacion.

Tomando como ejemplo el web sobre Japón, nos preguntaríamos: ¿Dónde puede el desarrollador obtener informacion sobre Japon? , ¿Existe algún curso que el desarrollador deba tomar antes de tratar de construir un web?, entre otras.

c. Finalmente, se debe planear como dar mantenimiento y actualización a la informacion.

Tomando en consideración el ejemplo del web sobre Japón, podríamos preguntarnos: ¿Cuándo la informacion sobre Japón perdera su utilidad? ¿Cuáles serán los costos del mantenimiento? entre otras.

4.1.2.5. Especificación del web.

La especificacion de un web es el refinamiento de los objetivos en terminos mas especificos. Por ejemplo, mientras que uno de los objetivos de un web es " proveer enlaces hacia los sitios de web mas importantes de Japon", la especificacion correspondiente es " Se proveeran por lo menos 5 enlaces por cada sector en el Japón: comercial, educacional, religioso..." .

De la misma manera que el establecimiento del objetivo puede cambiar mientras se cumpla el mismo propósito del web, asi tambien, la especificacion de un web puede

cambiar mientras se cumpla el mismo objetivo. Por ejemplo: Los URL previstos para el sector comercial de Japon pueden cambiar.

4.2. Etapa de Análisis de un servidor web

El analisis de un servidor web es el proceso de recoger y comparar informacion sobre dicho servidor y su operación para mejorar la calidad general del mismo. Esto ayuda a la toma de decisiones en otras etapas como el diseño y la implementación. Por ejemplo, analizando los datos que se tenga sobre el interes tecnico del publico, se puede definir que informacion debe ser publicada (esto es una ayuda al proceso de Planificacion).

A continuación veremos los principios propios del Web que deben considerarse al momento de realizar el analisis de un servidor web.

4.2.1 Principios considerados en el analisis de un servidor web

El analisis de un servidor web debe evaluar en que medida el servidor web es consistente con los siguientes principios:

4.2.1.1. Público multinacional y multicultural

Debido a que un servidor web debe estar disponible las 24 horas a nivel mundial, el análisis de su contenido y operación debe tomar en cuenta un público multinacional y multicultural. El analista deberá entonces, aconsejar a los planificadores para que tomen en cuenta este aspecto y brinden informacion o reciban con amabilidad a este tipo de público (El Guestbook o libro de invitados es un buen mecanismo de atencion hacia esta clase de visitantes. Otro gran ayuda para un web, es el presentarlo en diferentes idiomas).

4.2.1.2. Validez y correcta operación de los enlaces

El analista debe verificar para todos los enlaces o URLs colocados en el web, los siguientes aspectos:

a. Verificación retórica: se debe chequear si los enlaces cumplen con el propósito de la hoja. Por ejemplo, hablar en una hoja del servidor web sobre un determinado tema y colocar enlaces que nos lleven a otras secciones que no guardan relación con el tema.

b. **Verificación semántica:** se debe chequear si el contenido de la informacion de los enlaces es correcta, relevante y completa. Por ejemplo, evitar mencionar que

un URL dara mas detalles al respecto de un asunto y encontrarnos luego que tal URL no proporciona suficiente informacion.

c. **Verificacidn técnica:** Se debe chequear si los enlaces se encuentran disponibles, en la medida de lo posible. Por ejemplo: Encontrarnos con enlaces inexistentes en el mismo Homepage. Se recomienda utilizar herramientas automatizadas de chequeo de URLs, por ejemplo: Verify Links (http://www.cs.dartmouth.edu/~crow/lvrfy.html).

4.2.1.3. Controlar la porosidad del web

Existe una caracteristica de todo servidor web llamada la porosidad. La porosidad consiste en que los servidores web pueden ser accesados por los usuarios, desde diferentes puntos, y no necesariamente desde la pagina inicial o Homepage. El analista debe estar consciente de esto y examinar si todos los puntos de entrada ofrecen al usuario suficiente información como para no perderse y llegar a conocer todo el servidor web. Se recomienda, como control de la porosidad, el hacer notar en la mayoría de las hojas del web, en que sección del servidor web, nos encontramos. Esto se puede lograr mediante la utilización de botones, barras o hiperenlaces.

4.2.1.4. Estar atento a los avances tecnológicos en el Web

La naturaleza cambiante del Web en lo referente a tecnologia e informacion inducen al analista a estar siempre al dia en estos temas, de tal manera que pueda evaluar si el servidor web esta trabajando de manera efectiva. Por ejemplo, el analista debe estar al dia en conocimientos sobre herramientas de programacion para servidores web (Java, Active X, CGIs), de tal manera de saber cuáles son las herramientas más adecuadas a las necesidades del servidor web.

4.2.2 Factores considerados en el análisis de un servidor web

Al analizar un servidor web, debemos tomar en cuenta los siguientes factores influyentes al momento de presentarlo a los usuarios finales:

4.2.2.1. Rendimiento

Este es un factor sumamente importante que debe ser considerado tanto en la etapa de planificacion (ver sección 4.1.1.3) como en la de análisis de un servidor web. Una de las impresiones mas importantes que el servidor web puede dar a sus usuarios es el considerar cuánto trabajo toma accesar a su informacion. Muchas imagenes dentro del documento o páginas extremadamente grandes, pueden causar tiempos de recuperación largos. El rendimiento para los usuarios varía ampliamente, basado en el browser que

usan, el tipo de conexión a Internet que poseen, y en cuán ocupada esth el servidor web o la red.

4.2.2.2. Legibilidad

Esta es una prueba simple para determinar si el usuario final puede leer el texto de las páginas del servidor web. Con el advenimiento de imagenes de fondo (background), los desarrolladores a menudo crean fondos texturizados y coloreados que hacen la lectura desagradable y algunas veces casi imposible. Otros problemas incluyen el uso de tamaños de letras extremadamente grandes y texto centelleante.

4.2.2.3. Interpretabilidad

La interpretabilidad es tambien un factor preponderante en la planificacion (ver sección 4.1.1.2) y análisis de un servidor web. El analista debe probar el servidor web en varios browsers para estar seguro que la informacion está dispnible a los usuarios en la forma deseada. Si la informacion esencial esth disponible en texto, el analista debe usar browsers basados en texto (por ejemplo el browser Lynx de la U. de Kansas) para estar seguro que dicha informacion (incluyendo el texto en el camp ALT de una imagen), está configurada para guiar a los usuarios sin graficos.

ESPOL
ESCUELA SUPERIOR POLITECNICA DEL LITORAL
BIBLIOTECA CENTRAL

4.2.2.4. Estética

La estetica es una impresion subjetiva del placer visual que produce interactuar con el browser. Por obvias razones, es difícil de probar, pero existen consideraciones que ayudan a mejorar la estetica de un servidor web, como las siguientes:

a. El analista debe preguntarse si el servidor web exhibe un diseño coherente con los objetivos del servidor que ayude al usuario a concentrarse en su contenido, al mismo tiempo que se ocasione una grata impresion y placer visual al visitante.

b. El analista debe sugerir al diseñador que coloque elementos graficos repetidos en muchas páginas de tal manera que ayuden a la consistencia del servidor web.

c. El analista debe observar si se ha hecho un buen uso de los colores de la pantalla, sin sobrecargarla o mezclar indiscriminadamente tales colores, lo cual puede confundir al usuario final.

4.2.2.5. Usabilidad

La usabilidad de un servidor web mide que tanto se adapta el servidor web para la satisfacción de las necesidades de los usuarios finales. Para medir esto, se pueden emplear las siguientes tecnicas:

a. **Hacer** un **simple recorrido del web.** Un analista puede chequear si se está cumpliendo el propósito del web y satisfaciendo a su público objetivo, haciendo un recorrido de todo el servidor, chequeando que exista consistencia entre las páginas, legibilidad, estetica y sobre todo que la informacion se relacione con los objetivos del servidor.

b. **Prever las tareas del usuario final.** Basados en la informacion que se tenga sobre las características del público (sección 5.3.1), se puede prever un conjunto de tareas que serán ejecutadas por el usuario final. Luego, el analista debe chequear si están implementadas estas tareas y anotar cualquier problerna que se presentase.

c. **Probar las tareas en usuarios representativos.** El analista debe observar el uso que hacen del servidor web un conjunto de usuarios representativos, para hacer recomendaciones sobre posibles mejoras al servidor.

4.3. Etapa de diseño de un servidor web

En el proceso de diseño de un servidor web se define, tomando en cuenta las especificaciones derivadas de los principios de planificación y análisis del web, la

manera en la cuál los componentes del servidor web serán construidos. Para esto, se debe conocer con que posibilidades se cuenta al momento de la implernentacion del web (Por ejemplo, considerar todas las posibilidades que se tengan al momento de diseiiar una página con HTML, lo cual implica usar o no usar el estándar, elementos extendidos, etc).

4.3.1. Principios considerados en el diseiio de un servidor web

Todo diseñador de servidores web debe tener en mente los siguientes principios generales al momento de crear un diseño del servidor web:

4.3.1.1. Mantener la competitividad del servidor web

Debido a que el Web es tan competitivo (la competitividad es producto de que en el mundo del WWW pueden existir otros servidores web orientados hacia el mismo público y que tengan un proposito parecido), los desarrolladores de servidores web deben competir por la atencion del publico, haciendo que sus diseños incluyan los costos mas bajos posibles para sus usuarios. Los costos del usuario final incluyen ***tiempo de recuperacidn de la informacidn y el esfuerzo requerido para usar y entender tal informacidn.*** En el diseiio e implernentacion de un web habrá que seleccionar caracteristicas que satisfagan las necesidades de los usuarios buscando un balance de espacio, tiempo de acceso, gráficos, y requerimientos de mantenimiento a largo plazo.

Es decir, el objetivo es crear un web que ***sea eficiente de operar, elegante y fácil de usar y simple de muntener.***

4.3.1.2. Crear una presentación consistente, placentera y eficiente

Cada pagina del web debe indicar al usuario final la identidad y proposito de la misma. La apariencia global del servidor web debe ayudar a los usuarios a cumplir sus objetivos a traves de interfases que sean simples, completas y esteticamente placenteras. Las paginas deben ser consistentes, es decir, que sus componentes se presenten de acuerdo a cómo el usuario espera que se presenten. Por ejemplo, la imagen de una tortuga dentro de una casa en el servidor web de la ESPOL, siempre indicara el regreso al homepage y siempre estará ubicada en la parte inferior de la pagina.

4.3.1.3. Soporte a la Retroalimentación

El usuario final debe tener una manera de contactarse con los desarrolladores del servidor web para poder hacer preguntas o alertar sobre problemas relacionados con el servidor web. Es por esto que es recomendable colocar en la mayoría de las paginas una dirección email de contacto a la cual se pueda recurrir.

4.3.2 Técnicas de diseño de un servidor web

Existen tecnicas de gran utilidad al momento de diseñar un servidor web. Entre ellas tenemos:

4.3.2.1. Empaquetar la información

Esto quiere decir, agrupar una pagina o un conjunto de páginas que están muy relacionadas, en un paquete informacional. Un paquete se conforma de la siguiente manera:

1. Se transcribe una copia de los objetivos del servidor web y se extraen de ellos todos los sustantivos.

2. Se dibuja un circulo por cada sustantivo y luego se los agrupa en terminos del topico al que hacen referencia. Los circulos resultantes serán los denominados "paquetes informacionales".

3. Finalmente, se genera una pagina de web por cada circulo que haya resultado de la agrupacion.

Por ejemplo, analicemos el siguiente servidor web, en el que se especifica el proposito y objetivo del mismo:

web: Computer-Medated Communication (CMC)

Propósito: Ayudar a las personas a compartir recursos

Objetivo: Constituirse en la principal coleccion de material relacionado con CMC, bibliografías, centros academicos y de investigacion relacionados con CMC, periodicos en linea, otros recursos en linea, y una lista de gente y actividades.

Del objetivo establecido aplicamos el primer paso:

1. Extraer los sustantivos del objetivo:

Los sustantivos son: Bibliografías en linea, periodicos en linea, recursos en linea, centro de investigacion, centro academico, lista de gente y lista de actividades.

2. Colocamos cada sustantivo en un circulo y los agrupamos según el topico al que hagan mención:

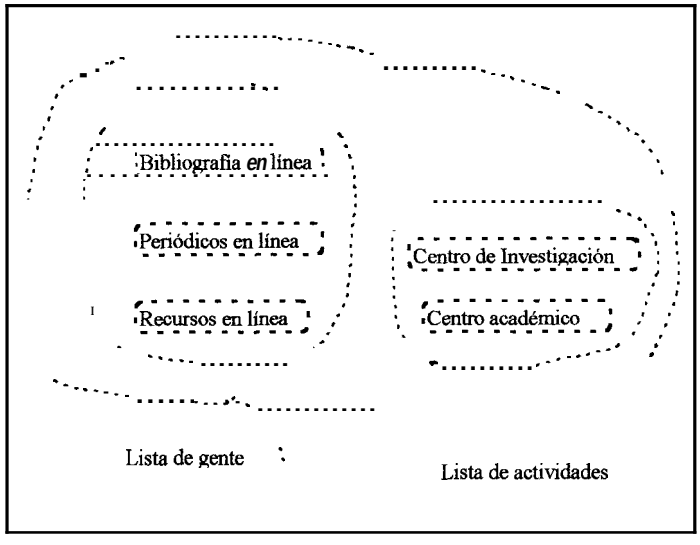


Figura 4.1.
Determinación de los "paquetes informacionales" para los servidores web
Fuente: December, John; HTML & CGI Unleashed, 1ra Edición

En el presente ejemplo, se han dividido los sustantivos por gente, actividades y recursos. Los recursos a su vez, están conformados par dos subgrupos, los cuales son: informacion en linea (bibliografia en linea, periodicos en linea y recursos en linea) y centros de informacion (centro de investigación y centro academico).

La lista de gente conforma el paquete "gente" y la lista de actividades conforma el paquete "actividades".

3. De este Qagrama se obtendrán los siguientes hojas de web:

1. Una hoja de web acerca de los recursos que se pueden obtener en CMC
   1.1 Una hoja de web hablando sobre los recursos en linea
   1.2 Una hoja de web hablando sobre los periodicos en linea
   1.3 Una hoja de web hablando sobre el Centro de Investigaciones y el Centro academico
2. Una hoja del servidor web listando a la gente miembro de CMC
3. Una hoja del servidor web listando las actividades a realizarse.

En total se generarán 8 páginas del servidor web, a raiz de los objetivos mencionados.

4.3.2.2. Unir las páginas

Luego de separar las ideas por paquetes de información se puede escoger varios metodos de union de paginas: haciendo una estructura en forma de árbol o enlazando todas-con-todas las paginas del servidor web (recomendable para servidores web pequeiios), o una combinación de ambos.

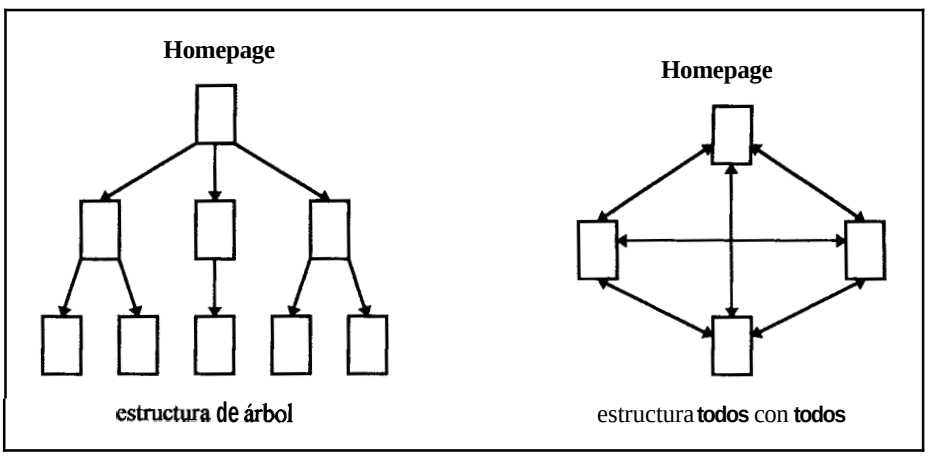


Figura 43.
Diferentes formas de implementación de un servidor web
Fuente: December, John; HTML & CGI Unleashed. Ira ed.

4.3.2.3. Especificar un formato universal de presentación

Se requiere crear una plantilla para dar a todas las paginas una presentación uniforme. Por ejemplo:

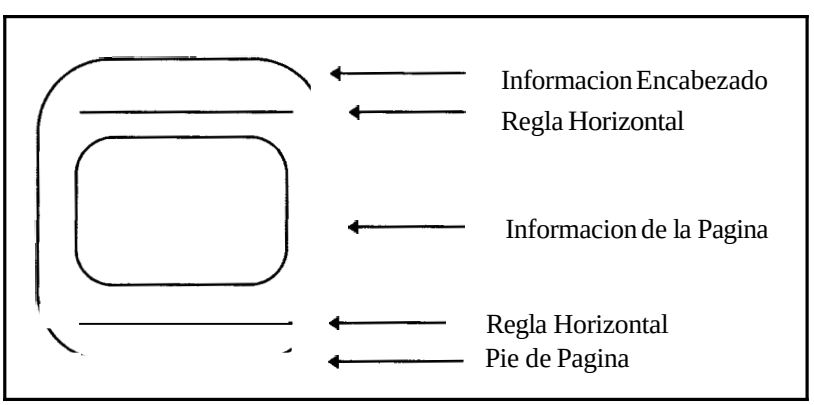


Figura 4.3.
Plantilla modelo para hojas HTML

Esta plantilla podra variar de acuerdo al paquete informacional en el que nos encontremos. Generalmente los paquetes informacionales que agrupan a otros paquetes pueden diferir en la plantilla usada. Un ejemplo de esto es el servidor web de la ESPOL **(http://www.espol.edu.ec)**, en el cual las hojas de hipertexto a un nivel de distancia del homepage, difieren en su formato con respecto a las hojas de niveles mas profundos.

El propósito de la pagina es necesario generalmente en los niveles de profundidad 1 o 2, pero no en todos los casos. La plantilla puede variar de acuerdo a que paquete de información estamos hablando.

4.3.2.4. Crear el índice del servidor web

Crear el índice por paquetes que contenga enlaces hacia los paquetes informacionales mas relevantes. Se recomienda utilizar herramientas automaticas de creación de indices, como la que se encuentra en la ESPOL (http://wwwespol.edu.ec/cgi-bin/leerdirectory.pl).

4.3.2.5. Diseñar las páginas para la mayoría de los browsers

Si debido a requerimientos o limitaciones especiales, se diseña una pagina de web con elementos solamente interpretables por un browser en particular, advertir al usuario que se requiere dicho browser.

4.3.2.6. Indicar la fecha de la última modificación de la página

Esto le permitira conocer con rapidez al usuario final, si ha existido algún cambio en el servidor desde su última visita o que tan actual es la información presentada. Tratar de especificar en letras, el mes del año en mencidn: 2-ABR-96 (y no 2-4-96, que puede tener diversas interpretaciones).

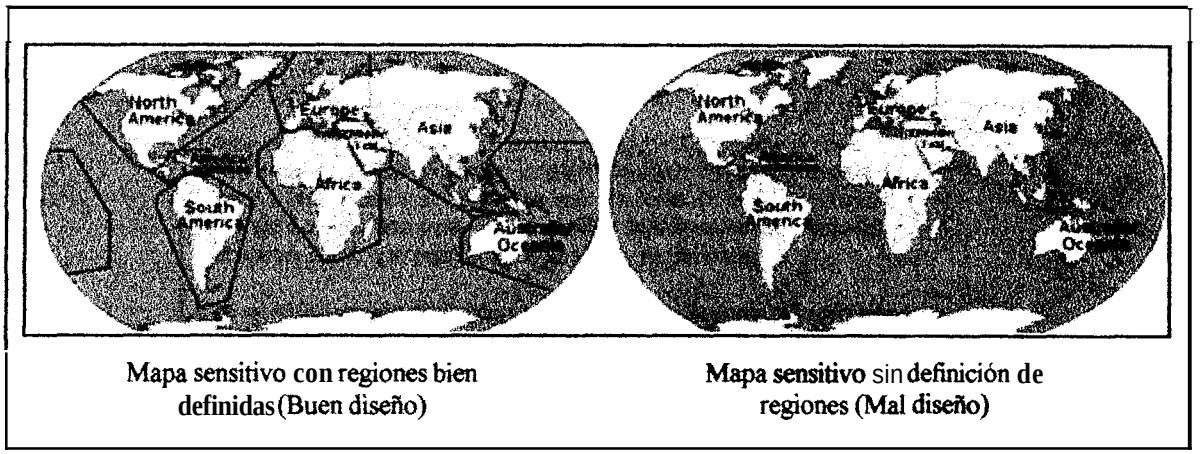
4.3.2.7. Imágenes sensitivas: proveer textos alternativos

Una imagen sensitiva es un grafico que se ha dividido por regiones, en las que cada region nos llevara hacia un enlace de hipertexto. Solamente son visibles en browsers graficos por lo que se deberá proveer un texto alternativo por cada region de la imagen, de tal manera que ambos (el texto y la region de la imagen) puedan llevar hacia un mismo hiperenlace. De esta manera, las personas que usan browsers solo texto o aquellas que tengan apagada la opción "cargar las imagenes" en el browser, o aquellas otras que simplemente no reconozcan hacia donde lleva una imagen sensitiva, podrán tener la alternativa de escoger simples enlaces de hipertexto.

4.3.2.8. Delinear las regiones de un mapa sensitivo.

Cuando se tiene mapas sensitivos con regiones en las que es difícil reconocer los limites, se aconseja delinear tales limites (con líneas o colores), de tal manera que el

usuario pueda tener una ayuda navigacional mas. **A** continuacion se da un ejemplo de una imagen sensitiva mal diseiiada y otra bien diseñada.



Mapa sensitivo con regiones bien definidas (Buen diseño)

Mapa sensitivo sin definición de regiones (Mal diseño)

Figura 4.4.
Comparación de diseños de mapas sensitivos

4.3.3 Problemas de diseño de un servidor web

Existen errores a nivel de diseño, que son bastante comunes de cometer, los cuales mencionamos a continuacion:

4.3.3.1. Páginas sin información ni enlaces

Una de las cosas mas decepcionantes que un usuario del servidor web puede encontrar en este, es una pagina como la siguiente:

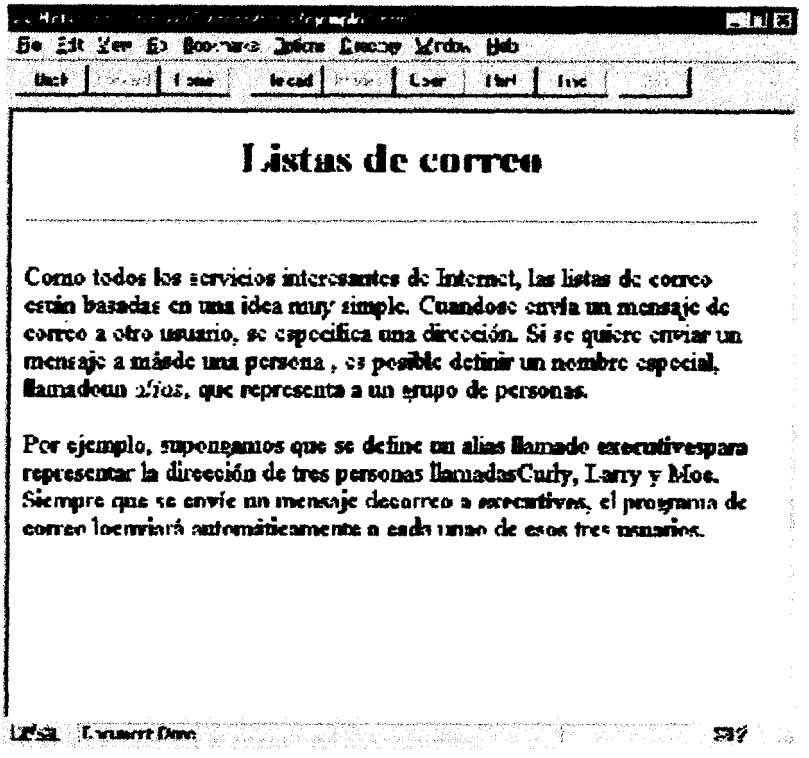


Figura 4.5.
Página sin ayudas navigacionales

El problema de esta pagina es que, a pesar de que está bien escrita (tiene un titulo descriptivo e incluye información que guia al usuario de web a traves de sus puntos principales acerca de las "listas de correo"), cualquier usuario que entre a esta pagina no sabra contestar preguntas como: ¿quién la escribio?, ¿porqué se escribio?, ¿Cuál es **el** nombre del servidor web del que forma parte?. No existe ninguna ayuda navigacional, ni siquiera un elemento <TITLE> o titulo en la barra superior del browser.

Se debe evitar el hacer este tipo de paginas, pues se debe recordar que el web tiene una caracteristica llamada porosidad, la cual consiste en que un **usuario** puede entrar al servidor web desde cualquier punto del mismo. Si el usuario entrase por esta pagina, no sabria como llegar al Homepage ***o*** a la pagina anterior.

4.3.3.2. Página con excesivo uso de multimedia

Este tipo de paginas son aquellas en las que un diseiiador, acostumbrado a trabajar con browsers graficos, coloca una gran cantidad de recursos como imagenes, sonidos, peliculas u otros archivos de multimedia en una misma pagina. Para evitar esto, se debe segmentar la pagina en otras mas pequeiias.

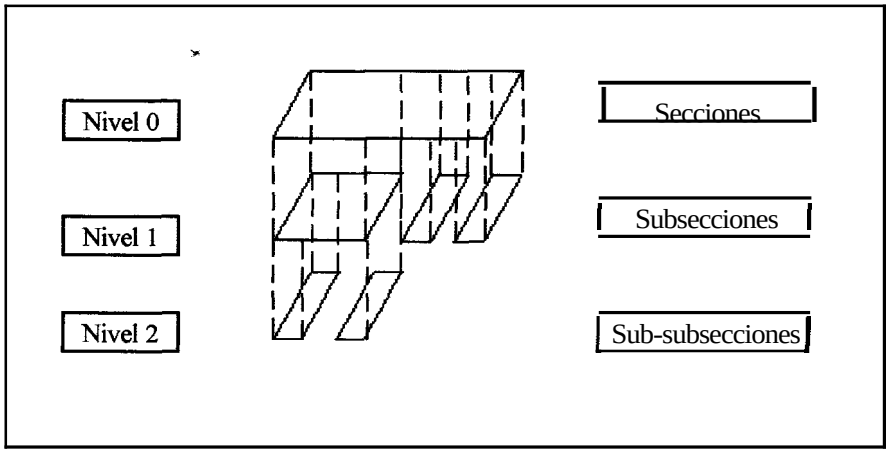


Figura 4.6.
Segmentación de una hoja de hipertexto en otras de menor tamaño
Fuente: December, John; HTML & CGI Unleashed, 1ra ed

Como criterio de separación puede considerarse el hecho de que muchos autores recomiendan que una pagina del servidor web **no** debe exceder los **30** Kbytes.

4.3.3.3. Psiginas con una estructura informacional dispareja

Se refiere al hecho de mezclar de manera indiscriminada, hiperenlaces de un alto nivel dentro del árbol organizacional del web con hiperenlaces de un nivel inferior, de tal manera que para el usuario final , no queda claro cual es el objetivo de la pagina.
Antes de colocar un hiperenlace en la pagina, se recomienda analizar si este está de acuerdo con el proposito de la misma.

4.3.3.4. Enlaces sin sentido

Evitar usar hiperenlaces que hagan referencia a la accion sobre el hiperenlace, por ejemplo:

Para mas infonnacion, presione aqui

Es preferible poner lo siguiente:

Usted Puede obtener **más información**

Esto se debe a que el hiperenlace de por sí invita a la accion de presionar.

4.3.3.5. Psiginas sin organización

Evitar páginas de organización caotica, con enlaces y gráficos por doquier. Para evitar esto, usar tablas. Observar el siguiente ejemplo:

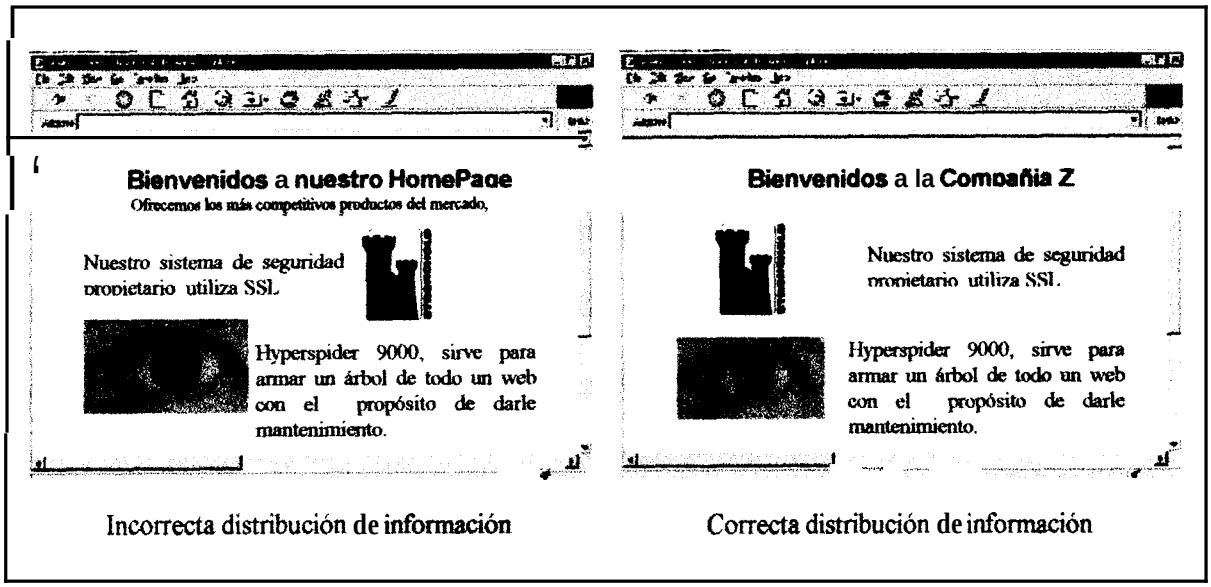


Figura 4.7.
Ejemplos de correcta e in correcta distribución de la información en una página del servidor web

4.3.3.6. Uso exagerado de elementos HTML

No tratar de incluir todos los elementos de HTML en una sola pagina. No abusar del uso de elementos tales como las extensiones del browser Netscape.

4.4. Etapa de implementación de un servidor web

La irnplementacion es el proceso en el cual, realmente se construye el web, usando el Hypertext Markup Language (HTML).

4.4.1. Principios considerados en la implementación de un servidor web

Existen un conjunto de principios considerados en la implernentacion de un servidor de web, que han sido recogidos en base a la experiencia. Estos son:

4.4.1.1. Planear un trabajo continuo en el web.

Se debe considerar que jamás se dejara de actualizar un servidor de web, pues la naturaleza de los mismos es cambiante, por lo que habrá que considerar en el plan de trabajo, el realizar continuas reimplementaciones y rediseños.

4.4.1.2. Separar las tareas del implementador

La tarea del implementador es el confeccionar las hojas de hipertexto del web, en base a las especificaciones dadas por el planificador, las sugerencias hechas por el analista y el esquema especificado por el diseñador. El implementador debera tomar en consideración algunos aspectos al momento de confeccionar el servidor web:

a. **Nivel de compatibilidad HTML**

En base a las especifficaciones y al disefio del servidor web, el implementador debera encontrar los elementos que permitan llevar a cabo la tarea encomendada. Mas adelante se detallara los diferentes niveles de compatibilidad existentes.

b. **Técnicas en la elaboración de imágenes**

Existen un conjunto de tecnicas que permitirán minimizar el tamaño de las imagenes o mejorar el tiempo de acceso a las mismas, como veremos a continuación:

b.1. Utilizar archivos GIFs entrelazados (los GIFs entrelazados son archivos graficos en los que la imagen va mejorando su resolución a medida que se va cargando en el browser) , de tal manera que el usuario pueda descubrir de que se trata la imagen y no tenga que esperar a que haya terminado de cargarse por completo.

b.2. Acostumbrarse a usar los elementos de HTML: HEIGHT (altura) y WIDTH (ancho) de la imagen, que le indican al browser de antemano , el tamafio del grafico, lo cual ahorra tiempo de acceso al servidor de web.

b.3. No utilizar archivos JPEGs para colocar imagenes con información importante. Esto se debe a que no todos los browsers soportan JPEGs. Es el mismo caso de las extensiones de browsers como Netscape o Microsoft Explorer. Si se utilizan, el efecto final que se le quiso dar a una presentación, se verá disminuido.

b.4. A veces es mejor utilizar imagenes grandes: En ciertos casos es preferible utilizar imagenes grandes a varias pequefias, sobretodo si la conexión hacia Internet es lenta. Esto se debe a que, cuando se tiene un ancho de banda limitado, es mas rapido bajarse una sola imagen grande

que varias pequeñas, ya que el servidor de web, por cada imagen bajada, abre una conexión con el browser, lo que desacelera el acceso a la pagina.

4.4.1.3. Utilizar plantillas

El implementador debe crear, en base a las especificaciones de diseño, un conjunto de plantillas en las cuales basar la confección de las hojas de hipertexto. Este paso le ahorrará mucho tiempo y esfuerzo al momento de la implementación. Para conocer cuál debe ser el bosquejo de una plantilla, consultar la sección 4.3.2.3 del presente trabajo.

4.4.2 Nivel de compatibilidad HTML

Es menester conocer los diferentes niveles de HTML existentes (estandar de HTML), de tal manera que se pueda escojer el mas adecuado de acuerdo a los recursos con que cuente nuestro público objetivo. Estos son:

4.4.2.1. Nivel 0

Descripción:

Este es el "nivel mas bajo" de HTML, es decir aquel nivel que todos los browsers pueden soportar (inclusive el email browser agora@mail.w3.org -browser que trabaja

con comandos via correo electronico- y otros browsers de modo texto). Este nivel puede ser usado en cornputadores tales como los PDA (Personnal Digital Assistant, tales como el Newton de Macintosh). Para ver en detalle todos los elementos de este nivel, referirse al apéndice B (Lenguaje HTML)

Desventajas:

Este nivel no tiene capacidades expresivas, carece de graficos y no soporta la mayoria de los elementos HTML.

4.4.2.2. Nivel 1:

Descripcidn:

Este nivel adiciona imágenes embebidas y muchos elementos HTML. Mantiene los mismos elementos HTML del nivel 0 pero se diferencia de este en que puede mostrar imagenes y soporta un grupo adicional de elementos HTML (Referirse al apendice B para una lista completa de los elementos soportados por el nivel 1).

Desventajas:

A pesar de que el nivel 1 adiciona alguna funcionalidad sobre el nivel 0, la mayoria de los browsers modernos soportan el nivel HTML 2.0. Esto se debe a que HTML 1.0 no representa ningún cambio significativo sobre HTML de nivel 0, mientras que el HTML

2.0 como veremos a continuación, introdujo un elemento muy importante en el mundo de los servidores web: Las FORMAS.

4.4.2.3. Nivel 2

Descripcidn:

Este nivel adiciona FORMAS, que son hojas HTML que permiten recoger informacion proveniente de un usuario. Las FORMAS permiten enviar informacion desde el browser a los programas CGIs. (Ver apéndice B para mayor informacion sobre los elementos HTML de nivel 2.0)

Desventajas:

Los browsers que soportan nivel 0 ***o*** **1** no reconocen las FORMAS. Se debe tomar esto en cuenta sobretodo si nuestro público objetivo posee browsers que solo soportan tales niveles.

4.4.2.4. Nivel 3

Descripción:

Este nivel de HTML esta todavia en proceso de estandarizacion. Se espera que este nivel adicione tablas, ecuaciones matematicas, pie de páginas, frames y otras caracteristicas.

Desventajas:

La especificación HTML 3.0 ya es en parte soportada por browsers como Netscape Navigator o Microsoft Explorer. Sin embargo, en la actualidad existen un gran número de browsers que no soportan esta especificacion, por lo que se debe esperar a que se haya estandarizado para poder hacer uso de ella.

4.4.2.5. Extensiones

Las extensiones mas populares de la actualidad son las del browser de Netscape Communications, llamado tambien Mozzila

Descripción:

Añade caracteristicas como centelleo de letras, fondos texturizados, cambios de tipos de letras, alineacion de graficas, listado de elementos y centrado. Usadas creativamente, estas caracteristicas añaden interes al web. (Ver las extensiones en el apéndice B)

Desventajas:

Muchas de estas caracteristicas son visibles solo con el browser Netscape Navigator.

4.5. Etapa de promocion de un servidor web

La promocion de un web es el proceso de manejar las relaciones públicas del servidor web, como por ejemplo, el anunciar la existencia del servidor web en grupos de discusion , o sitios de gran concurrencia en la Internet.

4.5.1. Principios de promoción de un servidor web

La promocion de un web es una tarea laboriosa y requiere conocimiento de los principales foros de promocion a nivel mundial. Para promocionar un web se deben tomar en cuenta, los principios que se detallan a continuación.

4.5.1.1. Usar apropiadamente los Foros o Grupos de discusión

Una de las reglas básicas para participar en foros (tales como los newsgroups, que reunen a un grupo de personas alrededor del mundo que hablan sobre un tema en especifico), es buscar el foro apropiado al topico del que queremos hablar o hacer mención. Esto es importante de entender al momento de promover un servidor web en la Internet, puesto que la promocion no se puede hacer indiscriminadamente, enviando informacion sobre el servidor web a todos los foros imaginables. Esta acción causaria un efecto contrapuesto a lo que se quiere lograr con la promocion del servidor web, ocasionando molestias a los miembros del newsgroup enviandoles informacion que no

guarde ninguna relación con el proposito del mismo. Por ejemplo, si el propósito de nuestro servidor web es "El hablar sobre temas religiosos y difundir la fe catolica en el mundo", sería contraproducente enviar un mensaje de promoción a grupos tales como: alt.atheism (newsgroup referente a ateismo), alt.astrology (newsgroup referente a temas sobre la Astrologia), que no tienen ningún punto en común con el propósito de nuestro servidor. Se recomienda que, antes de enviar un mensaje a un foro de discusion, se lea el FAQ del mismo (FAQ son las siglas de Frequently Asqued Questions o preguntas mas comunes, que generalmente un newsgroup posee y que recoge todas aquellas preguntas que han surgido en el foro, a lo largo del tiempo).

4.5.1.2. Dar , no sólo recibir.

Una de las caracteristicas de la Internet que la hace atractiva, se centra en el hecho de que existen muchas personas u organizaciones que comparten información o software que creen que servira para una tarea util a los usuarios de la Internet. Esto redunda en beneficio de los autores de aquel "regalo" puesto que les da reconocimiento público y atrae en gran medida, a muchos visitantes a su servidor web, que al fin y al cabo, es el objetivo de todo promotor de servidores web.

4.5.1.3. Conocer y aplicar los "protocolos especializados"

La comunicacion especializada y los foros de interacción tienen sus propias normas de conducta. Un promotor de web debe conocer tales normas para ser aceptado en determinado grupo. Existen fuentes de información al respecto en los siguientes sitios:

"The Internet Advertising Resource Guide"

(http://www.voyager.net/adv/internet-advertising-guide.html)

"Net Etiquette Guide" por Arlene H. Rinaldi

(ftp://ftp.lib.berkeley.edu/pub/net.training/FAU)

"Blacklist of Internet Advertisers", por Axel Boldt

(http://math-www.uni-paderborn.de/~axel/BL/blacklist.html)

4.5.2. Técnicas de promocion de un servidor web

Existen diferentes elementos en el mundo del WWW que *se* deben tomar en cuenta para hacer una promoción efectiva del servidor web. Entre los cuales tenemos:

4.5.2.1. Publicitar el servidor web en el momento justo.

El promotor del servidor web debe trabajar en estrecha colaboracion con el implementador y el planificador del web, para decidir en que momento el servidor web

podra hacerse publico. Uno de los momentos mas intensos del servidor web (es decir, el momento con mayor nutnero de visitas) sera cuando se haya anunciado la disponibilidad del inismo a la comunidad de Internet, por lo que es importante no anunciar al servidor web hasta que esté listo para causar una buena primera impresion.

4.5.2.2. Promocionar el servidor web al publico objetivo

El promotor del servidor web deberá publicitar dicho servidor al publico objetivo especificado en el proceso de planificacion (ver sección 4.1.2.1). Lo podra hacer enviando un mensaje que detalle las caracteristicas que más le interesaran del servidor web al publico objetivo (esto lo puede deducir junto con el planificador del servidor web). Se podra enviar el mensaje a:

Todos los usuarios de la red de la institución a la que pertenece el servidor web.

Newsgroups o listas de discusion que tengan el mismo proposito del servidor web.

Organizaciones profesionales y/o sociedades.

lndividuos u organizaciones que mantienen sus propios indices de servidores web.

añadido una nueva seccion al servidor web. En estos casos se debe especificar con detalles, las caracteristicas de la nueva seccion y no sera necesario publicitar tal cambio en los newsgroups *o* en la Internet.

4.6. Etapa de innovacion de un servidor web

La innovacion es el proceso que asegura que los otros procesos de desarrollo del servidor web (planificacion, analisis, diseño, implementación y promoción), continuan y mejoran cada vez mas la calidad del web. Este proceso también incluye el investigar nuevas tecnologias en el WWW que sirvan para el desarrollo futuro del servidor web

4.6.1 Técnicas de innovacion de un servidor web

A continuación se detallan algunas tecnicas importantes en el proceso de innovacion:

4.6.1.1. Mantener todos los procesos de desarrollo siempre activos.

Una de las tareas de los innovadores del web es el estar en permanente contacto con los miembros de los otros procesos involucrados en el desarrollo del servidor web: el planificador, el analista, el diseñador, el implementador y el promotor del servidor, de tal manera de informarles sobre nuevas tecnologías nacientes en el Web, lugares de

interes en la Internet, etc. Se debe tomar el plan del servidor web (producto del proceso de planificación) y reevaluarlo, para adaptarlo a las nuevas condiciones existentes en el mundo del WWW.

4.6.1.2. Monitorear el ambiente informacional del usuario

Monitorear el ambiente informacional del usuario significa que el innovador del servidor web, debe investigar que tipo de información le interesa al publico. Para esto, es útil conocer , por ejemplo, a que sociedad de profesionales pertenece el usuario, a qué convenciones asiste, los periodicos que acostumbra leer, entre otros. Esto puede hacerse con una encuesta a un grupo de usuarios representativos (que puede ser realizada a traves del mismo servidor web).

4.6.1.3. Mejorar continuamente la calidad del servidor web

Los innovadores del web deben buscar exceder las espectativas y necesidades del usuario final , mejorando continuamente el valor del web, su precision, actualidad, competitividad e interes.

La calidad se la puede definir como el proceso continuo de planificacion, analisis, diseño, implementación, promoción e innovación para asegurar que la infonnacion

concuerde con las necesidades del usuario final en tenninos de contenido, interfase, rendimiento y funcionalidad.

4.6.1.4. Monitorear los requerimientos y necesidades del usuario

Otra de las tareas del innovador es estar en permanente contacto con los usuarios. Existen dos tipos de usuarios: los usuarios proveedores de información y los usuarios consumidores de la misma. Los usuarios proveedores son aquellos que proporcionan información a nuestro servidor, mientras que los usuarios consumidores (usuarios finales) son los visitantes del servidor. El innovador debe estar en permanente contacto con el usuario proveedor de informacion para proporcionarle datos útiles que le puedan servir para mejorar el contenido del servidor web. Los usuarios consumidores de la información podrán contactarse con el innovador a traves de una dirección electrónica (que puede colocarse en el servidor web). Con esto, el innovador recibira retroalimentacion acerca de las posibles mejoras o falencias que tenga el servidor web podrá tomar las medidas del caso.

CAPITULO V

PLANIFICACIÓN, ANÁLISIS Y DISENO DEL SERVIDOR WEB DE LA ESPOL

En base a los principios vistos en la sección anterior, se han fundamentado las etapas de planificacion, análisis y diseño del servidor web de la ESPOL. Antes de empezar a hablar en detalle de cada una de estas etapas, tenemos que referirnos a los condicionamientos o limitaciones en cuanto a Hardware y Software existentes en la ESPOL, además de los multiples requerimientos por parte de los usuarios finales. De esta manera podremos entender el porqué de la adopción de varias politicas en cada etapa de desarrollo del servidor de web.

5.1. Requerimientos del servidor we6 de la ESPOL

Un servidor web, como se ha visto, no puede ser desarrollado, si no se centra en la satisfacción de la mayor cantidad posible de usuarios potenciales del mismo, en **lo** que a información, orden y estética se refieren. El web de la ESPOL no estuvo excento de tal requerimiento, por **lo** que a continuación inencionamos, algunas premisas que fueron el pilar para la confección del servidor web de la ESPOL.

a. El web de la ESPOL debe representar a la Universidad en su conjunto, aglutinando a los diferentes sectores *o* grupos humanos dentro de la misma, como son: personal docente, alumnado, personal administrativo e investigadores.

b. El web de la ESPOL debe ser un web esteticamente atractivo, de gran utilidad para la comunidad politécnica, y eficiente en términos de acceso a la informacion. Debe basar su diseño en un formato preestablecido para guardar consistencia (plantillas).

c. El web de la ESPOL debe constituirse en un medio por el cual, los miembros de la ESPOL puedan familiarizarse rapidamente con el mundo de Internet, agrupando la mayoría de las aplicaciones que antaño se encontraban dispersas

como son: FTP, TELNET, GOPHER, WWW, MAIL, y que ahora, gracias a los browsers, pueden ser vistas a traves de una interfase comun.

d. El web de la ESPOL debe brindar facilidades para la busqueda de información textual, o información sobre usuarios de la ESPOL, de tal manera que para cualquier visitante sea fácil su ubicación dentro del servidor de web.

e. El web de la ESPOL debe ser amigable con los visitantes, permitiendole dejar sus impresiones o comentarios a traves de un "Libro de Invitados" o " Guestbook"

f. El web de la ESPOL debe ser administrable y actualizable de una manera eficiente, de tal manera que no interese el cambio de la persona encargada de administrarlo y que en lo posible, exista una interface unica para su administración

g. Siempre deberá tenerse información sobre el impacto que produce el incluir una nueva sección en el web, a traves de programas que realicen una estadística de acceso al servidor Estos programas tambien deberan recoger estadísticas de las páginas mas vistadas, etc.

BIBLIOTECA
CENTRAL

h. El web de la ESPOL debera tener enlaces frescos siempre, es decir, que enlaces hacia otros sitios que no esten dentro del servidor, sean siempre validados para comprobar que el documento destino sea accesible.

i. El servidor web de la ESPOL debera ser un servidor de profundidad dos, es decir, un servidor web en el que un usuario final, unicamente podra salir hacia otro servidor en la Internet desde una pagina dentro del servidor web que se encuentre a una profundidad o lejania de dos páginas desde el Homepage (ver sección 4.1.1.7)

j. El servidor de la ESPOL debera ser un servidor seguro, es decir, que aquellas transacciones en las que se maneje información confidencial a nivel de CGI, deberan ser encriptadas o protegidas.

5.2. Limitaciones def servidor web de la ESPOL

Para implementar el servidor, se contó con los siguientes recursos:

a. Computador Sun SPARC Solaris **2.4** 32Mb RAM

b. Conexión a Internet limitada y compartida: 64Kbps

Por otro lado, tambien se encontraron las siguientes limitantes:

a. El ambiente de trabajo de la mayoría de los usuarios de Internet de la ESPOL es el Sistema Operativo Unix, por lo que el browser mas utilizado es Lynx (especifico para este tipo de Sistemas).

b. Falta de una base de datos con la información del alumnado y profesorado, que tenga conexión con TCP/IP. De esta manera, el desarrollo de un Sistema Informacional con acceso a la base de Datos de la ESPOL hubiese sido factible.

A continuación especificaremos los pasos a seguir en la planificacion del web de la ESPOL, definiendo el alcance de cada uno de los siguientes factores: el publico objetivo, el proposito del web, los objetivos, la informacion sobre el dominio y las especificaciones del web.

5.3. Planificación del servidor web de la ESPOL

5.3.1. Información sobre el público objetivo

Se definio como publico objetivo a los "Miembros de la ESPOL, aspirantes a miembros y visitantes, interesados en conocer nuestra Universidad y el mundo del

WWW", mientras **que** se detinieron algunas caracteristicas importantes de los mismos, las cuales anotamos a continuación:

a. Como miembros de la ESPOL nos referimos a los profesores, administrativos y alumnos que actualmente pertenecen a la ESPOL, sin distingo de carrera universitaria ***o*** nivel academico.

b. Como aspirantes nos referimos a todos aquellos que deseen integrarse al mundo de la Politecnica en calidad de alumnos.

c.Como visitantes nos referimos a todos los ex-miembros y no miembros de la ESPOL que deseen mantener vínculos con la Institución.

d. Todos ellos deben compartir el deseo de conocer y utilizar todas las herramientas que se brindan en el servidor web de la ESPOL, además de aportar con sus ideas al mismo.

A todos ellos, de ahora en adelante los llamaremos de manera genérica " La coinunidad de Internet"

5.3.2. Proposito del web

El proposito del servidor web de la ESPOL se definió como sigue:

"Proveer información general sobre la Universidad y sus recursos a la comunidad de Internet, con el objeto de fomentar el conocimiento y uso del Web". De este propósito se pueden discernir las siguientes partes:

a. El area a tratar: La Universidad y sus recursos

b. El público: La cotnunidad de Internet

c. Nivel de detalle de la información: Información general sobre la Universidad y sus recursos

d. El beneficio del usuario: fomentar el conocimiento y uso del Web

5.3.3. Establecimiento de los objetivos

Para lograr la consecucion de tal proposito, se establecieron los siguientes objetivos:

1. Desarrollar un sistema de información que de a conocer la Universidad y nuestro País, a la comunidad de Internet, a través de la implementación de páginas de fácil consulta y acceso, con métodos de busqueda y mapas sensitivos que faciliten el uso del sistema **por** parte del usuario final.

2. Desarrollar un medio por el cual los miembros y no mieinbros de la ESPOL puedan integrarse al mundo del World Wide Web a traves de herramientas como: La generación de Homepages, el libro de invitados o Guestbook, la estructuración del directorio de personas y la recopilación de avisos de interes para el alumnado, todo esto de manera automática.

3. Explorar el servidor web como interfase con el usuario final, integradora de los diferentes servicios de Internet, a traves de la adicihn al servidor web del correo electronico, transferencia de archivos o FTP y Gopher.

4. Estudiar las diferentes herramientas de los sistemas de información basados en el World Wide Web tales como las máquinas de busqueda y los servidores Proxy y escoger e implementar las que presenten el mejor rendimiento y la maxima eficiencia, adaptandolas a la realidad de Hardware y software de la ESPOL.

5. Desarrollar herramientas de administración del Sistema de Información que permitan controlar y inonitorear diferentes parametros del mismo, tales como, el chequeo de hiperenlaces y las estadisticas de acceso al servidor.

Luego de haber definido los objetivos del servidor, pasamos a la siguiente etapa: Información sobre el dominio.

5.3.4. Inforrnacion sobre el dominio

De acuerdo al proposito y objetivos planteados, podemos definir que un desarrollador y el usuario final deberan tener información acerca de:

a. La Universidad

Sobre sus Facultades, Tecnologías, Maestrias, Institutos, proceso de admisiones, sobre el Campus Prosperina y Campus las Peñas, en fin, con todo lo que tenga relación a la ESPOL.

b. Servicios

Los servicios actuales y futuros de la Politécnica, incluyendo servicios internos, externos y servicios de Internet.

c. Sobre las actividades universitarias

Inforrnacion actualizada sobre el informativo, calendarios, cursos, entre otros

d. Sobre la Comunidad

Información actualizada sobre Ecuador y nuestra ciudad Guayaquil.

La información a su **vez,** sera obtenida de las siguientes fuentes, por parte del desarrollador:

a. Centro de Difusion y Publicaciones de la ESPOL

Quienes publican el folleto titulado "Catálogo General de la ESPOL" cada 3 años. Además publican el "Informativo Politecnico".

b. Departamento de Planificacion:

Quienes poseen todos los mapas y situación actual y futura de la geografía de la ESPOL.

c. Secretarias de las respectivas unidades academicas:

Las cuales nos proveeran informacion actualizada sobre Calendario de Actividades, Cursos, Seminarios, etc

d. Libros varios, la Internet: De los cuales podremos obtener informacion sobre nuestro país, nuestra ciudad, entre otros temas.

Toda esta información debe ser mantenida y actualizada. Para esto se deberá tener una *o* varias personas asignadas de tal manera que realicen las actualizaciones respectivas. Los costos de actualización seran los siguientes:

a. Costo por hora de la persona que mantiene el servidor web.

b. Costos por utilización de maquina (computadores) y equipos periféricos (scanners). Se incluyen estos costos ya que, a pesar de que las herramientas fuesen propias, existen costos relacionados con la luz que consumen. Además, si no fuesen propios, se deben incluir los costos de alquiler de tales equipos.

c. Costo de entrenamiento del personal: En caso que la persona contratada para realizar el trabajo, no tuviese los conocimientos necesarios para el mismo.

5.3.5. Especificación del servidor web

La siguiente es la especificacion en la que se basaron los trabajos de los diseñadores e implementadores del web de la ESPOL. Este es el resultado de un estudio en conjunto hecho por el autor de la presente Tesis y los miembros del departamento de Redes del Centro de Servicios Computacionales de la ESPOL (Cesercomp). Se basa en

un estudio realizado aproximadamente a 20 universidades en el WWW; se obtuvo un factor comun de todas ellas y se lo adaptó a la información que sobre la ESPOL se tenia hasta el inomento. Estos son los resultados del estudio, en referencia a la estructuracion del servidor web de la ESPOL:

1. Se deberá dividir al servidor web de la ESPOL en 5 secciones: La Universidad y Admisiones, Servicios, Vida Universitaria, Directorios y Busquedas, Comunidad.

2. La seccion Universidad debe constar de las siguientes partes:
Información General, Campus, Carreras, Admisiones y Proyectos

3. La seccion Servicios debe constar de las siguientes partes:
Servicios Internos, Servicios Externos y Servicios de Internet.

4. La seccion Vida Universitaria debe constar de las siguientes partes: Infonnativo, Calendario Academico, Eventos culturales y deportivos, Cursos y Seminarios.

5. La seccion Busquedas debe constar de las siguientes partes:

Directorio Telefónico, HomePages en la ESPOL, Busqueda de Personas y Busqueda en el Web de la ESPOL

6. La seccion Comunidad debe constar de las siguientes partes:

Nuestro Ecuador y Nuestra Ciudad

7. Existira una seccion mas, que sera considerada de utilidad adininistrativa y que estara conformada por: Libro de Invitados, Chequeo de Hiperenlaces y las Estadisticas de acceso.

Estas son las secciones y subsecciones más importantes. Conforman el nivel 0 y 1 del web de la ESPOL. Todas las hojas deben brindar facilidades para usuarios con o sin browsers gráficos (esto significa que se debe incluir un enlace tipo texto por cada enlace gráfico en todos los niveles del servidor web de la ESPOL).

5.4. *Factores controlables y no controlables*

Existen factores controlables tales como:

5.4.1. Mensajes de error

Los mensajes de error enviados por el servidor web (cuando no se encuentre un hiperenlace dentro del servidor o la ejecucion de un programa CGI de una salida erronea) que pueden ser reconfigurados y manejados por parte del Administrador.

Un mensaje enviado por el servidor web de la ESPOL tendra la siguiente forma:

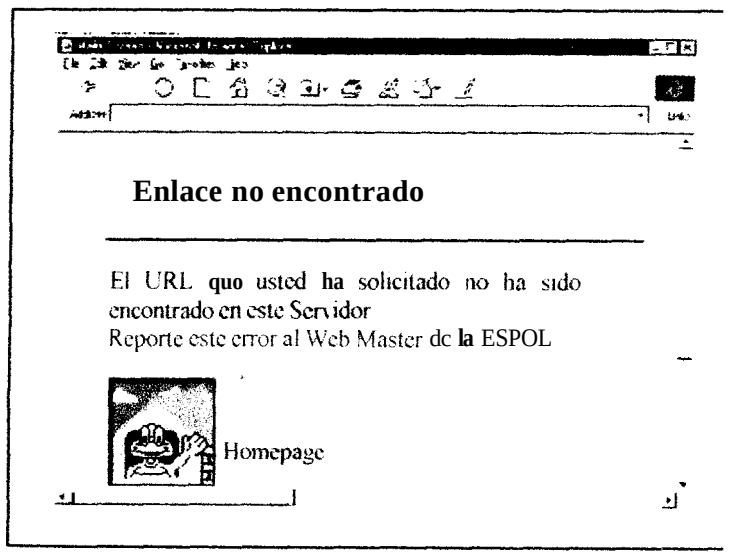


Figura 5.1.
Mensaje de error enviado por el servidor web de la ESPOL
Fuente: Servidor web de la ESPOL

Si no estuviesen configurados los mensajes de error, se obtendria el siguiente resultado:

Error 404 : Not Found, Lo cual, no es ninguna ayuda para el usuario final.

5.4.2. La respuesta de los CGIs

Se deberan considerar todos los casos posibles y resultados de error que un CGI pueda dar, de tal manera que sea practicamente imposible hacer caer el programa servidor web o el Sistema Operativo sobre el cual corre.

5.4.3. Seguridad

Elaborar CGIs seguros de tal manera que cualquier intruso no sea capaz de violar la seguridad del servidor de web y causar daños al sistema. Se debe tomar este cuidado en el caso de los CGIs ya que cualquier intruso puede enviar comandos del S.O. embebidos (incluidos en los datos de una FORMA HTML como se explica en el ejemplo de la figura 5.2.), que se ejecuten en paralelo con el CGI con los consiguientes daños al sistema. El método mas común consiste en la utilización de los simbolos pipe (|) o punto y coma (;) para ejecutar en paralelo cualquier proceso en los sistemas operativos Unix.

Los comandos que se puedan ejecutar a traves de los CGIs dependeran de los permisos de usuario bajo los cuales este corriendo el programa servidor web. Supongamos que el programa servidor web esta comendo con los permisos del administrador del Sistema Operativo. Entonces, el intruso podra ejecutar cualquier comando, incluso:

/bin/rm -fr /* (comando que borra el contenido de todo el disco duro del Sistema)

Es por esto que el servidor web de la ESPOL corre bajo el usuario httpd, que tiene permisos restringidos.

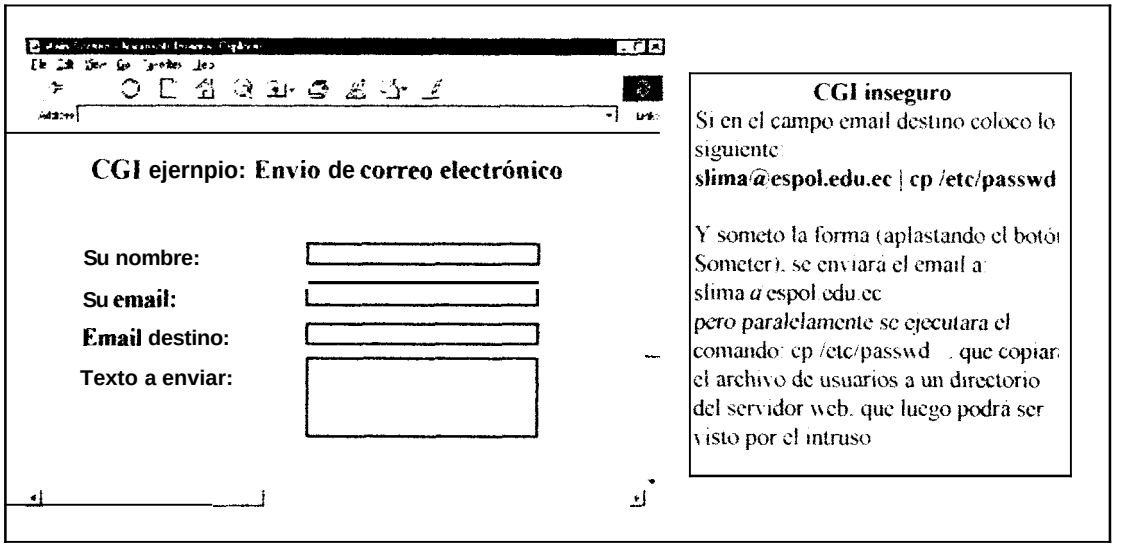


Figura 5.2.
Ejemplo dc un CGI inseguro en un servidor web

5.4.4. Promoción

Se podrán seleccionar las maquinas de registro de servidores web idóneas, de tal manera que el servidor de la ESPOL sea conocido en la comunidad de Internet y en los otros medios mencionados anteriormente

5.4.5. Nivel de compatibilidad HTML

Se escogera el nivel de compatibilidad mas adecuado, de tal manera que el acceso al servidor de la ESPOL tenga un caracter universal y no restringido solo a ciertos browsers.

5.4.6. Programación multiusuario

Se debe considerar que la elaboración de CGIs deberá tomar en cuenta un ambiente multiusuario. Esto es de suma importancia al momento de hacer programas con accesos a archivos e informacion que puede ser accesada o modificada simultáneamente.

Y los **factores no controlables** son:

5.4.7. Presentación de la información

La forma en que sea vista la informacion dependera de las selecciones que el usuario final haga en su browser. Hay que recordar que HTML no es lenguaje de presentación, sino un lenguaje SEMANTICO, es decir, trata de dar sentido a los elementos del documento, mas no trata de darles forma. Sin embargo, se hicieron comprobaciones con los browsers más populares (como Netscape, Microsoft Explorer y Lynx), y se observo que la informacion es legible en todos ellos.

5.4.8. Porosidad

El administrador del servidor no conocera jamás por donde ingresará el usuario final a la información. No necesariamente ingresará a partir del HOMEPAGE. El usuario

final podra grabar el URL que mas le agade, en su browser, para luego entrar a partir del mismo. Es por esto que se coloco en cada hoja, señaladores o titulos **que** le dan al usuario un sentido de orientación dentro de todo el servidor web de la ESPOL.

5.5. Análisis dei servidor web de la ESPOL

5.5.1. Rendimiento

Al analizar el rendimiento de un servidor web se toma en consideracion cuanto le toma al usuario final obtener la informacion del servidor. Y en esto influye el tamaño de la página y las imágenes que contenga. En el servidor web de la ESPOL se ha tendido a desarrollar la mayoría de las páginas con un tamaño no superior a 35kbytes contando texto e imágenes. **A** pesar de la optimización hecha, la velocidad con la que el usuario verá la inforrnacion dependera también de otros factores como: el tipo de conexión a Internet, **el** browser que usa, entre otros.

5.5.2. Legibilidad

En el servidor web de la ESPOL, se ha tenido en consideracion colocar un fondo lo suficientemente claro para no confundirse con el texto y hacer imposible la lectura.

Además se ha considerado el tamaño de las letras, diferenciandolas en titulos, subtitulos, etc. No se ha tratado de hacer un uso exagerado de HTML, sino que todas las hojas se han implementado con los elementos necesarios para transmitir la información al usuario.

La siguiente es una hoja de Hipertexto perteneciente al servidor web de la ESPOL.

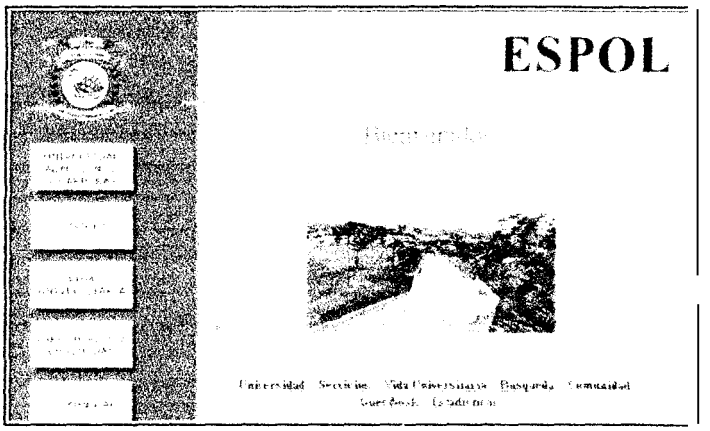


Figura 5.3.
Homepage de la ESPOL usando un browser gráfico
Fuente: Web de la ESPOL http: www.espol.edu.ec

En ella se nota un marcado uso de colores obscuros (azul) y claros (amarillo y blanco), los cuales al combinarse no confunden al usuario. Mas bien, los colores claros resaltan sobre los obscuros. Por otro lado , el tipo de letras usada para los hiperenlaces es del mismo tamaiio.

5.5.3. Interpretabilidad

La información en el servidor web de la ESPOL puede ser accesada tanto en browsers gráficos (tales como Netscape *o* Microsoft Explorer) como en gráficos textuales (como el browser Lynx). Esto se debe gracias a que en cada pagina se tomó la siguiente medida:

Por cada imagen **que** llevase hacia algún URL, existe su equivalente en texto que lleva hacia el inismo URL. Esto explica la "redundancia de información " que existe en el servidor web de la ESPOL. Son ayudas navigacionales que sirven tanto para los browsers textuales corno para los graficos.

La figura que se muestra a continuación presenta como ser veria la infonnacion del Homepage de la ESPOL en un browser textual

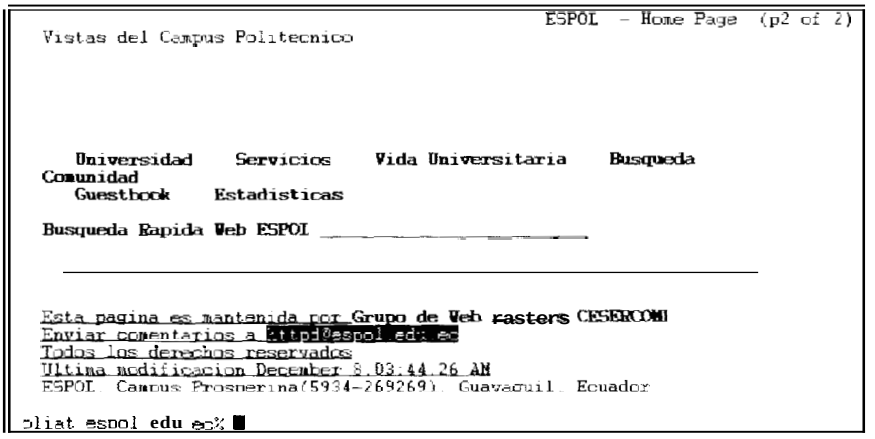


Figura 5.4.
Homepage de la ESPOL usando **un** browser textual Lynx

POLITÉCNICA DEL LITORAL
BIBLIOTECA
CENTRAL

5.5.4. Estética y Funcionalidad

Existen algunas consideraciones que se tomaron en cuenta al momento de implementar el servidor web de la ESPOL.

1. Diseño coherente con el proposito del web, de tal manera que ayude al usuario a concentrarse en el contenido.

En el servidor web de la ESPOL, el propósito del servidor es "Proveer información general sobre la Universidad y sus recursos a la comunidad de Internet, con el objeto de fomentar el conocimiento y uso del Web' (sección 5.3.2), es decir que el servidor de la ESPOL debe ser un sitio en el Web donde se informa y educa a los usuarios finales. La estetica del web entonces, ayuda a crear una ambiente de aprendizaje, sin usar colores muy Ilamativos, una gran cantidad de información gráfica (ideal para un proceso de aprendizaje) y un conjunto de ayudas navigacionales (que le indican al usuario final en que lugar del servidor web se encuentra)

2. Se ha cuidado de que la información se encuentre ordenada y que su distribución no sea aleatoria. Se hizo un estudio de las 20 mejores Universidades en la rarna de Ingenieria en los EEUU y se sacó un factor común de cada una de ellas, lo que sirvio para implementar el servidor web de la ESPOL. Tambien se

form6 un equipo en CESERCOMP (Centro de Servicios Computacionales de la Universidad) que evaluo e **hizo** sugerencias en cuanto a la organización de la información.

3. Cada pagina tiene un uso mesurado de los elementos de HTML. La intención es que la gente se fije en la importancia y contenido de la pagina.

5.6. Diseño del servidor web de la ESPOL

5.6.1. Mantener la competitividad

Esta necesidad del servidor web de la ESPOL, se ha logrado haciendo a dicho servidor:

Eficiente de operar

A nivel de usuarios se ha desarrollado un conjunto de botones y mensajes de ayuda que indicaran al usuario final la manera mas fácil de llegar a su objetivo (ver figura 5.5). A nivel de administrador, se ha unificado el mantenimiento de los CGIs y de las páginas del servidor web, en una sola interface común.

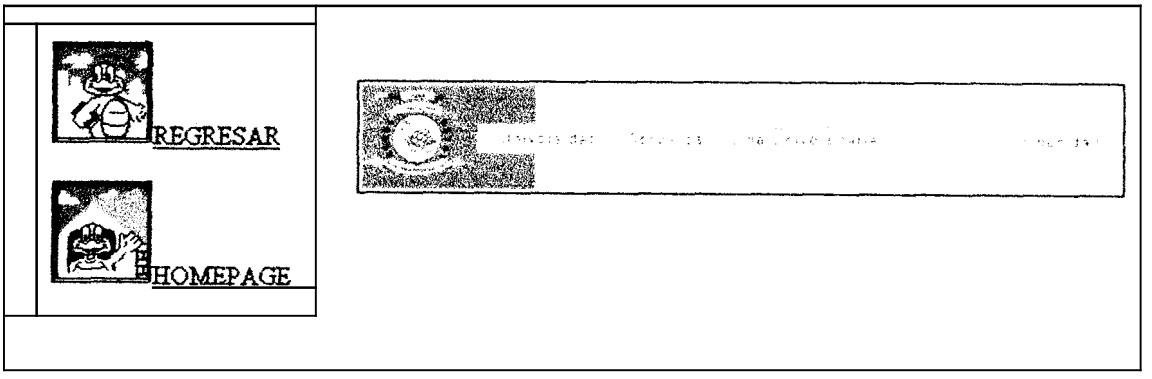


Figura 5.5.
Botones de ayuda navigacional en el servidor web de la ESPOL
Fuente: web de la ESPOL -http: www.espol.edu.ec-

Elegante y fácil de usar

Se ha utilizado imagenes para ayudar al entendimiento de la informacion, mapas sensitivos para hacer mas amigable la navegacion, maquinas de busqueda para encontrar un deterrninado tema y se ha ordenado la informacion por categorias y subcategorias (ver sección *5.3.5)*

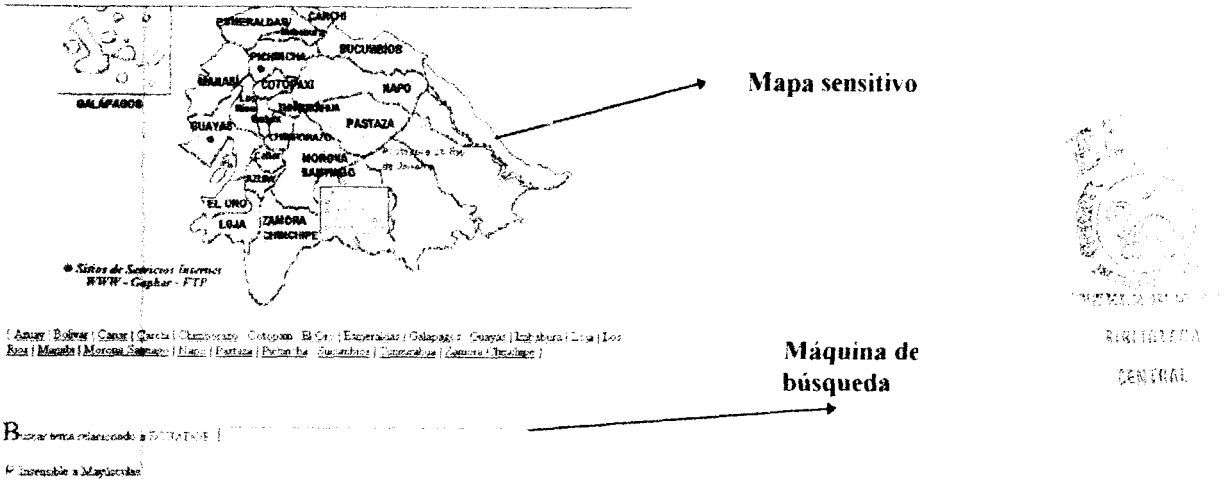


Figura 5.6.
Mapa sensitivo del Ecuador y su respectiva máquina de Búsqueda
Fuente: Servidor web de la ESPOL -http: www.espol.edu.ec-

Fácil de mantener

Se ha formado una estructura de directorios que agrupan a cada segmento o "paquete" del servidor de web. De esta manera es fácil deducir de donde viene cada pieza de información , tan solo leyendo el path (termino usado en los Sistemas Operativos que es sinonimo de directorio) que nos lleva hacia ella.

```
drwx------  2 httpd          512 Dec 11 17:08 ADMIN
drwx------ 11 httpd          512 Dec 11 04:12 CATALOGO
drwx------  7 httpd          512 Dec 23 01:00 CLASIFICADOS
drwx------  2 httpd          512 Feb  7  1996 CONFLICTO
drwx------  2 httpd          512 Feb  7  1996 CONVENIOS
drwx------ 24 httpd         1024 Dec 11 04:34 ECUADOR
drwx------  6 httpd         2560 Dec 23 00:30 ESPOL
drwx------  2 httpd          512 Jun 10  1996 FACULTAD
drwx------ 11 httpd          512 Dec  3 14:27 GRAFICOS
drwx------  2 httpd          512 Dec 11 04:34 GUAYAQUIL
drwx------  2 httpd          512 Jun 17  1996 HOMEPAGE
drwx------  2 httpd          512 Dec  4 11:07 IMAGENES_RANDOM
drwx------ 11 httpd          512 Dec 11 04:27 INFORMATIVO
drwx------  2 httpd          512 Feb  7  1996 MAESTRIA
drwx------  2 httpd          512 Feb  7  1996 MANTENIMIENTO
drwx------ 19 httpd         1536 Apr 24  1996 MANUAL
drwx------  2 httpd          512 Feb  7  1996 MUNDO
drwx------  2 httpd          512 Jun 11  1996 NOTICIAS
drwx------  2 httpd          512 Nov  7 23:13 PGP
drwx------  2 httpd         7168 May 24  1996 TUTOR
```

Figura 5.7.
Directorios del servidor web de la ESPOL formados a raiz de los "paquetes informacionales"
Fuente Directorio **home** *httpd* **docs** *del web de* **la** *ESPOL*

5.6.2. Presentación consistente y placentera

En el servidor web de la ESPOL se ha logrado dar una presentacion consistente a las páginas mas representativas del servidor. Esto quiere decir que las páginas del servidor web tienen elementos en comun que facilitan la navegacion del usuario y la

5.6.4. Empaquetamiento de la información

Para dividir al servidor web de la ESPOL en varios paquetes informacionales, se dibujo un diagrama basado en la seccion 4.3.2.1. Como se vio en esta seccion, el primer paso en la definición de los paquetes informacionales, es tomar los objetivos del servidor web y extraer de ellos todos sus sustantivos:

1. Los sustantivos encontrados fueron:

Objetivo 1:

Páginas de facil consulta y acceso con metodos de busqueda

Mapas sensitivos

Objetivo 2:

Generación de Homepages

Libro de invitados

Directorio de personas

Avisos de interes para el alumnado

Ubjetivo 3:

Correo Electrónico

Transferencia de archivos

Gopher

Objetivo 4:

Maquinas de busqueda

Servidor proxy

Objetivo 5:

Herramientas de administración del Sistema de Información

Chequeo de hiperenlaces

Estadisticas de acceso al servidor

2. Una vez extraidos los sustantivos, el siguiente paso fue armar un diagrama con ellos y agruparlos de acuerdo al topico al que estuvieran relacionados. Para el caso del servidor web de la ESPOL , se definieron 5 topicos a tratar (obtenidos luego de un estudio realizado por miembros del CESERCOMP -ver seccion 5.3.5 -). Estos son: "La Universidad", "Servicios", "Vida Universitaria", "Directorios y busquedas" y "Comunidad". Existe además un tópico adicional, que tiene que ver con las herramientas disponibles para el administrador del servidor web; este se llama "Administración" (la ubicacion de esta seccion dentro del servidor web solo la puede conocer el administrador del servidor por motivos de seguridad, ya que sus herramientas afectan la configuración de algunos archivos importantes del programa servidor web).

El diagrama quedo de la siguiente manera:

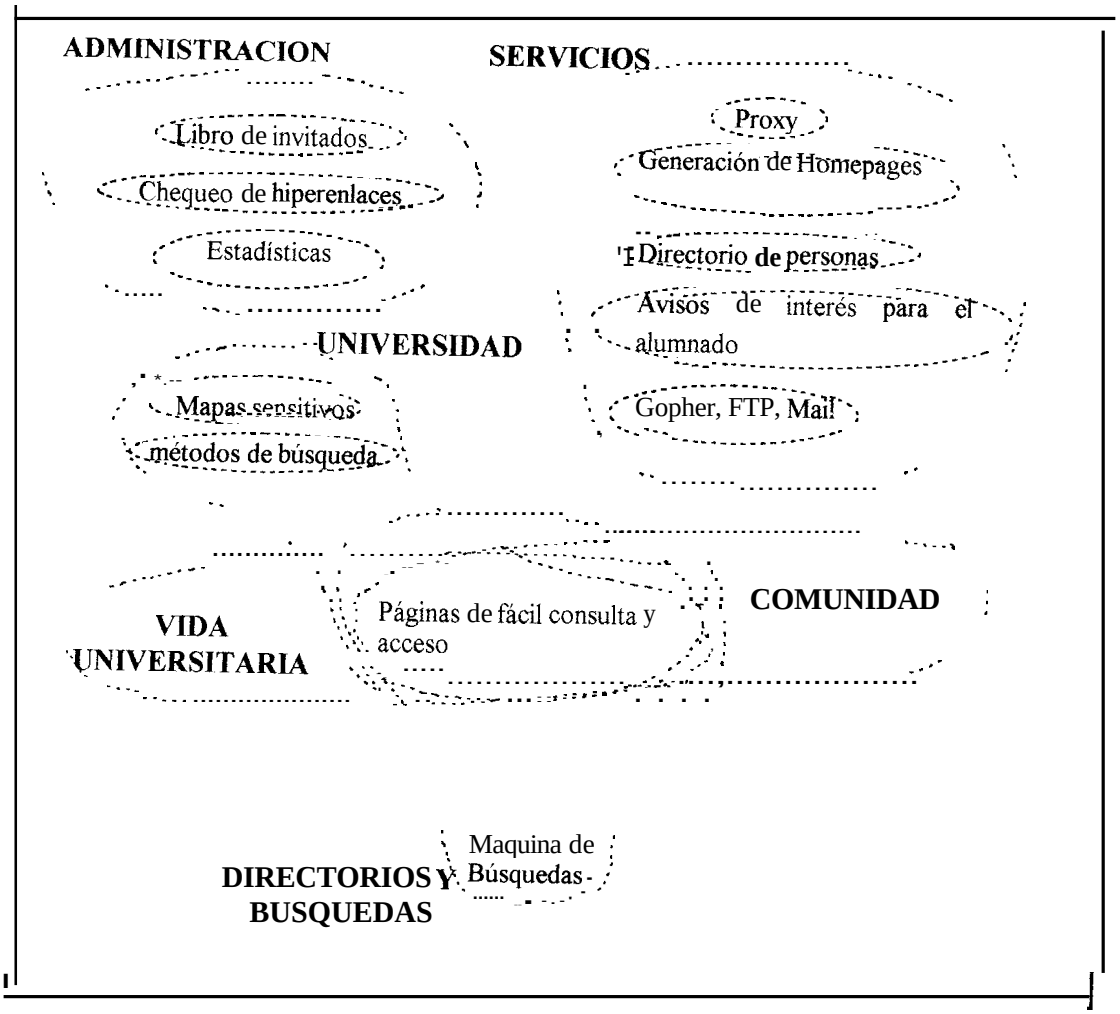


Figura **5.10.**
Paquetes informacionales del web de la ESPOL

Notese que los sustantivos de los objetivos del servidor web, fueron agrupados por topicos.

3. El tercer y ultimo paso consiste en generar una o mas hojas de hipertexto por cada paquete informacional. Un paquete informacional esta representado en la

figura anterior, por cada circulo existente. Por lo tanto, tendremos que generar la siguiente lista de hojas de hipertexto:

Universidad	1 Hoja de Hipertexto: Que contenga enlaces hacia) las otras hojas especificadas en esta seccion.
Paginas de facil consulta **y** acceso con Metodos de búsqueda	**4** Hoja de Hipertexto: 1 sobre informacion general de la ESPOL; 1 sobre Carreras de la **ESPOL;** 1 sobre Admisiones; 1 sobre proyectos en la **ESPOL**
Paginas con metodos de busqueda y Mapas sensitivos	1 Hoja de Hipertexto: 1 sobre el campus con el mapa sensitivo de la **ESPOL**

Tabla II
Hojas de hipertexto de la seccion Universidad del **web** de la ESPOL

Servicios	1 Hoja de Hipertexto: Que contenga enlaces hacia las otras hojas especificadas en esta seccion
Proxy	1 Hoja de Hipertexto: Referente al servicio Proxy y cómo configurarlo
Generación de Homepages	1 Hoja de Hipertexto: Con campos para ingreso de datos.
Directorio de personas	1 Hoja de Hipertexto: Con campos para ingreso de datos.
Avisos de interes para el alumnado	1 Hoja de Hipertexto: Con campos para el ingreso de datos.
Gopher, FTP, MAIL	1 Hoja de hipertexto: 1 Hoja con campos para ingreso de datos para lectura de correo via servidor web. Enlaces hacia los servicios de FTP y Gopher de la ESPOL

Tabla III
Hojas de hpertexto de la seccion Servicios del **web** de la ESPOL

Vida Universitaria	1 Hoja de Hipertexto: Que contenga enlaces hacia las otras hojas especificadas en esta seccion
Páginas de facil consulta **y** acceso.	1 Hoja de hipertexto: con inforrnacion sobre el "Informativo politecnico"

Directorios y Busquedas	1 Hoja de Hipertexto: Que contenga enlaces hacia las otras hojas especificadas en esta seccion
Paginas de facil consulta y acceso.	3 Hojas de hipertexto: 1 Hoja con campos de ingreso de datos para busqueda de telefonos. 1 Hoja con campos de ingreso de datos para busqueda de personas, 1 hoja con campos para ingreso de datos para busqueda de información y 1 hoja con las letras de la A a la Z conteniendo el directorio de personas

Tabla V
Hojas de hipertexto de la seccion Directorios y Busquedas del web de la ESPOL

Comunidad	1 Hoja de Hipertexto: Que contenga enlaces hacia las otras hojas especificadas en esta seccion
Paginas de facil consulta y acceso.	2 Hojas de hipertexto: 1 Hoja con datos e imagen sensitiva del Ecuador y 1 hoja con graficos sobre Guayaquil

Administración	1 Hoja de Hipertexto: Que contenga enlaces hacia las otras hojas especificadas en esta seccion
Libro de Invitados	1 Hoja de Hipertexto: conteniendo el mapamundi sensitivo.
Chequeo de hiperenlaces	1 Hoja de Hipertexto: conteniendo campos para ingreso de datos
Estadisticas	1 Hoja de Hipertexto: conteniendo gráfico y datos sobre estadisticas de acceso

De esta manera, se ha logrado determinar los "paquetes informacionales" del servidor web de la ESPOL, a la vez que se ha especificado para el implementador, cuales son las hojas mas importantes que deberá tener el servidor web.

5.6.5. Unión de páginas

Luego de haber determinado los paquetes informacionales del servidor web de la ESPOL, se procedio a enlazar las hojas resultantes, lo cual dio como resultados los siguientes diagramas de arbol:

1. Diagrama de nivel mas alto del servidor web de la ESPOL

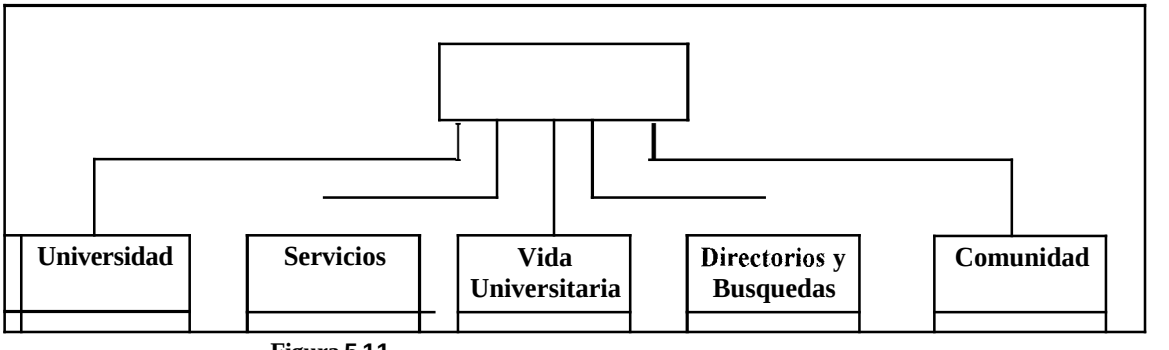


Figura 5.11.
Diagrama de árbol del web de **la** ESPOL- Nivel cero

2. Diagrama de arbol de la sección "Universidad' del servidor web de la ESPOL

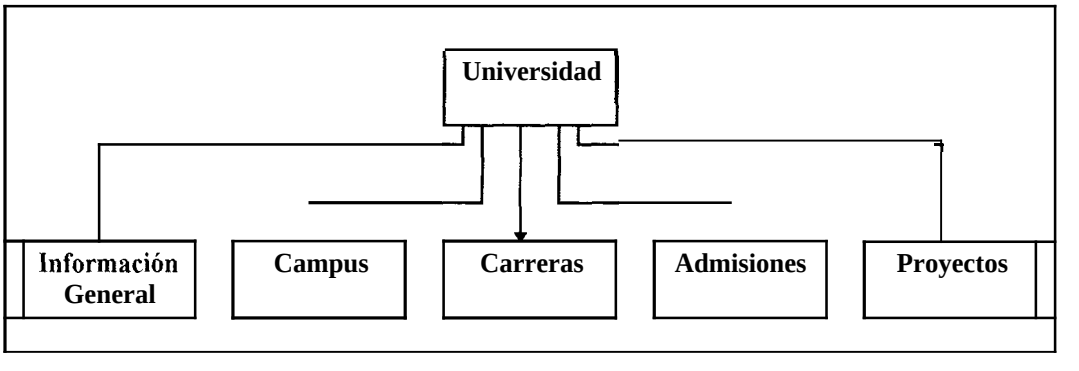


Figura 5.12.
Diagrama de árbol del web de **la** ESPOL- Rama Universidad

3. Diagrama de arbol de la seccion “Servicios” del servidor web de la ESPOL

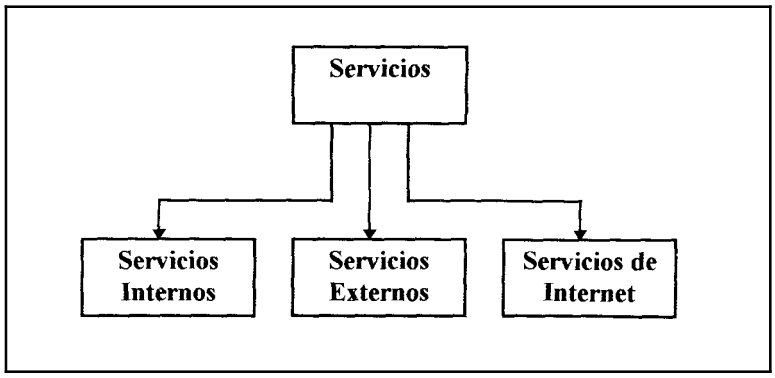


Figura 5.13.
Diagrama de árbol del web de **la** ESPOL- Rama Servicios

4. Diagrama de arbol de la sección “Vida Universitaria” del servidor web de la ESPOL

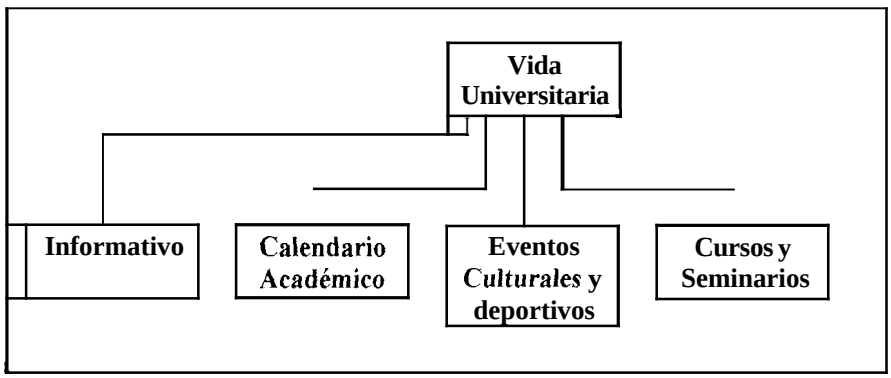
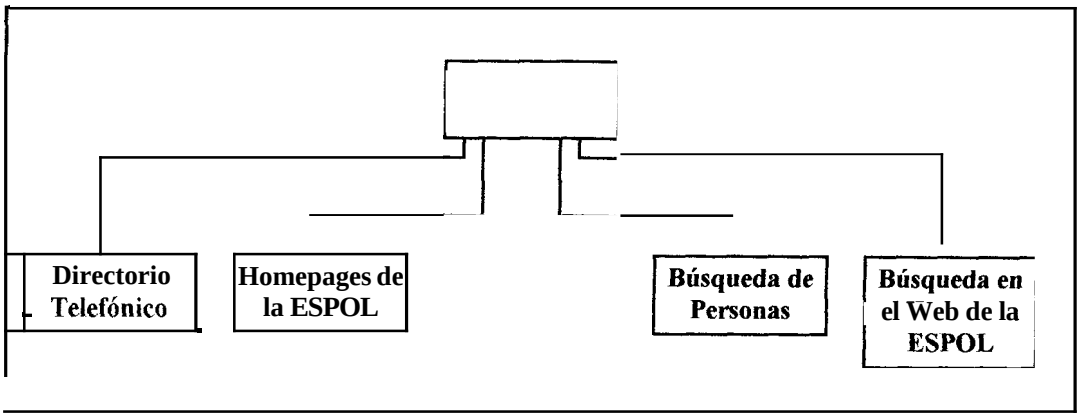


Figura 5.14.
Diagrama de árbol del web de **la** ESPOL- Rama Vida Universitaria

5. Diagrama de arbol de la seccion "Directorios y busquedas" del servidor web de la ESPOL



6. Diagrama de árbol de la seccion "Comunidad' del servidor web de la ESPOL

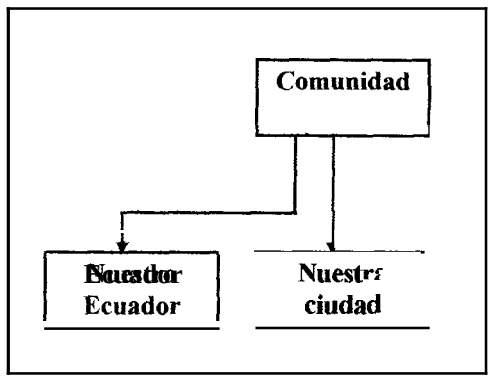


Figura 5.16.
Diagrama de árbol del web de la ESPOL- Rama Comunidad

7. Diagrama de arbol de la sección "Administracion"

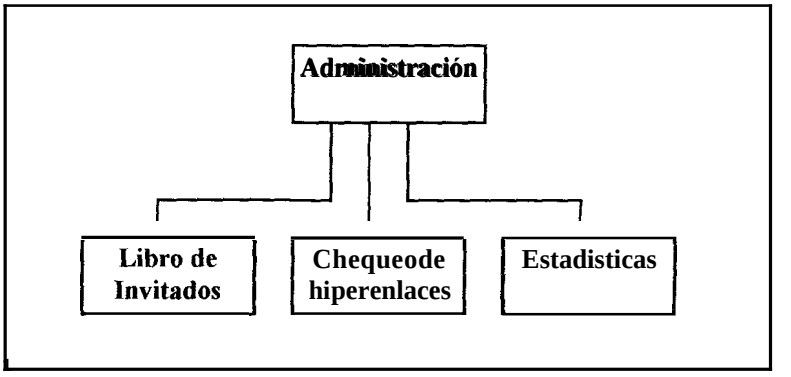


Figura 5.17.
Diagrama de árbol **del web de** la ESPOL- Rama Servicios

5.6.6. Especificación de un formato universal

Se crearon algunas plantillas a usarse en el servidor web de la ESPOL, de acuerdo al nivel en el que nos encontremos, tenemos las siguientes plantillas:

Plantilla de nivel 1 y 2:

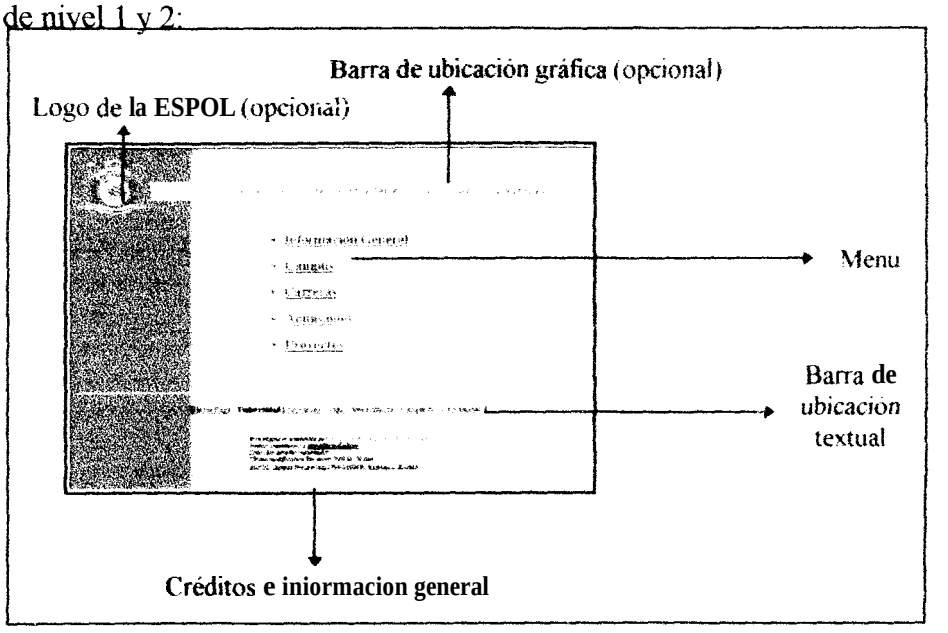


Figura 5.18.
Plantilla de hojas de hipertexto para el nivel 1 del servidor web de la ESPOL

Plantilla de nivel 3 en adelante:

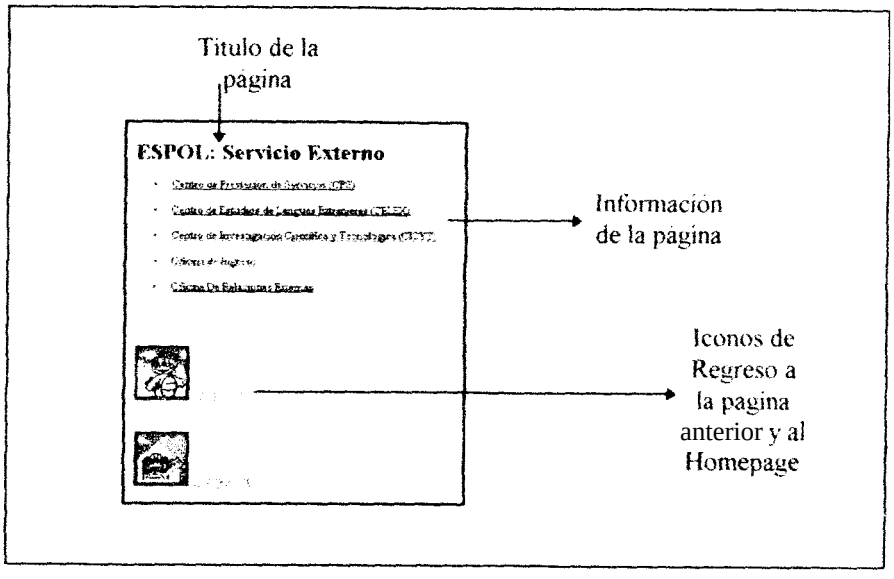


Figura 5.19.
Plantilla de hojas de hipertexto para el nivel 2 en adelante del servidor web de la ESPOL

5.6.7. Creación de un indice del web

El servidor de web de la ESPOL tiene una aplicacion (programa CGI) que se encarga de crear automáticamente el indice del servidor tomando en consideración los hiperenlaces del nivel 1 y 2. Esta es una hoja ejemplo del indice generado:

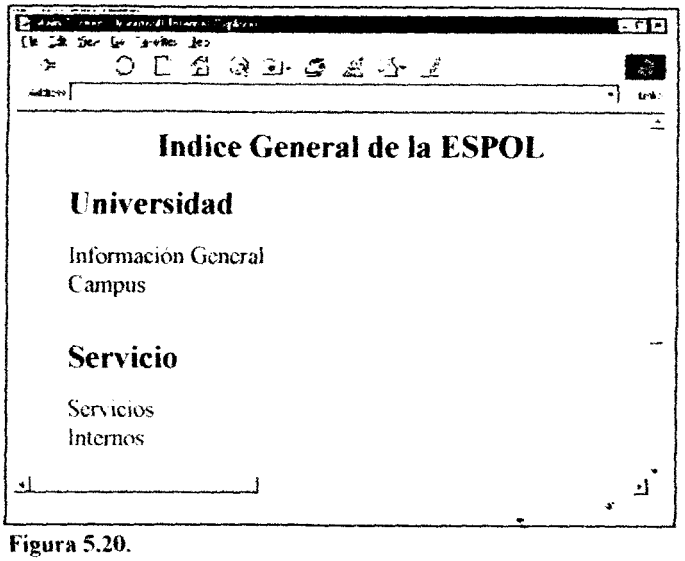


Figura 5.20.
Indice del servidor web de la ESPOL generado automáticamente

CAPITULO VI

IMPLEMENTACIÓN, PROMOCIÓN E INNOVACIÓN DEL SERVIDOR WEB DE LA ESPOL

6.1. Implementacion: tecnicas y principios aplicados en el sewidor web de la ESPOL

En el servidor web de la ESPOL, se tomaron en cuenta los siguientes aspectos de la implementación:

6.1.1. Nivel de compatibilidad HTML

En base a las especificaciones y al diseño del servidor web de la ESPOL, se utilizo en la medida de lo posible, los elementos de la especificacion HTML 2.0. Sin embargo, existen en el servidor web, ciertas hojas que fueron elaboradas con elementos

de la especificación HTML 3.0, como por ejemplo, el Homepage de la ESPOL, el cual usa tablas. Esto no es un impedimento para la legibilidad de la infonnacion, como se puede comprobar viendo la mencionada pagina en browsers como Lynx, Mosaic, entre otros.

6.1.2. Utilización de archivos GIFs entrelazados

Los gifs entrelazados (ver sección 4.4.1.2.b) fueron utilizados sobretodo en imagenes grandes, tales como el Mapamundi del Libro de Invitados o Guestbook, El Mapa sensitivo de la ESPOL y el mapa del Ecuador.

A simple vista no puede diferenciarse un gif entrelazado de un gif no entrelazado. Es solo en el momento de la carga de dicho gif, que se puede observar que un **gif** entrelazado permite observar detalles de la imagen, aunque sea en baja resolución,

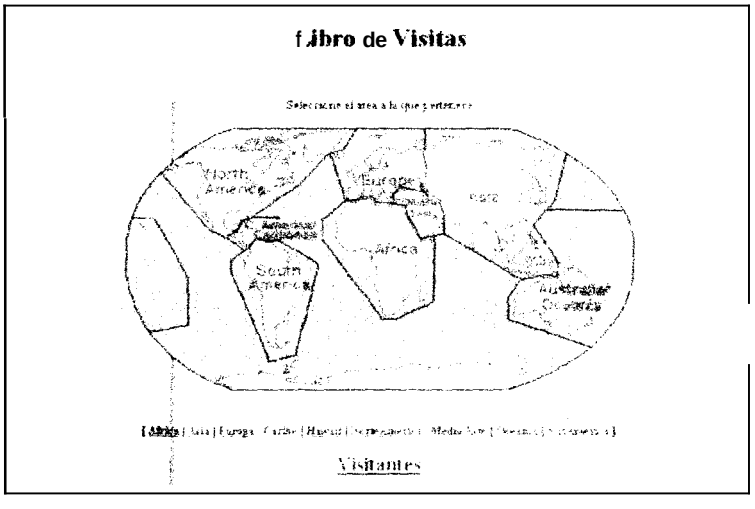


Figura 6.1.
Utilización de GIFs entrelazados en el web de la ESPOL
Fuente: Archivo ESPOL guestbook.html del web de la ESPOL

6.1.3. Uso de los elementos de HTML: HEIGHT (altura) y WIDTH (ancho)

Estos elementos son usados en el Homepage de la ESPOL, para conocer de antemano el tamaño de la imagen que se carga aleatoriamente. Veamos el ejemplo:

Figura 6.2.
Uso de elementos HTML para optimizar cargada de imágenes
Fuente: Servidor web de la ESPOL

6.2. Aplicaciones CGI implementadas

Al hablar de CGIs implementados, nos referimos a aquellos programas creados o instalados en el servidor web de la ESPOL.

Para elaborar tales CGIs se necesitaron de algunas herramientas, las cuales nombramos a continuación:

1. Interpretador Perl para Solaris 2.4

2. Compilador gcc para Solaris 2.4 y compilador cc para SunOS 1.4.3

3. Editor de Hipertexto Hotdog (www.sausage.com)

4. Editor de Imagenes: LviewPro 1B, Aldus Photostyler, Adobe Photoshop

A continuación, explicaremos cada CGI existente en el servidor web de la ESPOL, dividiendo cada uno de ellos en 4 segmentos explicativos: 1. Objetivos del CGI.- 2. Descripción del CGI .- 3. Particularidades del CGI.- 4.Técnicas empleadas la elaboración del CGI.

6.2.1. Directorio de Homepages

Objetivo:

Formar el directorio de usuarios de la ESPOL y fomentar el aprendizaje de HTML

Descripción

El "Directorio de Homepages" consta de dos partes: La primera parte la constituye un programa que recibe la información de un usuario a traves de una FORMA del servidor web, y la reenvía convertida en hipertexto, via correo electronico, al usuario que ingresó

automatica. La segunda parte la constituye otro programa, el cual permite registrar a un usuario en el "Directorio de Homepages", una vez que ha seguido el procedimiento del programa anterior. El usuario debe tener en su directorio personal del Sistema Unix de ESPOL (en caso de ser miembro de la ESPOL) , o en su directorio personal de otro Sistema foraneo (en caso de ser visitante), el archivo de hipertexto esperado por el servidor web correspondiente. Para aclarar esto veamos dos ejernplos:

En el primero, supongamos que un usuario de la ESPOL de username: slima, desea colocar su homepage en el "directorio de homepages". Para esto, el ha creado su homepage con la herramienta automatica de creación de Homepages de la ESPOL. Siguiendo las instrucciones que le llegan via email (enviadas por el primer programa CGI mencionado), genera un archivo llamado index.html y lo coloca en el directorio public-html de su directorio personal, Una vez hecho esto, trata de invocar su Homepage con la siguiente dirección:

```
http://www.espol.edu.ec/~slima
```

Que es el formato estandar utilizado para Homepages en la actualidad. El usuario logra accesar al URL con exito. De esta manera el usuario ha chequeado que el URL exista. Una vez hecho esto, procede a registrarlo en el "Directorio de Homepages", el cual comprobara nuevamente la existencia del URL y la identidad del usuario (chequeara si el

usuario es un estudiante, profesor, administrativo o visitante). El usuario se coloca en la letra L, por su apellido (Lima), y de esta manera queda ya registrado en el directorio.

Supongamos un segundo ejemplo, pero esta vez con un visitante al servidor web de la ESPOL . El visitante ya tiene su homepage creado, por lo cual no necesita utilizar la herramienta de creación automática de Homepages. Pasa directamente a registrarse en el "directorio de homepages". Ingresa su URL que es el siguiente:

```
http://www.telconet.net/-asalazar
```

Del cual se puede deducir que el username es asalazar, dentro del servidor web www.telconet.net. Se ingresa en el "directorio de homepages" en la sección "visitantes" y la entrada queda registrada exitosamente.

En los dos casos, si el usuario desea volverse a ingresar, el CGI no lo dejara puesto que se ha restringido a uno, el numero de veces que un usuario puede estar registrado en el directorio de homepages.

Además, el proceso de adicion de un usuario cualquiera al "directorio de homepages", implica que se valida su URL (se valida la existencia del archivo HTML dentro del host

especificado) y su dirección electrónica (por ejemplo slima@espol.edu.ec) es validada para comprobar si pertenece o no a la ESPOL.

Particularidades:

Programa hecho en Perl. Es un programa Multiusuario, con validación de que el URL exista (utilizando sockets de TCP/IP). En el caso de que, quien se añada a la lista sea un miembro de la ESPOL, se valida que este sea alumno, profesor o administrativo.

Técnicas empleadas en el Directorio de Homepages

A continuación se detallan ciertas partes de codigo mas relevante, del directorio de Homepages de la ESPOL.

1. Programacion Multiusuario:

El "Directorio de Homepages" es una aplicación capaz de atender multiples requerimientos secuencialmente. Para esto, se implement6 lo siguiente:

Un archivo de bloqueo llamado "lock", que se crea cada vez que un usuario final accesa a un recurso crítico (que en este caso es el archivo de hipertexto en donde se ingresara el usuario). Este archivo (lock) indica a los demas procesos que desean accesar al recurso critico, que tal recurso esta siendo usado. Mientras el

recurso esta siendo usado, los demas procesos permanecen esperando que el recurso sea liberado *(en un lazo* ***while*** *infinito).* Una vez liberado el recurso, el archivo de bloqueo "lock" es borrado y se procede a la atencion del siguiente proceso, quien a su vez creara nuevamente el archivo de bloqueo.

```
while (1){                          #lazo infinito
        if (-e $lock){              #Existe ya el archivo lock?
        next;                       #si existe, espere su turno
        }
        else{                       #si no existe, proceda
        open (LOCK,">$lock") ;      #se abre archivo lock
           .....
           .....                    (Acceso al Recurso Crítico)
   close(LOCK);                     #fin de la sección crítica
        unlink($lock) ;             #Se borra el archivo lock
        last;                       #se sale del lazo infinito
        }
}
```

Figura 6.3.
Segmento de código del programa "Busqueda de Homepages"
Fuente: Archivo /cgi-bin/enlaces pl del ***Web de*** *la ESPOL*

2. Validación de los usuarios miembros de la ESPOL:

Existe un procedimiento para verificar que un usuario final que pertenece a la ESPOL, efectivamente sea parte de la misma, y aún mas, se discierna si es un alumno, un profesor o un administrativo. Para esto, se chequea lo siguiente:

a. Que el usuario final exista en la base de datos de usuarios del Sistema Operativo (/etc/passwd)

b. Que el grupo al que pertenece el usuario final este correcto. Por ejemplo, si se desea determinar que un usuario final es un alumno, se compara **su registro** respectivo en **el archivo /etc/groups** del Sistema Operativo. **Si el campo "grupo" de tal registro,** indica que efectivamente es un alumno (puede pertenecer a la FIE, FIM o cualquier otra facultad), se le permite el acceso al "Directorio de Homepages". En este archivo (/etc/groups) existen hasta el momento 3 grupos: los alumnos, los profesores y el personal administrativo.

3. Chequeo de URLs externos.

El chequeo de URLs (Universal Resource Locator) es necesario puesto que de esta manera evitaremos en nuestro servidor web la existencia de enlaces que no llevan hacia ningun sitio. Para el chequeo, se utilizo sockets, los cuales son caminos de comunicacion entre dos programas, uno llamado cliente y el otro servidor, los que permiten intercambio de información entre el cliente y el servidor, a traves de una o varias redes TCP/IP. Para verificar que el URL exista, el programa cliente (programa que estamos estudiando) se conecta al servidor de web local (www.espol.edu.ec) o remoto (cualquier otro web en el mundo). Luego, envía comandos que pertenecen al protocolo HTTP, tales como "GET" o "HEAD" (ver capitulo sobre CGIs). Despues de esto, el cliente recibe una

respuesta del servidor de web, indicando la existencia o no existencia de la Hoja de Hipertexto del usuario final que se acaba de registrar. El codigo que implementa esto, se muestra a continuación:

```
if (!(connect(S,$that))) {          #conectándonos mediante sockets
  return -4;                        #si la conexión falla, retorne
}

select(S); $| = 1; select(STDOUT); # selección de STDOUT como
                                     camino de comunicacion

print S "HEAD $path HTTP/1.0\n\n"; #Envio de comando HTTP 1.0

$response = <S>;                    #recibimiento de la respuesta
```

Figura **6.4.**
Conexión con un servidor de web mediante sockets
Fuente: Archivo /cgi-bin/enlaces.pl del ***Web de la*** *ESPOL*

El siguiente cuadro muestra las posibles respuestas de cualquier servidor de web.

```
Status-Code    = "200"   ; OK
               | "201"   ; Created
               | "202"   ; Accepted
               | "204"   ; No Content
               | "301"   ; Moved Permanently
               | "302"   ; Moved Temporarily
               | "304"   ; Not Modified
               | "400"   ; Bad Request
               | "401"   ; Unauthorized
               | "403"   ; Forbidden
               | "404"   ; Not Found
               | "500"   ; Internal Server Error
               | "501"   ; Not Implemented
               | "502"   ; Bad Gateway
               | "503"   ; Service Unavailable
               | extension-code
```

Figura ***6.5.***
Posibles mensajes de retomo de un servidor web
Fuente: Tim *Berners Lee, INTERNET-DRAFT HTTP 1.0*

Talvez usted ya haya visto alguna de estas respuestas, cuando por ejemplo, un URL dado no ha sido encontrado o no esta autorizado a accesar a un documento. Estos codigos son universales y todos los servidores web daran, en similares circustancias, los mismos resultados.

6.2.2. Avisos Clasificados

Objetivo.

Contribuir para que el web sea utilizado como una herramienta de difusion de infonnacion de toda índole.

Descripción.

Los Avisos Clasificados estan constituidos por **3** partes. La primera de ellas consiste en un programa CGI que recibe los datos del Aviso Clasificado a traves de una FORMA en el servidor web. Miembros y no miembros de la ESPOL, pueden someter un aviso clasificado. Además se valida, solo en el caso de los miembros de la ESPOL, que tal miembro exista en la base de datos de Unix (archivo letcipasswd). Una vez ingresados los datos, ocurren dos cosas: Se envía via correo electronico la informacion ingresada

hacia e\\l usuario y al mismo tiempo, se genera un archivo conteniendo los datos ingresados, pero en formato HTML. Esto se hace para que el aviso clasificado ingresado se quede en una "cola de espera" hasta que el usuario responda el email que le fue enviado de manera afirmativa, para que su aviso sea publicado. El mensaje enviado al usuario final tiene la siguiente forma:

```
From httpd Wed Dec 25 17:11 GMT 1996
Date: Wed, 25 Dec 1996 17:11:54 +0500
From: httpd@espol.edu.ec
Subject: 12-25-96-CV-I-68-computacion
Identificacion: CV

.....
CLASIFICADOS DE LA ESPOL
Este es un mensaje enviado automaticamente por los
Clasificados de la ESPOL. Por favor,haga un Reply a este
mensaje colocando en el cuerpo del mensaje unicamente la
palabra "OK"
----_______-_______----------
Nombre: Servio Lima
Accion: VENTA
Articulo: Laptop AST Ascentia J Series
Pentium 100Mhz 16Mb RAM
800 Mb HD. Multimedia
Precio: 2300 DOLARES
Clasificacion: computacion
Fecha: 12-25-96
Transacción: 68
-------------------------------------------------------
```

Figura *6.6.*
Mensaje enviado por los **Avisos** Clasificados de la ESPOL al usuario final
Fuente: Mailbox de usuario stest en *la ESPOL*

Notese que el usuario que envía el mensaje (campo From) es el administrador del servidor web: httpd. Este mensaje debera ser respondido con REPLY (funcion de muchos lectores de email que permite responder **a** un mensaje recibido) y debera contener en su

cuerpo la palabra OK, que significa "Mensaje aceptado para su publicación en el servidor web de la ESPOL,".

La segunda parte de los Avisos Clasificados la constituye un programa que es activado cada vez que se recibe un correo electronico en el mailbox (archivo de correo) del administrador del servidor web (usuario httpd). Este programa lee el correo entrante y busca en el contenido de este, la existencia de ciertas palabras claves en el campo Subject del correo, para reconocer si el email pertenece a una respuesta enviada por un usuario para publicar un aviso clasificado. Si **el** campo subject tiene la siguiente forma:

```
Subject: 12-25-96-CV-1-68-cornputacion
6
Subject: Re: 12-25-96-CV-1-68-cornputacion
```

Entonces el email es aceptado como perteneciente a los "Clasificados de la ESPOL" , extraido de la "cola de espera de los avisos clasificados" y puesto en la sección que le correpondiese en el servidor web de la ESPOL. El significado de los campos que conforman al Subject del correo, se explican a continuación:

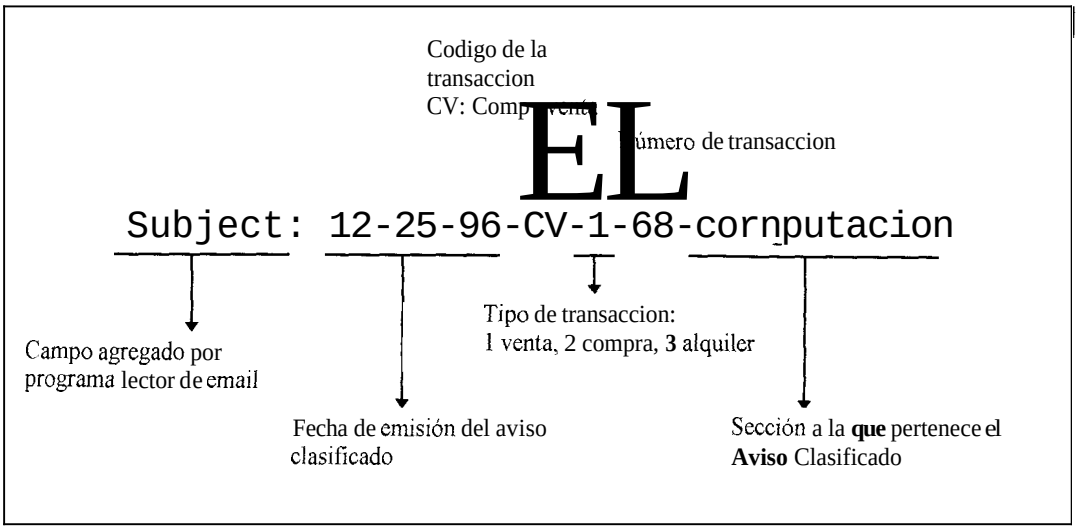


Figura 6.7.
Descripción del campo Subject enviado generado por los **"Avisos** Clasificados"

Una vez que el aviso clasificado es publicado, pasamos a la tercera parte. Esta consiste en un programa que corre periodicamente (gracias al demonio CRONTAB, que es un programa de los Sistemas Unix que permite ejecutar aplicaciones cada cierto tiempo), y que chequea la existencia de archivos superiores (en fecha de creación) a los quince dias (numero que puede ser cambiado en el archivo de crontab), en la cola de los Avisos clasificados. Si existen tales archivos, entonces, se los borra de la cola y del servidor web de la ESPOL. De esta manera termina el tiempo de publicación de cualquier aviso clasificado.

Particularidades.

Programahecho en Perl. Multiusuario. Ejecución de comandos via correo electronico.

Técnicas empleadas en los Avisos Clasificados

Este programa, al igual que el Directorio de Homepages, implementa el codigo para la atencion multiusuario. Además, presenta otras particularidades que se detallan a continuación:

1. Genera una "Cola de avisos clasificados"

Cada vez que un usuario final somete la informacion de un nuevo aviso clasificado, dicho usuario final recibira un correo electronico pidiendo que confirme los datos que acaba de ingresar. Si el usuario final responde con un OK, el mensaje es regresado de vuelta al administrador del web y añadido a las páginas de avisos clasificados por un período determinado (pueden ser 1 o 2 semanas, según lo que se haya prefijado por el administrador); esto es lo que se implementa en la figura **6.8.** Todo esto nos sirve para confirmar la existencia del usuario final que envía la informacion, de tal manera de que exista un responsable de los datos ingresados y pueda contactarselo. Si el usuario final pusiera su dirección electrónica incorrecta, el correo enviado automaticamente, jamas llegaria al usuario final que sometio la inforrnacion y por lo tanto, el usuario final jamas podría responder al email, haciendo que el aviso clasificado

nunca se publique. Este aviso, a pesar de todo, quedaria en la cola de espera, y deberá ser borrado por las herramientas de administración que se detallan mas adelante. Se chequea además, que un usuario final no pueda enviar mas de una contestación al administrador y sature el correo del mismo.

```
while(<FILE>){    #Mientras se reciba un correo
        if(/^[oO][kK]/){ #El correo contiene la
                $flag=1; #palabra OK?
                last;
        }
}
...
...

open(FILE5,">>$html") ;   #Se abre archivo temporal.
while(<FILE4>){           #Se escribe archivo tempo
      print FILE5 $_;      #ral en el archivo de -
}                           #publication de los
close(FILE5) ;            #avisos en el web de la
close(FILE4) ;              #ESPOL
```

Figura **6.8.**
Recepción via email del aviso clasificado
Fuente archivo *cgi-bin mailrcpt.pl del web de la ESPOL*

2. Ejecucion de programas via email para publicar los avisos.

Los avisos clasificados, como se ha mencionado anteriormente, son publicados una vez que el usuario responde al correo enviado por uno de los programas CGI de los avisos clasificados. Esto es posible gracias a una tecnica conocida como "ejecucion en paralelo de comandos via email". Esta tecnica se basa en el hecho de que es posible hacer que un comando se ejecute cada vez que enviamos un email a una dirección dada. Se debe utilizar para esto un archivo conocido como

".forward" que se encuentra en el directorio home del usuario (usuario httpd en el caso del administrador del servidor web). Este archivo debe contener una linea con **el** siguiente formato:

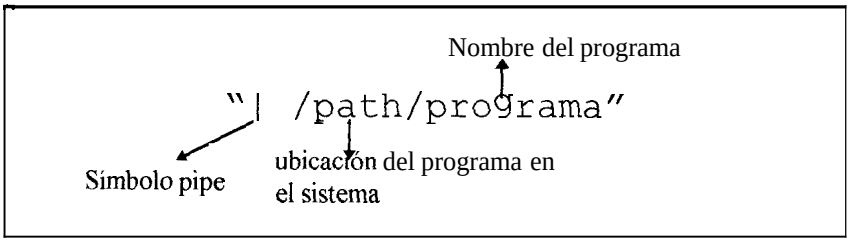


Figura 6.9.
Formato del archivo .forward para ejecución de programas via email
Fuente: Newsgroup comp.mail.sendmail de USENET

Las dobles comillas son necesarias. Solo se puede ejecutar un programa **por** archivo .forward (o lo que es lo mismo, un programa por usuario). El simbolo pipe, indica que el programa en mención (denotado como "programa"), se ejecutara en paralelo con el programa MAIL (que es el programa del Sistema Operativo Unix que recibe los email **y** los coloca en sus respectivos archivos mailbox). Esto se debe a que, el programa **MAIL** lee siempre el archivo .forward de un usuario, antes de colocar la información entrante en el mailbox del usuario y si este encuentra un comando en el, lo ejecuta.

Una vez recibido el email con la información del aviso clasificado, se procede a su publicación. El archivo HTML que se publica tiene el siguiente formato:

```
<META 12-25-96-CV-1-68-computacion>

<img src="/GRAFICOS/CATALOGO/blueball.gif">Servio Lima (email: <a
href="mailto:stest@espol.edu.ec">stest@espol.edu.ec</a>), desea para
<font size=+1>
<b>
la VENTA
</b>
</font>
<br>
Laptop AST Ascentia J Series
Pentium 100 Mhz 16 Mb RAM
800 Mb HD. Multimedia<br><b>Precio</b> 2300 DOLARES<br>FECHA
12-25-96
<br><p>
</META 12-25-96-CV-1-68-computacion>
```

Figura 6.10.
Formato del archivo HTML generado por los "Avisos Clasificados"
Fuente Archivo /COMPRAVENTA/12-25-96-CV-1-68-computacion del web de la ESPOL

Estos archivos son almacenados en el directorio /home/httpd/docs/CLASIFICADOS/COMPRAVENTA. El elemento HTML llamado META que se observa en la figura anterior indica el comienzo y fin del archivo, lo cual servira para que posteriormente pueda ser eliminada esta entrada del archivo de Avisos Clasificados.

6.2.3. Busqueda de Personas y Teléfonos

Objetivo.

Facilitar al usuario final la busqueda de personas o de telefonos relacionados con la ESPOL

Descripción

La busqueda de personas y de telefonos son dos programas que se explicaran por separado.

La busqueda de personas esta constituida por 2 componentes: El primero es un programa CGI hecho en C, que recibe los datos de un listbox (lista de datos relacionados) de una FORMA en el servidor web. Este dato es cualquier departamento, facultad, tecnologia o instituto de la ESPOL que haya sido registrado en el arreglo de datos (conjunto de datos agrupados en una variable, dentro del programa en mención) del programa. **A** continuación mostramos una parte de dicho arreglo:

```
char *string[] ={
        "BIBLIOTECA\0",
        "Biblioteca\0",
        "CEAA\0",
        "CECYP\0",
        "CELEX\0",
        "CENAIM\0",
        "CESERCOMP\0",
        "CETE-FIM\0",
        "CICYT\0",
        "CONTABILIDAD\0",
        "CPS\0",
        ...
        ..
        }
```

Figura 6.11.
Departamentos de la ESPOL agrupadas en un arreglo
Fuente: Archivo /cgi-src/personas.c ***del*** *servidor* ***web de*** *la ESPOL*

Una vez recibido el dato del departamento, se procede a la busqueda de todas las personas miembros de aquel departamento, en el archivo /etc/passwd. Esta lista es impresa en hipertexto, y contendra la dirección email de cada persona junto con datos sobre su usuario en el Sistema Unix de la ESPOL (esto se genera mediante un finger - aplicacion Unix que me permite ver datos como: directorio home; lectura de email; Nombre completo, etc- al usuario).

El segundo componente de la busqueda de personas lo constituye un programa CGI que recibe como datos cualquier cadena de caracteres completa o incompleta, sobre un usuario de la ESPOL. Por ejemplo. Si buscamos al usuario Servio Lima, podremos realizar las siguientes búsquedas:

Busqueda por Servio, bdsqueda por Lima o cualquier combinación de estas palabras.

Supongamos que conocemos el número de matricula de esta persona; entonces podremos buscar por tal numero: Busqueda por 0906487

Supongamos que Servio Lima pertenece a la FIEC (Facultad de Ingenieria en Electricidad y Computacion). Podremos buscar entonces por FIEC.

La busqueda es hecha en Perl y se utiliza como datos el archivo /etc/passwd del Sistema.

La busqueda de telefonos consiste de un programa CGI que recibe como datos, cualquiera de los siguientes campos: Nombre del usuario, Area a la que pertenece, Número telefonico, o una combinación de cuaiquiera de los tres, como se ve en la siguiente figura.

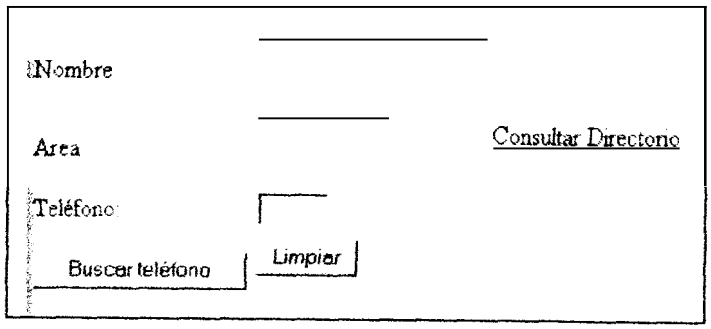


Figura 6.12.
FORMA para ingreso de datos en la Busqueda de Teléfonos
Fuente: Archivo telefonos.html del servidor web de la ESPOL

La busqueda se la realiza en base a un conjunto de archivos que deben ser ingresados por la persona que tenga la inforrnacion de los numeros telefonicos al dia (Jefe del departamento de Voz y Datos). Generalmente, estos datos se encuentran en formato de Word para Windows y ordenados en tablas. La macro hecha en Word de la figura **6.13.** ayudara a convertir tales datos en texto. El administrador deberá encargarse de enviar el archivo resultante hacia el servidor de la ESPOL, mediante un simple FTP (al directorio /home/httpd/docs/teléfonos)

```
Sub MAIN
SeparadorDeCampo$ ":"
TablaEnTexto .ConvertirEn = 3
EdiciónCopiar
ArchivoNuevo .Plantilla = "Normal", .NuevaPlantilla = 0
EdiciónPegar
ArchivoGuardarComo .Nombre = "DOC2.TXT", .Formato = 2,
.BloquearAnotación = 0, .Contraseña = "", .AgregarAAUR = 1,
.EscribirContraseña = "", .RecomendadoSóloLectura = 0, .FuentesIncrustadas
= 0, .FormatoImagenOriginal = 0, .FichasDeDatos = 0
End Sub
```

Figura 6.13.
Macro en Word 6.0 **para** conversion de tablas de telefonos a texto

Particularidades.

Programas hechos en Perl y en C. Sensibilidad a letras mayúsculas. Soporte de Substrings. Programa de busqueda basado en Perl.

Técnicas empleadas en la Busqueda de Personas y de Teléfonos

La Busqueda de Personas y de Telefonos son dos programas que tienen algunas peculiaridades interesantes. Analicemos cada uno de ellas.

La busqueda de personas es un programa hecho en C. El algoritmo de busqueda utilizado fue el de busqueda lineal simple. El programa accesa al archivo /etc/passwd del Sistema Operativo, (gracias a funciones propias de C), lo cual le permite tener información actualizada de los usuarios del sistema. Debido a la gran importancia que tiene este archivo en esta y otras aplicaciones mencionadas, a continuación se da el formato de un registro del archivo /etc/passwd.

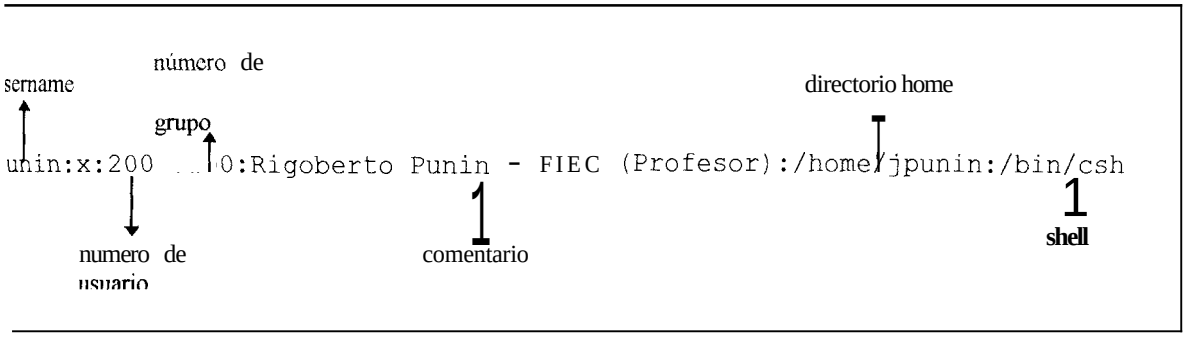


Figura 6.14.
Formato de un registro del archivo /etc/passwd
Fuente: Archivo /etc/passwd del servidor Unix (Solaris 2.5) de la ESPQL

La busqueda se la realiza por Facultad, Institutos, departamentos, etc (FIE, CESERCOMP, Rectorado, etc). Además, existe un miniprograma hecho en Perl, que realiza la busqueda en el mismo archivo, por nombres de personas, numero de matricula, cargo que ocupa, etc, que sirve de complemento al programa hecho en C en mención. **A** continuacion, se comparan dos fragmentos de ambos programas, como un ejemplo de las diferencias en la implementación de CGIs que constituye el hacer un programa en C o en Perl.

La figura inferior muestra un fragmento de codigo en C, en el que se realiza la busqueda de un usuario en el archivo /etc/passwd. Existen funciones propias de C que realizan esta tarea(setpwent, getpwent, entre otras).

```
for(x=0;x<=m;x++){                          #Mientras existan datos
            setpwent();                       #resetear el archivo /etc/passwd
      while((p=getpwent()) !=NULL){ #Mientras no se termine el archivo
            if(buscar-string(string[atoi(entries[x].val)],p->pw_gecos))
                        imprimir(p);      #se realiza la busqueda de informa
                  }                           #ción y se imprime el resultado
            }
```

Figura 6.15.
Implementación simple de una busqueda hecha en *C*
***Fuente Archivo cgi-src personas c del web de** la ESPOL*

En Perl nos evitariamos todo este codigo anterior, simplemente utilizando el comando 'index" (el cual permite buscar un string dentro de otro), como veremos a continuacion.

```
while(<PW>){    #Mientras el archivo /etc/passwd no se termine
        ...
        ...      #Se comparan el campo de comentarios con el
                 #nombre ingresado
        if(index($gcos2,$FORM{'nombre'}) ge 0){

              print FILE "<b>$gcos:</b> <a href=\"/cgi-
bin/correo.pl?$user\">$user\@espol.edu.ec</a><p>";
        }        #Se imprime el nombre ingresado, en Hipertexto
        ...
        ...
}

close(PW);      #Se cierra el archivo /etc/passwd
```

Figura 6.16.
Busqueda de información en Perl con el comando "index"
***Fuente** Archivo /cgi-bin busqueda2.pl del web de la ESPOL*

En la busqueda de telefonos, se utiliza la capacidad de Perl para busqueda de infonnacion. Pero esta vez, se utiliza otro operador, la doble barra inclinada, el cual realiza la misma funcion del comando "index". Pero ahora, el codigo es aun mas simple, como se muestra a continuación:

```
if(/$nombre1/ && /$areal/ && /$telefono1/){#Búsqueda por nombre, area
                    $flag=1;                #o telefono
                    &imprimir_datos($_);    #el operador // realiza
}                                           #la busqueda
```

Figura 6.17.
Busqueda de informacion en Perl utilizando **el** comando //
***Fuente**: Archivo /cgi-bin/telefonos.pl del Web de la ESPOL*

En la figura anterior se ve que con el simple uso del operador //, se pueden realizar busquedas sobre archivos. Esto le da una gran ventaja a Perl sobre otros lenguajes de programacion, cuando hablamos de programacion en el web, pues hay que recordar que

lo que mas se maneja en el mundo del WWW son archivos (envio y transmision de hojas de hipertexto, graficos, sonidos, etc)

6.2.4. Lectura de correo en Hipertexto

Objetivo.

Convertir al browser en la herramienta de mayor uso en la ESPOL, unificando en él, servicios de Internet como el Correo electronico.

Descripción

Este programa permite leer el correo electronico de un usuario de la ESPOL (y solamente de la ESPOL) a traves del servidor web, convirtiendo el archivo de correo (llamado mailbox) del usuario a hipertexto. Para esto, el usuario debe ingresar su login y su password, informacion que viajara encriptada hacia el servidor (a traves del mecanismo de seguridad conocido como SSL). Luego, el programa CGI recogerá esta información, la cual le servirá para poder hacer un FTP con tales datos y acceder al archivo mailbox del usuario (que se encuentra en el directorio /var/spool/mail). La petición del login y password del usuario es necesaria ya que el programa servidor web de la ESPOL corre como un usuario normal del sistema (Esto es, corre como el usuario

httpd), el cual no tiene permiso de lectura sobre el mailbox de cualquier otro usuario, por lo que necesariamente necesita tener los datos de login y password para poder leer el correo. Luego de realizado el FTP, el CGI ejecuta un programa conocido como Hypermail (http://www.eit.com/software/hypermail/hypermail.html), el cual es una aplicacion escrita en Perl que permite convertir texto a hipertexto. Una vez convertida la información a hipertexto, se procede a mostrarla por pantalla. Se creara un directorio por cada usuario que lea su correo electronico.

Particularidades.

Programa hecho en Perl. Multiusuario. Seguridad implementada a traves de los Secure Socket Layer . Transferencia de archivo utilizando FTP.

Técnicas empleadas en la lectura de correo en Hipertexto

La lectura de correo via web, es una herramienta unica en el mundo del WWW. Esta herramienta esta basada en Hypermail, un software de dominio público (que se lo encuentra en http://www.hypermail.com) que permite leer el correo electronico de un ólo usuario (generalmente el usuario de una lista de discusion). Se adaptó de tal manera ste programa, que actualmente permite ver el correo electronico de cualquier usuario

final, haciendo un FTP a su archivo de la cola de correos. Debido a los permisos del sistema operativo, la unica manera de accesar a tal archivo es ingresando el Login y el password del usuario final y haciendo un FTP de dicho archivo. El archivo mailbox del usuario es guardado temporalmente en el directorio del Administrador del servidor web y luego de ser convertido a varios archivos de hipertexto (gracias a Hypermail), es borrado. Además, se implementaron mecanismos de seguridad a traves de los SSL, de tal manera que la información que estuviese leyendo el usuario final, viaje en forma encriptada (al igual que su user y password). La lectura de correo via web tiene capacidades de lectura y contestación del correo, lo cual lo convierte en un lector de email simple. A continuación veremos los detalles mas importantes de su código:

```
#FTP del archivo de correo de un usuario
#.script es un archivo que contisne comandos FTP
open (FTP, "|ftp -in < .script > output.txt");
close(FTP);
...
#Creación de un directorio temporal para guardar el correo
mkdir("${mail_root}${numero}",0700)
...
#Conversion del correo a hipertexto
open(MAIL, "|$hyper -x -m \"/home/httpd/cgi-bin/$FORM{'login'}\" -d
${directorio}");
...
#Borrado de archivo temporal
unlink("/home/httpd/cgi-bin/$FORM{'login'}");
```

Figura 6.18.
Conversión de un archivo texto a hipertexto mediante hypermail
Fuente Archivo cgi-bin mail5.pl del Web de la ESPOL

Como se ve en la figura 6.18., es posible hacer una conexión a un servidor FTP, utilizando simplemente comandos del cliente de FTP. No es necesario abrir un socket con el servidor (esto seria un tanto mas complicado puesto que FTP trabaja con dos puetos: 20 y 21).

6.2.5. Interfaz unificada de administracion del servidor web

Objetivo.

Administrar y controlar todos los programas previamente mencionados , a traves de una interfaz unica. Realizar otras tareas de administracion, tales como: limpieza de archivos temporales, ejecucion de programas, actualización de estadisticas y Guestbook, etc.

Descripción

La interfaz unificada de administracion del servidor web, esta fonnada por un conjunto de programas CGI que permiten la ejecucion de las siguientes tareas administrativas:

a. Ejecucion inmediata de tareas del CRONTAB.

El crontab es una aplicacion de los sistemas Unix que permite ejecutar tareas periodicamente. Existen hasta el momento, 6 tareas relacionadas con el servidor web de la ESPOL que se deben ejecutar periodicamente:

1. Actualización de estadisticas de acceso del servidor web de **la** ESPOL

Opcion que generara un reporte de los accesos al servidor web; ver seccion 6.2.6

2. Actualización de **las** entradas en el Guestbook

Opcion que leerá los nuevos comentarios de los visitantes **y** actualizara la lista de "comentarios del Guestbook"; ver seccion 6.2.6

3. Borrado de los Avisos Clasificados expirados

Opcion que borrara aquellos avisos cuyo período de vida sea superior a dos semanas (limite de vida prefijado; ver seccion 6.2.1)

4. Borrado de los archivos generados por la lectura del correo en hipertexto

Opcion que borrara el directorio **y** los archivos generados por la lectura del correo en hipertexto (ver seccion 6.2.4)

5. Borrado de archivos lock

Opción que borrara los archivos de bloqueo (archivos lock) generados por la mayoría de las aplicaciones del servidor web de la ESPOL y que sirven para bloquear un recurso mientras un usuario lo accesa, para que otros usuarios esperan hasta que el recurso es liberado de tal forma que dicho recurso no se corrompa o se pierda (ver sección 6.2.1.). El borrado es hecho por la misma aplicacion que genera el archivo lock, pero puede ocurrir que un usuario haya perdido la conexión con el servidor web (fallas con el servidor, la red, etc) y el archivo lock haya quedado creado, lo que ocasionaria que la aplicacion quedara bloqueada indefinidamente.

6. Limpieza del directorio temporal del servidor Proxy

El Servidor Proxy instalado en la ESPOL, tiene un area destinada al cache de la información (memoria temporal que almacena los archivos de hipertexto de servidores webs recientemente visitados). Esta area se llena cada vez que un usuario usa el servidor proxy y el servidor visitado por este, no ha sido visitado previamente. Esto ocupa espacio en el directorio del usuario httpd (Administrador del servidor web de la ESPOL), que tiene una cuota limitada. Se recomienda borrar los datos semanalmente.

b. Desbloqueo de usuarios de la seccion "Directorio de Homepages"

En el Directorio de Homepages de la ESPOL (ver seccion 6.2.1.), luego de que un usuario se ha registrado exitosamente por primera vez, tal usuario no puede volverse a registrar (esta es una restricción del programa para evitar que un usuario sature al servidor web ingresandose indiscriminadamente mas de una vez). Escogiendo esta opcion, se puede permitir a un usuario volverse a registrar, en casos en los que desee incluirse en otro grupo de usuarios (profesor, alumno, etc), su URL ya no exista, etc.

c. Chequeo de Hiperenlaces

Con esta opcion se pueden chequear los hiperenlaces incorrectos de una hoja, paquete infonnacional o de todo el servidor web de la ESPOL. El resultado del chequeo de hiperenlaces puede ser enviado por pantalla o via correo al administrador del servidor web. Para realizar esta labor se utilizo una herramienta llamada Verify Links que podra ser encontrada en http://wwwcs.dartmouth.edu/~crow/lvrfy.html.

d. Generación del índice del servidor web de la ESPOL

Con esta opcion se genera un indice del servidor web de la ESPOL, en base a un archivo que contiene los nombres de las hojas a ser indexadas en el servidor web. Este archivo lo podra encontrar en el directorio /home/httpd/cgi-bin/indice.txt. Luego de esto, podra ver el indice generado en: http://www.espol.edu.ec/directory.html

e. Actualización del indice de las Maquinas de Busqueda del servidor web

GLIMPSE, la maquina de busqueda de información en la ESPOL, genera varios archivos indices por cada seccion indexada. Si una seccion cambia, debe actualizarse el indice, para que los resultados de la busqueda esten actualizados. Las secciones a las que se hace referencia son: Todo el servidor web, la seccion del catálogo de la ESPOL, el informativo politécnico y la seccion sobre Ecuador.

f. Envio de email a todos los usuarios de la ESPOL

Con esta opcion, el administrador del servidor web puede enviar un mensaje que llegara a todas las personas miembros de la ESPOL, con el objeto de anunciar o promocionar algun cambio, mejora, actualización, tema de interes, etc. del servidor web.

Particularidades.

Conjunto de programas hechos en Perl. Se ejecutan seleccionando el hiperenlace que describe la tarea. Solo se puede ingresar a ellos con el password del administrador. (Usando el esquema htpasswd de **NCSA)**

Técnicas empleadas en la Interfaz unificada de Administración del servidor web.

Mediante esta interfaz, se trata de agrupar todas las tareas que permitiran dar mantenimiento al servidor web de la ESPOL. Si no existiese esta interfaz, no fuera posible seguir manteniendo la funcionalidad (Índice de busqueda actualizados, Limpieza de espacio en disco de los archivos temporales) que el servidor web de la ESPOL actualmente tiene.

La lnterfaz unificada del web de la ESPOL la conforman un conjunto de programas CGI, los cuales son simples en su estructura. **A** pesar de esto es importante hacer referencia a su codigo, ya que por primera vez en el web de la ESPOL se hace uso de funciones como **fork** (funcion de Unix que permite derivar uno o varios procesos hijos de un proceso padre) y de la funcion **exec** (funcion que permite la ejecucion de comandos del Sistema Operativo en un programa). Estas funciones nos permitiran dividir al programa en dos procesos, un proceso hijo (el cual ejecutara un comando del sistema con la funcion **exec)**

y un proceso padre (el cual esperara la tenninacion del proceso hijo, con la funcion **wait).** El uso de estas funciones, se realizó a manera de ejemplo, para demostrar que tambien son una alternativa viable al momento de programar un CGI que tenga que ejecutar comandos del sistema. El codigo de una de estas aplicaciones se muestra a continuación:

```
unless(fork){  #division de un proceso en padre e hijo
#proceso hijo
exec("/bin/rm -fr $dir/mail*");
}
#proceso padre
wait;
```

Figura 6.19.
División **de un proceso padre en un proceso** hijo **para la** ejecución **de un comando del** SO
Fuente ***Archivo cgi-bin borradocorreo pl del Web de la ESPOL***

6.2.6. Estadísticas y Guestbook

Dbjetivo.

Permitir que el administrador del servidor web conozca el status del servidor, los :nlaces mas visitados, la opinion de los usuarios finales, fallas en el servidor, etc.

)escripción

.as estadisticas del servidor web de la ESPOL permiten al administrador del servidor, onocer acerca de los siguientes parametros: Número de accesos totales a la semana,

hojas del servidor mas visitadas, lugares que mas nos visitan por region (Sudamerica, Norteamerica, Asia, etc) o por dominio (ESPOL, Ecuanet, Telconet, etc)

El Guestbook de la ESPOL es un programa hecho en Perl que recoge las opiniones de los visitantes del servidor. Esto sirve como una cortesia del servidor web a todos sus visitantes y como un medio para saber cual ha sido la aceptacion del servidor en la comunidad de Internet.

Particularidades.

Las Estadisticas fueron implementadas con seguridad usando el esquema htpasswd de NCSA. El Guestbook, valida que un usuario exista, enviando un email al usuario. Si el email rebota (regresa al administrador del servidor), se asume que el usuario no existe y por lo tanto el mensaje es borrado de la cola del Guestbook.

Técnicas empleadas en las Estadisticasy el Guestbook.

. Las estadisticas del servidor web de la ESPOL son generadas por un paquete público e Internet conocido como WUSAGE. Este es un programa hecho en C, que lee el rchivo LOG o de accesos del servidor de la ESPOL, y luego grafica los resultados. Para onvertir los datos numericos a graficos, se utilizo una librería hecha en C llamada

GDImage. Tanto el programa estadistico como la librería fueron hechos por Thomas Boutell (http://www.boutell.com para mayor informacion). Tiene la particularidad de que se basa en un archivo de configuración (llamado wusage.conf), el cual permite excluir o incluir de las estadisticas ciertos archivos (imagenes por ejemplo: gif, jpg, etc), especificar grupos de dominios (Suram= ec -Ecuador-, pe -Perú-, co-Colombia-, etc; Norteam: us -EEUU-, edu - Universidades de EEUU-, gov -Gobierno de EEUU-, etc), entre otras tareas. La desventaja de este programa (La versión 3.2 -usada en la ESPOL- es freeware), es que no puede manejar grandes cantidades de informacion, puesto que utiliza arreglos y no punteros para almacenar los datos. Esto hace que la información que puede manejar sea limitada. Es por esto que se recomienda mantener los datos en el log del servidor web de la ESPOL (logs/access_log) hasta aproximadamente la 14ta o 15ta semana. Luego encerar tal archivo para que el registro de accesos empiece de nuevo.

. El Guestbook de la ESPOL es un programa hecho en Perl, el cual permite a un usuario nal de la ESPOL o a cualquier visitante a nuestro servidor web, el dejar sus opiniones obre el mismo. Para evitar que la informacion que se ingrese, sea malintencionada, se a utilizado un mecanismo por el cual se comprueba la existencia del casillero postal ectronico (o email) de la persona que dejo sus comentarios. Se envía un email gradeciendo sus comentarios. Si este email rebota y retorna al administrador,

inmediatamente es borrado de la cola de espera, puesto que se asume que la dirección de la persona que ingreso la información, no existe. **A** continuación, detalles del codigo:

```
while (<FILE>){                    #Se lee el correo que acaba de llegar

       if(/[Mm][Aa][Ii][Ll] [Eel[Rr}- [Dd][Aa][Ee][Mm][Oo][Nn]/){
              $flag=2;              #El correo es enviado por MAILER-DAEMON?
       }                            #MAILER-DAEMON envia mensajes que han sido
                                    #rebotados en los sistemas  Unix.

       if(/^Seccion/ && $flag eq 2){ #Se pregunta por otras secciones
              $flag =3;               #dentro del correo entrante,
              last;                   #de tal manera de estar seguros
       }                              #de que el rebote del email,
}                                     #se debió al Guestbook

..............
..............                       #Una vez confirmado que el email
                                     #que reboto corresponde al
                                     #Guestbook, se procede a
                                     #Borrarlo.

if (-e"$directorio/$region.$user.$domain"){              #Existe mensaje?
      unlink("$directorio/$region.$user.$domain");        #borrado
                                                          #del mensaje
}                                                         #Guestbook
```

Figura 6.20.
Recepción **y** borrado de un correo que ha rebotado. del Guestbook de la ESPOL
Fuente: Archivo /cgi-bin/mailrcpt.pl ***del web de*** *la ESPOL*

Si el mensaje enviado al Guestbook de la ESPOL, contiene palabras obscenas (definidas en un arreglo de datos) este tambien es borrado de la cola del Guestbook.

```
#CHEQUEO  DE OBSCENIDADES
for($i=0;$i<=$#obscenidades;$i++){
        if(index($FORM{'comentario'},$obscenidades[$i]) ge 0){
        &error("No se permiten palabras obscenas\n");
        }
```

Figura 6.21.
Chequeo de Obscenidades en el Guestbook de la ESPOL
Fuente: Archivo /cgi-bin/guestbook.pl del Web de la ESPOL

6.3. Maquina de Búsqueda: GLIMPSE

Dos de los programas de busqueda mas utilizados en el mundo del WWW son: GLIMPSE (GLobal IMPlicit SEarch) y freeWAIS (Wide Area Information System). Las tablas que se presenta a continuación recogen diversos parametros de comparacion entre estas dos maquinas de busqueda que explican el porque GLIMPSE fue considerada la mejor opción para la ESPOL.

Maquina de busqueda	estándar Z39.50	Espacio en disco	Velocidad de busqueda	Resultados de la busqueda	expresiones booleanas, palabras incompletas, sensitividad a mayusculas	Facilidad de creacidn del índice de bu squeda y control de lo que está siendo indexado
GLIMPSE	NO	Eficiente	3-10 seg	Se muestra el párrafo donde reside la palabra buscada	SI	SI
freeWAIS	SI	Ineficiente	5-10 seg	Se muestra solo el titulo del documento	PARCIAL	PARCIAL

Tabla MI1
Comparacion entre GLIMPSE **y freeWAIS**
Fuente: **Archivo /cgi-bin/guestbook.pl del Web** de **la** ESPOL

Estándar 239.50

Uno de las caracteristicas deseables en una maquina de busqueda es que utilice protocolos estandarizados a nivel mundial. En el area de comunicacion con bases de datos, el estandar mas relevante es el 239.50, que define un protocolo para la busqueda y recuperación de la información proveniente de bases de datos distribuidas en una red (como por ejemplo, la red Internet). Esto significa que el

estandar 239.50 se refiere unicamente a la manera en que un sistema de busqueda se comunica con otras bases de datos, y en particular, como tal sistema intercambia informacion que se encuentra en su base. FreeWAIS soporta este estandar, permitiendo a una maquina de busqueda de alcance local, tener contacto con otras maquinas de busqueda que utilicen el protocolo WAIS, de tal manera de poder accesar a la informacion de tales maquinas de busqueda. GLIMPSE carece de esta propiedad. Pero a pesar de esto, esta caracteristica no es necesaria, sino simplemente deseable.

Espacio en disco

Tanto GLIMPSE como freeWAIS crean un indice del contenido de las hojas de hipertexto a buscar. GLIMPSE es la mas eficiente en este aspecto ya que permite crear indices desde 2% hasta 30% del total de archivos indexados, mientras que WAIS es ineficiente, pues crea un indice que va desde el 40% o mas del total de archivos indexados.

Velocidad de busqueda

El tiempo de respuesta para ambas maquinas de búsqueda es muy bueno (3-10 seg para GLIMPSE y 5-10 seg para WAIS). Esto se debe a que ambas utilizan el mecanismo de crear un indice previo de las palabras mas comunes que se

encuentran en el total de archivos a indexar, de tal manera que la búsqueda se realiza hallando en tales indices la palabra buscada y luego accesando al archivo que contiene dicha palabra. No se hace la busqueda tradicional de accesar directamente **a** los archivos para realizar la busqueda, lo cual seria mucho mas lento e ineficiente ya que consumiria grandes recursos del sistema.

Resultados de la blisqueda

En este aspecto, GLIMPSE es superior a WAIS. GLIMPSE provee una respuesta a la busqueda, que detalla el parrafo donde se encontro la palabra, y además resalta dicha palabra dentro del párrafo. WAIS, simplemente muestra el titulo del documento y da una puntuacion (score) a los archivos (dependiendo de cuantas veces haya estado repetida la palabra en el archivo y si tal palabra se encuentra tambien en el titulo del documento) que a veces no refleja lo que el usuario desea encontrar.

Expresiones booleanas, palabras incompletas, sensibilidad a mayúsculas

Tanto GLIMPSE como WAIS soportan expresiones booleanas, es decir, construcciones de frases con los operadores lógicos AND (;), OR(,) y NOT (~), este operador es soportado solo por la version 4.0 de GLIMPSE). Por otro lado, solo GLIMPSE permite hacer la busqueda de una palabra **a** pesar de que este

incompleta, y puede controlar la sensibilidad a mayusculas (que permite hacer una busqueda respetando las mayúsculas de las palabras o simplemente ignorandolas). WAIS no posee estas caracteristicas.

Facilidad de creación del índice de busqueda y control de lo que esta siendo indexado

GLIMPSE posee una herramienta llamada glimpsindex que permite con facilidad indexar un conjunto de archivos. WAIS carece de este utilitario, pero hay terceras personas que han desarrollado un script para realizar esta tarea.

Por otro lado, GLIMPSE puede incluir o excluir en su indice los archivos que se deseen ingresar o excluir de la busqueda. WAIS necesita de un script adicional para realizar esta tarea.

De todo este análisis podemos observar que GLIMPSE presenta un mejor desempeño en lo referente a la busqueda de información que freeWAIS. Una ventaja de freeWAIS es que soporta el protocolo WAIS para la conexión con otras maquinas de busqueda. Pero en la ESPOL, debido a las necesidades de una maquina de busqueda eficiente en cuanto a recursos consumidos y busqueda de inforrnacion (debido a lo limitado de los recursos existentes), se optó por la alternativa presentada por GLIMPSE, puesto que no fue considerado un factor crucial el hecho de que no soportase el estandar 239.50.

Detalles de GLIMPSE

GLIMPSE es una herramienta desarrollada en la Universidad de Arizona **(http://glimpse.cs.arizona.edu)** por Udi Manber. Está conformada por los siguientes programas:

1. Una maquina de busqueda basada en grep (llamada agrep)

Esta herramienta es la que permite buscar la información en la estructura de archivos indexados. Puede buscar por palabras incompletas, soporta sensitividad a mayusculas, "mejor coincidencia de la palabra" y expresiones booleanas **(AND,** OR, NOT)

2. Un indexador (llamado glimpseindex)

Es el programa que atraviesa una estructura de directorios e indexa los archivos tipo texto. Existen 3 opciones de indexamiento: La enana **(2-3%** del total de archivos), la pequeña **(7-8%) y** la mediana (20-30%). Mientras mas grande es el índice, mas rapida es la busqueda.

3. El programa de busqueda (llamado glimpse)

Hace las veces que agrep, pero este busca sobre el archivo indice. Soporta las mismas caracteristicas de agrep.

4. Un manejador de archivos para sitios HTTP (amgr) y

Permite definir y controlar multiples colecciones de archivos HTML.

5. Un script CGI para conducir las busquedas a traves de FORMAS HTTP. (llamado aglimpse)

Este es el programa CGI que permite recoger los datos del usuario, luego ejecuta el programa "glimpse", formatea el resultado en HTML, **y** los pasa al servidor web para mostrarlos al browser del usuario.

La forma en que trabaja GLIMPSE en la ESPOL, es la siguiente:

El **indice es generado por secciones** (Sección ESPOL, Sección Informativo Politécnico, ntre otras) y cada sección **ocupa un subdirectorio dentro del S.O.** (este es un requisito undamental), de *tal manera que* este *aislada del resto* de *secciones. El* indice *es* enerado y guardado en un directorio de GLIMPSE (/home/httpd/glimpse). GLIMPSE rdena la información de sus indices, por palabras mas comunes (llamadas keywords).

Al hacer una busqueda, GLIMPSE revisa estas keywords, que apuntan hacia archivos en el servidor web de la ESPOL. De esta manera, GLIMPSE llega rapidamente a la información sin necesidad de recorrer todos los directorios del servidor web. Además GLIMPSE genera un archivo indice muy pequeño (ocupa poco espacio en disco duro) lo cual lo hace ideal para un ambiente como el de la ESPOL (en donde se debe tender a la optimización de recursos, por ser estos limitados).

Si se desea indexar una nueva seccion dentro del servidor web de la ESPOL, para que esta posea su propia maquina de busqueda, se debe realizar lo siguiente:

a. Crear un directorio donde se almacenara el indice generado, con el nombre de la nueva seccion a ser indexada (crearlo en el directorio /home/httpd/glimpse). Por ejemplo: si la seccion nueva se encuentra es el "Manual de Internet", crear el directorio /home/httpd/glimpse/MANUAL_INTERNET.

b. Ejecutar el programa "glimpseindex" (Archivo /cgi-bin/glimpse-3.0/glimpseindex del web de la ESPOL) para generar el indice de la nueva sección. Se debe especificar en el comando, el subdirectorio que va a ser indexado. (ver ejemplos que se encuentran en este directorio)

c. Generar un archivo de configuración para el nuevo índice llamado amgrxx.cfg, donde **XX** es un numero secuencial que se va colocando a medida que van surgiendo nuevas secciones en el servidor web de la ESPOL (ver el directorio /home/httpd/wwwlib para confirmar cual es el ultimo dígito de los archivos amgrxx.cfg). Este archivo contiene lo siguiente:

```
#sección indexada     #version GLIMPSE    #numero     #constantes
/home/httpd/glimpse/CATALOGO GlimpseHTTP Demo Archive     xx          yes    no
```

Figura 6.22.
Archivo de configuración para el indexmiento de una sección del **web** de la ESPOL con GLIMPSE
Fuente: Archivo **/home/httpd/wwwlib/amgr02.cfg** de la ESPOL (goliat.espol.edu.ec)

l. Crear una FORMA para que se pueda realizar la búsqueda a traves del servidor web le la ESPOL. El codigo en HTML se muestra a continuación:

```
<FORM ACTION="/cgi-bin/aglimpse/xx"> Ejecución del CGI
<font size=6>B</font>uscar tema relacionado a la ESPOL:
<INPUT type=text size=50 NAME="query">  <p>
:INPUT NAME="case" TYPE="checkbox" CHECKED>Insensible a
may&uacute;sculas
<p>
:/form>
```

Figura 6.23.
Código HTML de una forma para busqueda a través de GLIMPSE
Fuente: Archrvo /CATALOGO/bien.html del web de la ESPOL

6.4. Servidor Proxy Apache de la ESPOL

El programa servidor web instalado en la ESPOL se conoce con el nombre de Proxy Apache (para mayor información consultar http://www.apache.org). La version instalada es la 1.1.1. Se hicieron pruebas con tres de los mejores programas servidores web de libre distribución en la Internet, para plataforma Unix (Solaris 2.4). El cuadro que se muestra a continuación detalla los parametros que se tomaron en cuenta al momento de escoger el servidor web, y justifica el hecho de haber escogido al servidor Proxy Apache como la mejor opción.

Servidor	Prefork	Soporte para Proxy	Actualizable por módulos	SSL	SSI	VirtualHosting
Apache 1.1.1	**Si**	**Si**	**Si**	**Si**	**Si**	**Si**
NCSA 1.5.1	**Si**	No	No	No	**Si**	**Si**
CERN 3.0	**NO**	**Si**	No	No	No	No

Tabla IX
Parámetros considerados en la elección del programa servidor web de la ESPOL

. **Prefork**

Esta es la caracteristica por la que un programa servidor web separa (apenas se inicia) un pool de servidores" para manejar los requerimientos de los usuarios finales (ver sección **4).** De los servidores estudiados, solo Apache 1.1.1 **y NCSA** 1.5.1 soportan esta caracteristica.

b. **Soporte para Proxy**

Solo Apache 1.1.1 y CERN 3.0 soportan el almacenamiento temporal de páginas web (a traves del servidor Proxy). Esta caracteristica, junto con el prefork de procesos, convierten al Apache 1.1.1 en uno de los servidores con tiempo de respuesta mas rapido. El servidor proxy Apache puede manejar los siguientes protocolos: FTP, CONNECT (usado para los SSL) y HTTP/1.0 (cache de hojas de hipertexto). Corre en el mismo puerto que el Servidor Apache (80 por default).

c. **Actualizable por módulos**

Apache 1.1.1 esta diseñado de tal manera, que si se desea extender las capacidades del programa servidor web, se lo puede hacer a traves de modulos desarrollados por los creadores del servidor Apache o por terceras personas. Como ejemplo, estan los modulos de: Autenticacion con Kerberos (esquema de validación de usuarios); FastCGI (modulo que permite al servidor Apache actuar como un interpretador de Perl, haciendo que los programas Perl se ejecuten mas rapidamente), entre otros.

d. **SSL (Secure Socket Layer)**

El esquema de encriptamiento conocido como Secure Socket Layer (ver sección 2.5.1), es considerado por el Gobierno de los Estados Unidos como “Armamento” y ha sido restringida su exportación (Reglamento ITAR -International Traffic in Arms

b. **Soporte para Proxy**

Solo Apache 1.1.1y CERN 3.0 soportan el almacenamiento temporal de páginas web (a traves del servidor Proxy). Esta caracteristica, junto con el prefork de procesos, convierten al Apache 1.1.1 en uno de los servidores con tiempo de respuesta mas rapido. El servidor proxy Apache puede manejar los siguientes protocolos: FTP, CONNECT (usado para los SSL) y HTTP/1.0 (cache de hojas de hipertexto). Corre en el mismo puerto que el Servidor Apache (80 por default).

c. **Actualizable por módulos**

Apache 1.1.1 esta diseñado de tal manera, que si se desea extender las capacidades del programa servidor web, se lo puede hacer a traves de modulos desarrollados por los creadores del servidor Apache o por terceras personas. Como ejemplo, estan los modulos de: Autenticacion con Kerberos (esquema de validación de usuarios); FastCGI (modulo que permite al servidor Apache actuar como un interpretador de Perl, haciendo que los programas Perl se ejecuten mas rapidamente), entre otros.

d. **SSL (Secure Socket Layer)**

El esquema de encriptamiento conocido como Secure Socket Layer (ver sección 2.5.1), es considerado por el Gobierno de los Estados Unidos como "Armamento" y ha sido restringida su exportación (Reglamento ITAR -International Traffic in Arms

Regulations) fuera de este pais. A pesar de esto, el equipo desarrollador de Apache (http://www.apache.org), que nacio en los EEUU, ha sido el unico hacer un esfuerzo paralelo de desarrollo del servidor, fuera de dicho país (específicamente en el Reino Unido -http://stronghold.ukweb.com), con lo cual se ha podido realizar la distribución del servidor Apache-SSL sin ninguna restricción.

e. **SSI (Directivas Server Side Includes)**

Estas directivas, vistas en la sección 3.6.1, permiten incluir los valores de variables de ambiente y la salida de la ejecucion de comandos en una hoja de hipertexto. Apache 1.1.1 y NCSA 1.5.1 soportan esta caracteristica. Apache va aun mas alla y ha desarrollado un modulo que permite la ejecucion de un mayor numero de directivas SSI. (Llamadas XSSI: extended **SSI)**

f. **VirtualHosting**

Esta es una caracteristica que no ha sido mencionada anteriormente, pero que es importante nombrarla. VirtualHosting es la capacidad que tiene un servidor web de poder servir requerimientos de los browsers, bajo distintos nombres de servidor (nombres que deberan existir en la tabla de DNS). Esta es una caracteristica muy importante si se desea que en una misma maquina residan varios servidores web, con distintos nombres (reiteramos: los nombres de servidor deben existir en las tablas de DNS), y que sirvan

hojas de hipertexto de diferentes directorios. Por ejemplo: El nombre del servidor web de la ESPOL es **www.espol.edu.ec.** Si utilizamos VirtualHosting para servir, digamos, las hojas de la FIM (Facultad de Ingenieria Mecanica), bajo el nombre de **fim.espol.edu.ec,** podremos colocar las hojas de hipertexto de tal Facultad en directorios separados e independientes del servidor www.espol.edu.ec dentro de la misma maquina (en la actualidad, Sun Sparc 20). Existe una relación uno a uno entre un noinbre de servidor y su direccion IP (debido a que así lo especifican las reglas de DNS, a excepción de los nombres de servidores aliases, que no son tomados en consideración). Es por esto que por cada VirtualHost, debe existir una direccion IP. El numero de direcciones IP soportados por un Sistema Operativo es variable. La siguiente Tabla, muestra tres Sistemas Operativos populares en la Internet **y** sus respectivas capacidades de direcciones IP.

Sistema Operativo	**Direcciones IP Soportadas por el S.O. en una misma maquina**
Solaris **2.4** o superior	**255** direcciones IP
SCO Unix 5.0	14 o 15 direcciones IP tiene problemas de configuración
Windows NT 3.51 o superior	16 direcciones IP

Tabla **X**
Direcciones IP soportadas por el S.O. en una misma maquina

6.5. Promoción del servidor web de la ESPOL

Promocionar el servidor web de la ESPOL es una tarea que ha necesitado planificacion, la busqueda de los newsgroups idoneos para su prornocion y el desarrollo de nuevas herramientas que contribuyan a los fines de esta etapa del desarrollo de un servidor web.

Como ya se ha definido en la etapa de planificacion del servidor web, el público objetivo lo constituyen los "Miembros de la ESPOL, aspirantes a miembros y visitantes, interesados en conocer nuestra Universidad y el mundo del WWW". Se implementaron diferentes estrategias para alcanzar a dicho publico:

1. Se implementó un programa para envio de email a todos los usuarios de la ESPOL, por parte del administrador del servidor web, de tal manera de anunciar los cambios realizados al servidor web o algún tema de interes general. (para mayor información, consultar sección 6.2.5)

2. Se busco entre los newsgroups, cuáles eran los grupos idoneos para prornocionar el servidor web de la ESPOL, y se encontro los siguientes:

```
comp.infosystems.announce
comp.internet.net-happenings
```

Que son newsgroups en los cuales se puede promocionar la existencia de nuevos servidores web, de toda índole y en los cuales se promociono el servidor web de la ESPOL.

3. Se dio cabida para que los miembros y no miembros de la ESPOL pudieran accesar a servicios muy útiles para la diseminacion de información como es el Directorio de Homepages y Los Avisos Clasificados. El objetivo es promover al servidor web de la ESPOL, atrayendo visitantes a estas secciones del servidor.

6.6. Innovación en el servidor web de la ESPOL

1. Mantener una constante actualización del servidor de web

Gracias a las nuevas tecnologias emergentes, como por ejemplo, Java (lenguaje de programacion de Sun Microsystems), ActiveX (lenguaje de programacion de Microsoft), VRML (Virtual Reality Modeling Language), JDBC (Java Data Base Connectivity), entre otras, el web deberá necesariamente evolucionar. El presente trabajo ha dado pautas para que se empiece toda una era de desarrollo de webs en la ESPOL. Ha popularizado el uso de HTML (Hojas Personales HTML), ha integrado los servicios de Internet (TCP/IP) como el correo electronico (SMTP); ha integrado el servidor web a las actividades de sus miembros (avisos clasificados de interes); en fin, ha dado un conjunto de ideas que despertaran el deseo de la comunidad politecnica por participar en el desarrollo del servidor de web de la ESPOL.

2. Monitorear los requerimientos y necesidudes del usuario final

Es de gran utilidad recoger (a traves de herramientas como las estadisticas, los Guestbook y el email al administrador del servidor), las opiniones de los usuarios finales. En el servidor web de la ESPOL esto ha servido para realizar algunas mejoras como:

La utilización de máquinas de busquedas por secciones en el servidor web: debido a la necesidad de encontrar la informacion en forma rapida y precisa, sin necesidad de navegar por todo el servidor, se implementaron dichas maquinas en secciones como: El mapa sensitivo de la ESPOL, mapa sensitivo del Ecuador y el Informativo Politecnico. Los datos de las estadisticas nos dijeron que estas son las secciones mas visitadas del servidor (ver **http://www.espol.edu.ec/wusage3.2/usage)**; además, los email enviados al administrador nos sugerian colocar tales mecanismos de busqueda.

Estructuracibn de la informacibn: Se hizo un estudio de la mejor manera de distribuir la informacibn en base al análisis de algunas Universidades Norteamericanas (ver sección **5.5.3.2)** y a la retroalimentación de los usuarios, lo que dio origen a las hojas que actualmente componen el servidor web.

Interfaz unificada de administracibn: Al administrador puede considerarselo como un usuario mas del servidor. Es por esto que, en base a sus necesidades se creo dicha interfaz unificada que le permitiese dar mantenimiento con facilidad a los diferentes parametros del servidor web y los programas CGI que corren en él (ver figura **6.24).**

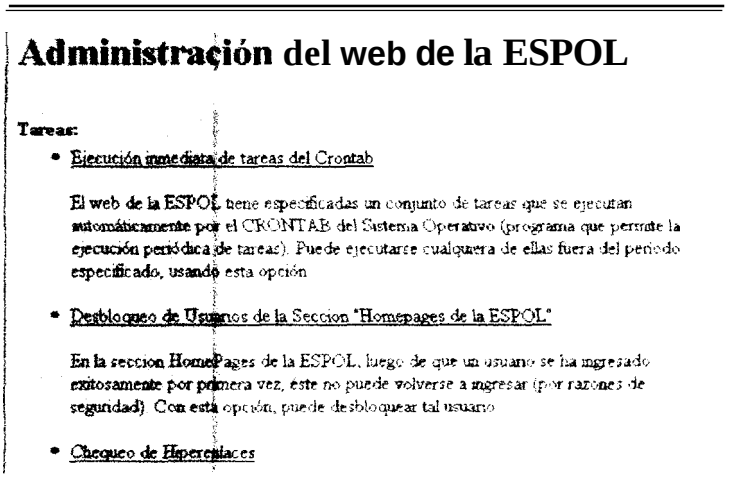


Figura 6.24.
Página de administración del web de la ESPOL
Fuente: http://www.espol.edu.ec

3. Mejorar continuamente la calidad del sewidor web

La tarea del servidor web de la ESPOL aún no esta terminada. El trabajo que se haga sobre el depende en gran medida de las necesidades del usuario final de la ESPOL, que iran evolucionando a medida que pase el tiempo. El usuario final cada vez pedira mas, a medida que se vayan ofreciendo mas y mas servicios. Es recomendable designar a un grupo de personas para la investigación de nuevas tecnologias en Internet, con el objeto de adaptar estas tecnologias al desarrollo de nuevas aplicaciones para webs que satisfagan la demanda de los usuarios.

CONCLUSIONES Y RECOMENDACIONES

En base a lo estudiado y a los objetivos planteados inicialmente, se pueden obtener las siguientes conclusiones y recomendaciones:

Conclusiones

1.Existen dos falencias del protocolo HTTP 1.0 que deben ser superadas en futuras versiones del mismo. La primera es que HTTP 1.0 es un protocolo de comunicacion entre cliente y servidor de web que permite realizar un unico requerimiento (por parte del cliente), por cada conexión y no multiples requerimientos como seria lo deseado, para no congestionar al servidor. Y segundo, La información **que** fluye entre cliente/servidor es de tipo texto (ISO 8859-1), lo cual lo hace un mecanismo muy inseguro de envio de datos.

2. Las FORMAS hechas en lenguaje hipertexto, no ofrecen mecanismos de validación de campos, chequeo de direcciones email o de URLs. Estas verificaciones deben ser

implementadas por el programador. Para realizar esto, se pueden emplear las siguientes tecnicas, sugeridas en la presente Tesis:

a. Validar los datos ingresados a traves del programa CGI.

b. Validar la direccion electrónica de un usuario utilizando el artificio de enviar un correo al usuario y chequear que el correo no regresó de vuelta (lo cual implica que el usuario no existe o que su host es inalcanzable).

c. Chequear los URLs (enlace), utilizando mecanismos de conexion Cliente/Servidor como son los sockets, utilizando el protocolo TCP y enviando comandos compatibles con el protocolo HTTP/1.0, tal como se especifica en el RFC del mismo.

3. Los servidores Proxy mejoran el tiempo de respuesta de una conexion, visto desde el punto de vista de los usuarios o clientes del servidor web. Pero además tienen otra ventaja que debe ser explota y es que evitan gastar una dirección IP por cada cliente web. Esto se deduce del hecho de que el servidor Proxy redirige todo requerimiento hecho por un cliente web (al servidor web) y es el servidor Proxy quien establece la conexion TCP (conexion proxy/servidor) y recibe los datos del servidor que a su vez los redirige hacia el cliente. De esta manera ya no es necesaria una conexion directa cliente/servidor en la que si se necesita una direccion IP para cada una de las partes, sino que el servidor Proxy actuara como intermediario entre cliente y servidor de web.

4. Perl es una herramienta ideal para el manejo de informacion textual, lo cual es una gran ventaja si hablamos del protocolo HTTP 1.0, el cual se basa en el envio y recepción de este tipo de inforrnacion. Existen otras ventajas que hacen de Perl un lenguaje ideal para el desarrollo de aplicaciones para servidores web, como son: ser un lenguaje sencillo de aprender con sintaxis parecida a C, estar optimizado para el manejo de archivos, bases de datos y busqueda de informacion, su actualización de versiones es constante y soporta mecanismos de comunicacion cliente/servidor como son los sockets.

5. Al momento de desarrollar aplicaciones para servidores web, es importante conocer las herramientas que ofrece el Sistema Operativo en el cual se desarrolla la aplicacion. De esta manera podremos utilizar aquellas herramientas útiles para nuestros fines. Un ejemplo de esto es el programa conocido como Crontab, aplicacion que ha facilitado enonnemente la implementación de los CGI que necesitan correr cada cierto periodo.

6. Debido al ambiente multiusuario en el que corren las aplicaciones CGI y que, en el caso de Perl no existen mecanismos propios del lenguaje para manejar tal ambiente, debe utilizarse alguna tecnica para controlar dicho ambiente. Una de las mas utilizadas es el uso de “archivos de bloqueo”, el cual fue implementado con exito en la ESPOL.

Esto evita que los recursos criticos (tales como archivos modificables o programas) a los que accesan los usuarios, se corrompan o se pierdan cuando multiples clientes tratan de accesarlos y/o modificarlos.

7. Al momento de escoger un buen programa servidor web, tal como el Apache server implementado en la ESPOL, se debe tomar en cuenta lo siguiente:

a. La forma en que fue diseñado: El programa servidor debe soportar preforking, de tal manera que los procesos en el servidor hayan sido creados antes de que un usuario resuelva abrir una conexion. Esto mejorara el tiempo de respuesta del servidor.

b. Su modularidad: modular quiere decir que pueda ser actualizable a traves de la adicion y compilacion de un conjunto de librerias o modulos que adicionen una nueva funcionalidad al programa servidor web.

c. Ofrecer servicios extra: Debe presentar servicios adicionales a los programas servidores web tradicionales, tales como: incorporar un servidor proxy y permitir mecanismos de seguridad como el Htpasswd de **NCSA.**

d. El protocolo o especificacion HTTP soportado. Esto es una caracteristica muy importante ya que, mientras mas actualizada es la especificacion, existiran optimizaciones al rendimiento del servidor que no pueden ser despreciadas.

8. Debido a que la transmision de información entre el browser y el servidor de web se realiza de manera textual, es necesario complementar a los servidores web con mecanismos de seguridad que permitan tener confidencialidad con la transmision de los datos que asi lo requieran. Uno de estos mecanismos de seguridad son los SSL o Secure Socket Layer.

9. Existen herramientas ya desarrolladas para servidores web, de libre distribución en la Internet, por lo cual, se necesita realizar siempre una investigacion previa a la implementación que nos permita conocer si lo que planeamos hacer, ya esta hecho. El llegar a conocer cuales son las mejores herramientas que pueden ser implementadas, siempre se deberá a una investigacion previa de tales herramientas, en sitios de webs, forums de discusion, entre otras fuentes. Tal es el caso de los programas de estadisticas de acceso, maquinas de búsqueda, entre otros.

Recomendaciones

1. Se recomienda la utilización de encriptamiento de la informacion (a traves de los SSL), solo en casos en los que sea necesaria (por ejemplo, en el envio de claves, informacion confidencial, etc), puesto que los SSL añaden una capa de software mas a los servidores web, disminuyendo el tiempo de respuesta de los mismos.

2. El programa lector de correos via web implementado en la ESPOL, soporta la lectura, envio y recepción del correo electronico. Puede mejorarse añadiéndole caracteristicas como: contestación a mensajes con copia del texto original (Reply), redirección de los mensajes (Forward), implementación de un esquema de seguridad adicional al que ya tiene (SSL) conocido como htpasswd, soporte de envio de archivos (attachments), entre otras mejoras, para de esta manera convertirlo en un lector de correos completo.

3. Se recomienda siempre validar las entradas de los usuarios, a traves de los mecanismos descritos en esta tesis, pues, de otra manera, se expone al servidor web a entradas indebidas o malintencionadas capaces de perjudicar al sistema.

4. Se recomienda adoptar los procesos de planificacion, analisis, diseño, implementación, promoción e innovación aqui planteados, de tal manera que el trabajo sea realizado por un equipo humano en el que se tengan diversos criterios que enriquezcan el desarrollo del servidor web.

5. Se recomienda re-utilizar las herramientas creadas e instaladas para implementar futuros programas CGI. Por ejemplo:

a. Se puede reutilizar el programa que convierte el texto a hipertexto (hipermail), para hacer publicaciones de cursos, conferencias, eventos deportivos, etc. utilizando un solo usuario al que se le envíe el correo electrónico de tales eventos y mostrando el contenido del mailbox de tal usuario.

b. Hipermail tambien puede servir para la lectura de un mailing list o lista de discusion a traves del servidor web, pues, este programa leera el contenido del mailbox del administrador de la lista.

c. La maquina de busqueda GLIMPSE puede servir para indexar cualquier seccion o "paquete informacional" nuevo que se agregue al servidor web de la ESPOL

6. Tomando en consideración las ideas que se exponen en **el** capitulo VI y en especial la referente a la validación de URL en el "Directorio de Homepages" (ver seccion 6.2.1),

podemos hacer un programa que se conecte al puerto 80 de un programa servidor web. Luego, enviamos requerimientos tipo HEAD (comando que forma parte de la especificacion HTTP/1.0), al programa servidor web tal cómo se muestra en la figura **6.4.** Finalmente, variamos la carga de acuerdo a lo que necesitemos (habria **que** implementar esta parte del codigo). Con este programa hecho, probariamos los diferentes servidores web que tenemos a nuestra disposición: Apache httpd, **NCSA** httpd, CERN httpd, etc. todos instalados en una misma maquina (por ejemplo la Sparc 20; 192.188.59.2).La prueba debe hacerse desde una estacion (podriamos utilizar la Sparc 2 para el efecto). Entonces, enviariamos los requerimientos HTTP desde la estacion (Sparc2) hasta el servidor (Sparc 20) donde residirian los programas servidores web. Habria que considerar el realizar la prueba en horas con poco trafico de red, de tal manera que tal trafico afecte en minimo grado la prueba.

APENDICES

APÉNDICE A

Glosario de Términos

ASCII: American Standard Code for Information Interchange. Codigo de caracter de **7** bits que puede representar 128 caracteres, algunos de los cuales son de control y no son imprimibles.

Ancla (anchor): El area de un documento hipertexto que es la fuente o el destino de un enlace de hipertexto. El enlace puede extenderse de esa area a otro documento o de otro documento a esa area. Cuando los anchors son los puntos de inicio de estos enlaces, estos son tipicamente resaltados o de otra manera identificados por el browser en hipertexto

Archie: Un sistema para indexar el contenido de los servidores de FTP

Browser: Software utilizado por el usuario final para navegar en el Web; sinonimo de "cliente del Web"

CERN: Centre Europeen pour la Recherche Nucleaire. El Centro Europeo para la Investigación Nuclear, donde el Web se originó en 1989. (Ver http://www.cern.ch/)

Competitividad: A nivel de servidores web, la competitividad es un factor que influye en el exito de un servidor, ya que exige el tomar en consideración el desarrollo de otros servidores web con el mismo proposito que el nuestro, de tal manera que nuestro servidor presente una mejor alternativa para atraer a los usuarios.

Consistencia: La consistencia es una cualidad que aplicada a los servidores web, determina que exista cierta relación con los elementos que conforman cada hoja de hipertexto (imagenes, posición del texto, sonidos, barras de ubicacion, etc) de tal manera que para el usuario resulta mas sencilla la navegacion.

CGI (Common Gateway Interface) Estandar para el desarrollo de programas que se comunican con el programa servidor web.

Cliente: Software que requiere la informacion o servicios de otro programa llamado servidor. El cliente muestra tal informacion en la forma requerida por su plataforma de hardware.

Domain Name: El nombre de un host según las tablas de DNS; Este nombre es mapeado a la direccion IP.

DNS: Servicio de Nombres de Dominio: Es una estructura en forma de arbol dispersa por todo el mundo en los llamados "servidores DNS" que permiten hacer la traduccion de un nombre de maquina a su direccion IP y viceversa.

FTP(File Transfer Protocol): Medio para intercambiar archivos a lo largo de la red.

FORMA : Elemento HTML que permite a los usuarios ingresar información a traves de hojas de hipertexto y enviarlas a una aplicación o programa CGI para su procesamiento.

FAQ: Frequently Asked Questions o preguntas mas comunes. Los **FAQ** recogen las preguntas mas comunes en un newsgroup, lista de discusion, servidor web, organización, etc. Son de gran ayuda para los novatos con poca experiencia en el tema.

GIF (Graphics Interchange Format) Formato de almacenamiento de imágenes; Este formato es soportado por los browsers graficos.

Gopher Un protocolo para diseminar informacion en la Internet usando un sistema de menues; Los items de dichos menues, pueden ser enlaces a otros documentos o servicios de informacion.

Grafico, browser:

Cliente del Web capaz de mostrar imagenes, tipos de fuentes, etc **y** que usualmente ofrece una operación basada en el ratón del computador

Helper Applications: Programas que permiten visualizar la información no soportada **por** un browser. Se definen a nivel del cliente o browser del Web.

HTML(HyperText Mark-up Language): Lenguaje usado para crear paginas de servidores web; Los browsers del Web muestran estas paginas de acuerdo nivel del lenguajes soportado: HTML 1.0, 2.0, 3.0, etc.

HTTP (HyperText Transfer Protocol) Protocolo nativo del Web, usado para transferir documentos de Hipertexto entre el cliente o browser **y** el programa servidor web.

Homepage: La pagina de entrada para accesar a un servidor web local; Tambien es una pagina que una persona define como su pagina principal y que a menudo contiene infonnacion personal o profesional.

HotJava: Browser del Web capaz de ejecutar applets escritos en el lenguaje de programacion Java

Hipermedia: Hipertexto que puede incluir multimedia: texto, gráficos, imagenes, sonido y video.

Hipertexto: Texto que no esta restringido a una pagina o conjunto de páginas, sino que contiene enlaces hacia otras fuentes de infonnacion. En el WWW, estas fuentes pueden estar en cualquier lugar del mundo, accesible via Internet.

ISO (International Standards Organization) Una organización internacional que establece los estandares para muchas cosas, incluyendo, por ejemplo, el conjunto de caracteres de ISO Latin 1 (ver http://www.iso.ch) usado en la comunicacion cliente/servidor del WWW.

Imagemap o Mapa Sensitivo: Un grafico dentro de una pagina HTML que puede potencialmente conectar cada pixel o region de una imagen a un recurso del Web; Las selecciones del usuario permiten accesar al recurso.

Internet: La coleccion de computadoras distribuidas globalmente, que intercambian información mediante el protocolo TCP/IP.

Java: Un lenguaje de programacion orientado a objetos para crear aplicaciones distribuidas y ejecutables.

Javascript: Lenguaje con ciertas semejanzas a Java, que no necesita de compilacion y puede ser incrustado en una hoja de hipertexto.

LAN Red de area local

Link: Conexión entre un documento de hipertexto y otro

Lynx Un browser en modo texto (no grafico), desarrollado por la Universidad de Kansas.

MIME: Multipurpose Internet Mail Extensions, una especificación para formato de documentos multimedia.

Mosaic: Un browser grafico originalmente desarrollado por el National Center of Supercomputing Applications (NCSA);

NCSA (National Center for Supercomputing Applications) Centro de Investigación en el que se han desarrollado el browser denominado Mosaic y programas servidores web (httpd NCSA).

Navegación: El acto de observar el contenido del Web por algun proposito

Paquete: Un conjunto de datos manejados como una unidad en la transmision de información.

Page: Se refiere a una hoja simple de hipertexto (un archivo simple de HTML)

Perl: (Practical Extraction and Reporting Language) Lenguaje escrito por Larry Wall usado comunmente para la manipulación de texto para escribir aplicaciones gateways para servidores web.

Programa Servidor web: Software que permite la instalacion de un sistema de infomacion basado en el WWW. Se lo denomina generalmente como httpd (HTTP daemon) y por lo general corre en ambientes multitarea como Unix, NT, etc. Ejemplo de tales programas son: Apache httpd, NCSA httpd, CERN httpd, entre otros.

RFC (Request for Comments): Una serie de documentos que describen estandares o nuevos estandares para los protocolos y tecnologias de Internet.

Robot: Software que automaticamente explora el Web por una variedad de propositos; los robots que coleccionan recursos para consultas a bases de datos posteriores por los usuarios, son llamados spiders.

SGML Standard Generalized Mark-Up Language; Estandar para definir lenguajes mark-up; HTML es un derivado de SGML (ver http://www.sgmlopen.org)

Servidor: Un programa que provee informacion o servicios basados en los requerimientos de los programas clientes.

Servidor web: Se refiere al sistema que provee la informacion de una organización, institución, Universidad, etc. al mundo del WWW. Hace referencia al conjunto de hojas de hipertexto que son servidas a traves de los programas servidores web. Ejemplo de esto es: www.espol.edu.ec (ESPOL), www.cnn.com (CNN), entre otros. Es sinonimo de web (en minusculas).

Site: Sección en la cual residen los documentos un servidor web (u otros documentos servidos en otro protocolo); por ejemplo, un Web site, un Gopher site, un FTP site.

Spider: Software que recorre el Web para coleccionar informacion sobre recursos para consultas posteriores hechas por usuarios buscando hallar recursos; Las principales especies activas de spiders incluyen Lycos y WebCrawler.

Tag o elemento HTML: Codigo usado para confeccionar parte de un elemento HTML; por ejemplo: el elemento TITLE tiene como tag de inicio: <TITLE> y como tag de fin: </TITLE>

Telnet: Un protocolo para compartir informacion a lo largo de redes usando tecnicas para emulacion de terminal; Simula que el usuario esta directamente conectado a un computador remoto.

URL (Uniform Resource Locator): El esquema de direccionamiento en el Web; Un URL identifica un recurso en la red. Por ejemplo: http://www.telebit.com es un URL.

Usenet: Un sistema para diseminar discusion información en la Internet; El espacio de discusion de Usenet esta dividido en newsgroups, cada uno de un topico o subtópico particular.

Usuario final: Es la persona o grupo de personas que utilizan el sistema desarrollado. En el Web, el usuario final es aquella persona que utiliza el browser para accesar a la información de los sewidores web.

Vbscripts: Lenguaje basado en Visual Basic, que se incrusta en el codigo HTML. Su autor es Microsoft.

web: Un conjunto de hojas de hipertexto que es considerado un trabajo simple; tipicamente, un web simple es creado por autores cooperativos o un autor e instalado en un sewidor simple con enlaces a otros servidores web. Es un subconjunto del Web y es un sinonimo de servidor web.

Web (World Wide Web): Es la gran telaraña de informacion que agrupa un conjunto de sistemas de informacion basados en HTML **y** utilizando caracteristicas de hipermedia, que se ha difundido **por** toda la Internet.

WWW El World Wide Web, sinonimo de Web

APÉNDICE B

Lenguaje HTML

En este resumen Usted encontrara los siguientes simbolos:

string	Cualquier conjunto de caracteres alfanumericos
URL	Uniform Resource Locator
""	Una serie de elementos
value 1|...	Una serie de posibles valores para atributos
N	Un numero positivo

Elementos de HTML Nivel 0

Estructura d	HTML y elementos	comentario
TAG (start...stop)	Atributos	Explicación
<HTML>..</HTML>		Identifica el archivo como HTML; solo HEAD, BODY y los elementos comentarios deben ir dentro de este tag.
<!--string -->		Comentarios pueden ser incluidos entre estos tags.

HEA		
TAG (inicio...fin)	Atributos	Explicación
<HEAD>...</HEAD>		Encierra un conjunto de información no ordenada descriptiva sobre e. documento. Los elementos dentro de HEAD incluyen TITLE, BASE, ISINDEX y NEXTII
<TITLE>...</TITLE>		String que indentifica e. contenido del documento;No puede contener anchors párrafos, o resaltadores Todo HTML debe tener este elemento.
<BASE>...</BASE>		Usada para grabar el URL de la version original del documento; util cuando el archivo fuente es transportado.
	-href=''URL	Define el URL base del documento.
<ISINDEX>		Marca el documento par? busqueda; el servidor en elque el documento está localizado debe tener una maquina de busqueda definida para soportar esta busqueda.
<LINK>		Jsado para definir una relación entre el documento y otros objetos del documento.
	iref=''URL''	Identifica al documento o parte del documento al cual se refiere el enlace
	Jame = ''rel lrev·'	Forma de nombrar al enlace como posible destino de otrc documento de hipertexto.
	Rev=''madel.	Similar a rel, pero el atributo rev indica la relacion inversa de Rel. Por ejemplo, el Link con Rel=''maed'' muestra que el atributo Href indica que el JRL dado en el Href es el autor del documento actual,

		Usando Rev=''made'' se indica que el documento actual es el autor del URI dado en el atributo Href.
LINK	Urn=''string''	Indica el Uniform Resource Name del documento; la especificación del URN y otros direccionamientos estar todavia en desarrollo.
	Title=''string''	Este atributo no es usado como un substituto del atributo TITLE del documento en si, sino como el titulo del documento dado por el atributo Href del elemento LINK. Este atributo es raramente usado o soportado por los browsers.
	Methods='	Describe los metodos HTTP que soporta el objeto referido por el Href de LINK. Por ejm. Un metodo es Busqueda; Un browser puede usar estos metodo para dar informacion al usuario acerca del documento definido por el elemento LINK.
<META>		Usado para identificar la meta-información (informacion acerca de la información) en el documento. Este elemento no toma ellugar de los elementos que tienen ya un propósito específico.
	Http-equiv -''...''	Este atributo conecta el elemento META a una respuesta de protocolo particular que es generada por el servidor HTTP que mantiene el documento.
	Name=''string''	Este atributo es el nombre para la informacion en el documento -No el titulo del documento. Es un meta nombre para clasificar esta informacion

	Content=''string''	Un meta-nombre para el contenido asociado con el nombre dado (definido por el atributo Name) o la respuesta definida por Http-equiv
NEXTID		Este elemento es usado por el software generador de textos en la creación de identificadores.
	N=''string''	usado para definir el siguiente identificador a ser separado por el programa generador de texto, Normalmente, los escritores humanos de HTML no usan este elemento, y los Web browsers lo icrnoran.

BODY y Elementos relacionados		
TAG (inicio...fin)	Atributos	Explicación
<BODY>...</BODY>		Delimitan el contenido de un documento HTML
<A>...</A>		El elemento anchor usado como la base para unir documentos.
	Href=''URL''	Este atributo identifica el URL de la referencia hipertexto para este anchor, en la forma Href=''URL'', donde el URL dado sera el recurso que **el** browser recupere cuando el usuario haga click en el hotspot del anchor.
	Name= ''string''	Este atributo crea un nombre para el anchor; este nombre puede entonces ser usado dentro del documento o fuera del documento en el anchor para referirse a la porción de texto identificada por el nombre.
	Title="string''	Este atributo es para el titulo del documento dado por el atributo Href del anchor. Un browser puede usar esta información para mostrar el

		titulo antes de recuperarlo c proveer un titulo para el documento Href cuando es recuperado. (por ejemplo, si el documento es un FTP site, este no tendra titulc definido)
	Rel=‘‘madel ’’	Define la relacion determinada desde el documento actual al objetivc (documento Href).
	Rev=‘‘madel... ’’	Define la relacion definida desde el objetivo (documento Href) al documento actual.
	Urn="string"	Esto indica el Uniform Resource Name del documento objetivo (Href); La aspecificacion URN esta todavia en desarrollo
	Methods="...’’	Provee informacion sobre las funciones que el usuario puede ejcutar en el objeto Href. Similar a la descripción para **el** atributo Title, esta informacion puede ser util para que el browser la muestre de antemano.

Bloque de caracteres: Elementos que unen texto en listas o bloques		
<PRE>...</PRE>		Configura un bloque de texto que sera presentadc en un font de tamaño fijo, con espacios sianificativos.
<BLOCKQUOTE>...</BLOCKQUOTE>		Encierra texto que es una anotacion de otra fuente.
<UL>...</UL> <OL>...</OL> <MENU>...</MENU> <DIR>...</DIR>		Lista informacion: todos usan el elemento LI para identificar sus elementos: 1.UL encierra una lista no ordenada de items. 2.OL encierra una lista ordenada de items

		3. MENU encierra una lista no ordenada de items. 4. DIR encierra una lista de items que son de maximo 20 caracteres.
	Zompact	Hace compacta la lista
<LI>		Identifica una lista de elementos en UL, OL, MENU,DIR
<DL>...</DL>		Una lista de definicion o glosario; usa DT para identificar terminos y DD para identificar definiciones.
	Compact	Hace la lista compacta
<DT>		Identifica terminos en la lista de definicion (DL)
<DD>		Identifica la descripción en la lista de definicion (DL)
<ADDRESS>..</ADDRESS>		Información de propiedad o autoría, tipicamente al final o al inicio del documento.

<H1>...</H1>		Encabezado Nivel 1
<H2>...</H2>		Encabezado Nivel 2
<H3>...</H3>		Encabezado Nivel 3
<H4>...</H4>		Encabezado Nivel 4
<H5>...</H5>		Encabezado Nivel 5
<H6>...</H6>		Encabezado Nivel 6
		s
<HR>		Divide las secciones de texto con una regla horizontal.
<P>		Identifica el inicio del parrafo. El tag de fin </P> es opcional.
 		Fuerza un quiebre de linea. Tipicamente, es usado para representar una dirección

Avisos clasificados

###
#Nombre del programa: compraventa.pl
#Ubicación: /usr/httpd/cgi-bin (máquina: 192.188.59.2)
#Autor: Servio Lima
#Lenguaje: Perl (Interpretador ver 4.0 o superior)
#Fecha Creación: 1/Mar/96
#Descripción:
Permite ingresar un aviso clasificado a la cola de avisos. Trabaja en conjunto con el programa mailrcpt.pl para publicar los avisos.
###

```
#Línea inicial de todo programa Perl
#!/usr/bin/perl

#Definición de variables
$anunc="/home/httpd/cgi-bin/anuncios";
$nombre_espol ="espol.edu.ec";

#Grupos de usuarios de la ESPOL, de acuerdo al archivo /etc/groups
$profesores = 100;
$estudiantes = 110;
$administrativos =120;

#Conversion de meses de número a letras.
%meses=("01","Enero","02","Febrero","03","Marzo","04","Abril","05","May
o",
        "06","Junio","07","Julio","08","Agosto","09","Septiembre",
        "10","Octubre","11","Noviembre","12","Diciembre");

#Obtención de datos enviados por el servidor web a traves del STDIN del
programa
read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});

# Descomponer los datos recibidos en pares nombre=valor
@pairs = split(/&/, $buffer);

#Almacenamiento de los pares nombre=valor en el arreglo FORM
foreach $pair (@pairs)
{
    ($name, $value) = split(/=/, $pair);

    # Convertir los + a espacio en blanco y los codigos hexadecimales a
      su caracter respectivo
    $value =~ tr/+/ /;
    $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
```

```
    #Almacenamiento de los pares nombre=valor en el arreglo FORM
    $FORM{$name} = $value;
}


#blank-response se ejecuta si los campos dados no han sido completados
&blank response unless $FORM{'nombre'};
&blank-response unless $FORM{'email'};
&blank-response unless $FORM{'ciudad'};
&blank-response uniess $FORM{'pais'};
&blank_response unless $FORM{'descripcion'};


#Verificar que el campo 'email' tenga un formato valido
#y separa el email en user y dominio
if ($FORM{'email'} =~ /@/&& $FORM{'email'} !- /;/){
($user,$domain) =split (/@/,$FORM{'email'});
}
else{
      &URL response3("Formato invalido de email");
      exitj
}

#Verificación de que el usuario pertenece a la ESPOL
if ($domain eq $nombre_espol){
#Uso de la funcion getpwnam que lee el archivo /etc/passwd
      ($dato)=(getpwnam($user))[3];
      if ($dato ne $profesores  && $dato ne $estudiantes && $dato ne
$administrativos) {
        &URL response3("Ud. no esta registrado en $FORM{'section'}");
        exitj
        }
      if ($dato ne $profesores && $FORM{'section'} eq "Profesores"){
      &URL response3("PROFESOR no registrado en $nombre_espol.");
      exitj
      }
      else{
      if($dato    ne    $administrativos    &&    $FORM{'section'}    eq
"Administrativos"){
      &URL_response3("ADMINISTRATIVO no registrado en $nombre_espol.");
      }
      if($dato eq $profesores && $FORM{'section'} ne "Profesores"){
      &URL response3("Ud. debe registrarse como Profesor");
      exitj
      }
   }
}
else{
      &URL response3("Ud. no est&aacute; registrado en la ESPOL");
      exitj
  }

($dato2)=(getpwnam($user))[0];
if ($user ne $dato2){
```

```
            &URL_response3("User no resgistrado en $nombre_espol");
            exit;

}

#Definición de constantes
$fecha =`date '+%m-%d-%y'`;
chop($fecha);
($mes, $dia,$ano) = split(/-/,$fecha);
$mes = $meses{$mes};
$nombre = $FORM{'articulos'};
$archivo ="/home/httpd/docs/CLASIFICADOS/COMPRAVENTA/${nombre}.html";
$archivo_url
="http://www.espol.edu.ec/CLASIFICADOS/COMPRAVENTA/${nombre}.html";

#Chequeo de que un usuario (email) no registre más de un anuncio por
dia
dbmopen(PANUNCIOS,"$anunc",0644);
      while(($key,$value) = each(GANUNCIOS)){
            if ($key eq $FORM{'email'} & $value eq $nombre){
                  &html_header("Solo se permite un anuncio por dia");
                  &html_trailer;
                  dbmclose(%ANUNCIOS);
                  exit;
                  }
      }
#En caso de que no se haya registrado antes, se lo registra en una base
de datos de Avisos Clasificados
 $ANUNCIOS{"$FORM{'email'}"} = $nombre;
 dbmclose(%ANUNCIOS);

$flag =0;
#Creación de un archivo que contenga únicamente el nuevo Aviso
#Clasificado en formato HTML
open(FILE, ">>$archivo");
if (-z $archivo){
        print FILE "<HTML>\n";
        print FILE "<HEAD>\n";
        print FILE "<TITLE>Compra/Venta de Hoy</TITLE>\n";
        print FILE "</HEAD>\n";
         print FILE "<BODY  BACKGROUND=\"/GRAFICOS/CATALOGO/espol6.gif\"
text=\"000000\" 1ink=\"0000ff\" alink=\"ff0000\" vlink=\"0000ff\">\n";
        print FILE "<H1>Compra/Venta de $FORM{'articulos'}</H1>\n";
        print FILE "<P>\n";
      $flag =l;
      }
      #Registro de los datos del usuario en formato HTML
      print FILE "<img
      src=\"/GRAFICOS/CATALOGO/blueball.gif\">$FORM{'nombre'} (email:
      <a href=\"mailto:$FORM{'email'}\">$FORM{'email'}</a>), de
      $FORM{'ciudad'}($FORM{'pais'}), con fecha: $dia-$mes-$ano, desea
      "A
      print  FILE "<font size=+1>\n";
```

```
#Colocación de las características del mensaje: BOLD,
#ITALIC o BLINK
if ($FORM{'propiedad'} eq "1") {
           print FILE "<b>\n";
           }
if ($FORM{'propiedad'} eq "2")     {
                        print FILE "<i>\n";
                        }
if ($FORM{'propiedad'} eq "3")     {
                        print FILE "<blink>\n";
                        }
#Definición del tipo de Aviso Clasificado: COMPRA,VENTA o
#ALQUILER
if ($FORM{'accion'} eq "1")  {
           print FILE "V E N D E R \n";
           } else {if($FORM{'accion'} eq "2")  {
                 print FILE "C O M P R A R\n ";
                 }else{
                 print FILE "A L Q U I L A R\n";
                 }
           }

#Fin de los elementos HTML BOLD, ITALIC, BLINK y FONT
if ($FORM{'propiedad'} eq "1")  {
        print FILE "</b>\n";
      }
if ($FORM{'propiedad'} eq "2") {
        print FILE "</i>\n";
      }
if ($FORM{'propiedad'} eq "3") {
        print FILE "</blink>\n";
      }
print FILE "</font>\n";

#Impresión de datos adicionales del Aviso Clasificado:
#Descripción, precio, estado del artículo, foto, dirección,
#teléfono
print FILE "<br>\n";

#Impresión del campo 'descripción'
print FILE "$FORM{'descripcion'}";

#Impresión del campo 'precio'
if ($FORM{'precio'} ne "")  {
print FILE "<br><b>Precio</b> $FORM{'precio'}<br>";
}

#Impresión del campo 'estado'
if ($FORM{'estado'} eq "1")  {
      print FILE "Articulo en buen estado<br>";
} else I if ($FORM{'estado'} eq "2") {
           print FILE "Articulo en estado aceptable<br>";
           }
           else {
```

```
                                print FILE "Articulo por reparar<br>";
                                }
                        }

                        #Impresión del campo 'foto'
                         if ($FORM{'foto'} ne "")  {
                         if ($FORM{'foto'} ne "http://")  {
                        print  FILE  "<dd>Vea  un  grafico  del  articulo:<a
href=\"$FORM{'foto'}\"><b>$FORM{'foto'}</b></a><br>\n";
                           i
                         }

                #Impresión del campo 'dirección'
                if ($FORM{'direccion'} ne "" | $FORM{telefono} ne "")  {
                        if ($FORM{'direccion'} ne "")  {
                        printf FILE"<b>Direccion:</b> $FORM{'direccion'}<br>";
                             }

                #Impresión del campo 'telefono'
                if ($FORM{'telefono'} ne "")  {
                        printf FILE "<b>telefono:</b> $FORM{'telefono'}<br>";
                        }
                printf  FILE "<br><p>";
        }
        else {
                printf  FILE "<br><p>";
                }

#Cierre de los elementos BODY y HTML del texto HTML
if ($flag){
      print FILE "</BODY>";
      print  FILE "</HTML>";
      }
close (FILE);
#Envío al servidor de la localización del archivo creado
      print "Location: $archivo_url\n\n";


#=============================================================================
# html_trailer termina el cuerpo de un documento HTML
#=============================================================================
sub html_trailer
{
        print"</BODY>\n";
        print"</HTML>\n";
}
#=============================================================================
#html_header empieza el cuerpo de un documento HTML
#=============================================================================
sub html_header
{
        $document-title = $_[0];
        print "Content-type: text/html\n\n";
        print "<HTML>\n";
```

```
        print "<HEAD>\n";
        print "<TITLE>$document_title</TITLE>\n";
        print "</HEAD>\n";
        print "<BODY BACKGROUND=\"/GRAFICOS/CATALOGO/espol6.gif\"
text=\"000000\" link=\"0000ff\" alink=\"ff0000\" vlink=\"0000ff\">\n";
        print "<H1>$document_title</H1>\n";
        print "<P>\n";
I

#=========================================================================
#blank-response envia mensajes de error en caso de campos incompletos
#=========================================================================
sub blank-response
{
    &html_header("Datos en blanco");
    print "<HR>";
    print "<blink><h3>Sus datos aparecen en blanco</h3></blink>";
    print "Por favor reintente o regrese a nuestro <i>HOMEPAGE</i><p>";
     print "<HR><A HREF = \"/ESPOL/guestbook.html\"><IMG SRC =
\"/GRAFICOS/CATALOGO/back.gif\">Regresar</IMG></A><HR>\n";
     print "<A HREF _ \"/index.html\"><IMG SRC =
\"/GRAFICOS/CATALOGO/home.gif\">
HOMEPAGE</IMG></A>\n";
    &html_trailer;
    exit;
}

#=========================================================================
#URL_response3 imprime el mensaje enviado en $status en formato HTML
#=========================================================================
sub URL_response3
{
    local($status) = @_;
    &html_header($status);
    print "<HR>";
    print "<blink><h3>$status </h3></blink>";
    print "Por favor reintente o regrese a nuestro <i>HOMEPAGE</i><p>";
         print "<HR><A HREF = \"/ESPOL/alumnos.html\"><IMG SRC =
\"/GRAFICOS/CATALOGO
/back.gif\">Regresar</IMG></A><HR>\n";
             print "<A HREF = \"/index.html\"><IMG SRC =
\"/GRAFICOS/CATALOGO/home.gif\">
HOMEPAGE</IMG></A>\n";
    &html_trailer;
    exit;
}
```

Recepcidn de Avisos Clasificados via email y de los mensajes rebotados del Guestbook

###
#Nombre del programa: mailrcpt.pl
#Ubicación: /usr/httpd/cgi-bin (máquina: 192.188.59.2)
#Autor: Servio Lima
#Lenguaje: Perl (Interpretador ver 4.0 o superior)
#Fecha Creación: 1/Dic/95
#Descripción:
Programa ejecutado al ingresar un email al usuario administrador del servidor web de la ESPOL (httpd). Su ejecucion se debe a que el nombre "mailrcpt.pl" esta especificado en el archivo **/usr/httpd/.forward** que es leido por el programa Servidor de email del Sistema Unix (Solaris 2.5) de la ESPOL.
Este programa tiene dos fines:
1. Recibir y detectar los mensajes que correspondan a una confirmación (OK) de la publicación de un aviso clasificado.
2. Recibir y detectar los mensajes que hayan rebotado (por diferentes motivos) del libro de Invitados o Guestbook.
###

```
#Linea inicial de todo programa Perl
#!/usr/bin/perl

#Definición de constantes
$f1ag=0;
$archivo="/home/httpd/cgi-bin/temp.txt";
$directorio="/home/httpd/docs/ESPOL/GUESTBOOK";
$lock2="/home/httpd/cgi-bin/.lock_mailrcpt";
$email="/var/mai1/httpd";
$correo="/home/httpd/cgi-bin/.correo";
@transaccion=("COMPRA","VENTA","ALQUILER");
%DIREC=(1,"VENTA",2,"COMPRA",3,"ALQUILER");

#Desde while(1) empieza la sección crítica del programa, la cuál
permitirá leer un email entrante a la vez, en caso de que dos o más
email lleguen al mismo tiempo.
while(1){
        if (-e $lock2){
        next;
        }
        else{
        open(LOCK,">$lock2");

        open(FILE,">$archivo");
```

```
        #El mensaje entrante del email es leido a traves de la entrada
              #estándar del programa.
        while(<STDIN>){
        print FILE $_;
        I
        close(FILE);

#Apertura del archivo que contiene el email para hacer una busqueda del
#contenido del mismo
open(FILE,"$archivo");
while(<FILE>){

        #Búsqueda de una confirmación de los avisos clasificados dentro
        #del cuerpo del email (busqueda del string OK)
        if(/^[oO][kK]/){
                $flag=1;
                last;
        I
      #Búsqueda del string MAILER-DAEMON (o mailer-daemon en
      #minúsculas)para reconocer si un email ha sido producto de un
      #rebote del libro de invitados o Guestbook.
      if(/[Mm][Aa][Ii][Ll] [Eel[Rr]-[Dd][Aa] [Eel[Mm][Oo][Nn]/){
            $flag=2;
      }
      #Búsqueda del nombre de la sección (dentro del libro de
      #invitados) a la que pertenece el email entrante.
      if(/^Seccion/ && $flag eq 2){
            $flag =3;
            last;
      }
}
close(FILE),

#Respaldo del archivo de email guardado en la variable $archivo
if(!$flag || $flag eq 2 || $flag eq 3){
       open(FILE,"$archivo");
       open(FILE2,">>$email");
       while(<FILE>){
                print FILE2 $_;
        }
}
close(FILE2);
close(FILE);
close(LOCK);
unlink($lock2);
last;
        }
I

#Llamado a la funcion 'parse' en caso de tratarse de una confirmacion
#de los Avisos Clasificados
if($flag eq 1){
      &parse();
```

```
      }
#Llamado de la funcion 'borrar' en caso de que el email entrante sea un
#email rebotado del Libro de invitados o Guestbook
if($flag eq 3){
&borrar();
}


#Subrutina que permite publicar un email entrante de los Avisos
#clasificados en la sección correspondiente en el servidor web de la
#ESPOL
sub parse
{
open(FILE,"$archivo");

#Extracción del campo Subject del email que contiene lineas de la
#forma: fecha-CV-tipo_transacción-#transacción-seccion
while(<FILE>){
      if(/^Subject:/){
      $dato= $_;
      last;}
     }
($dato2,$dato3,$dato4) = split(/ /,$dato);
if($dato4 eq ""){$dato4=$dato3;}


#Chequeo de que la confirmacion de un usuario solo pueda darse una vez
#Rechazo de cualquier otra confirmacion posterior.
dbmopen(%LINKS,"$correo",0644);
if($LINKS{"$dato4"} eq $dato4){
      dbmclose(%LINKS);
      exit:
  }
else{
      $LINKS{"$dato4"}= "$dato4"; #Linea critica se adiciona user
  }
dbmclose(BLINKS);
```

```
$dir="/home/httpd/docs/CLASIFICADOS/COMPRAVENTA";
$flag2=0;
$file ="/home/httpd/cgi-bin/directorio2";

#Obtención del listado de los Avisos Clasificados que se encuentran en
#la cola de espera
system("ls $dir >$file");
open(FILE3,$file);
while(<FILE3>){
```

```
        if($dato4 eq $_){
        $flag2=1;

        #Lectura de cada nombre de Aviso Clasificado encontrado en la
        #cola de avisos
        ($mes,$dia,$ano,$codigo,$accion,$numero,$tipo)=split(/-/,$_);
        last;
        }
}
if($flag2){
        #Adición del Aviso Clasificado en el servidor web de la ESPOL
        $pub="/home/httpd/docs/CLASIFICADOS/COMPRAVENTA/$dato4";
        chop($tipo);

        #Apertura del archivo que contiene el Aviso Clasificado
        open(FILE4,$pub);
        $html="/home/httpd/docs/CLASIFICADOS/$DIREC{$accion}/${tipo}.html
        ";

        #A partir de while(1) empieza la sección crítica que permite
        #bloquear el acceso al archivo del servidor web mientras se esta
        #adicionando el nuevo Aviso Clasificado
        $lock="/home/httpd/cgi-bin/.lock_$DIREC{$accion}_${tipo}";
        while (1){
          if (-e $lock){
          next;
          }
          else{
          open (LOCK,">$lock");

        #Apertura del archivo del servidor web para adicionar nuevo Aviso
        #Clasificado
        open(FILE5,">>$html");

        #Grabación del Aviso Clasificado en el servidor web de la ESPOL
        while(<FILE4>){
              print FILE5 $_;
        }
        close(FILE5);
        close(FILE4);
        close(LOCK);
        unlink($lock);
        last;
        }
      }
    }
}

#sub_borrar permite Borrar un mensaje de la cola del Guestbook (que
#guarda todos los mensajes sometidos), del que nos ha rebotado el email
#enviado automáticamente en el programa guestbook.pl
sub borrar
{
```

```
open(FILE,$archivo);
while(<FILE>){
        #Búsqueda del valor 'region' que corresponde a la region a la
        #que pertenece el email que ha rebotado.
        if(/^Seccion/){
        ($label1,$region)=split(/:/,$_);
        }
      #Búsqueda del valor 'email' que indica el email que ha sido el
      #causante del rebote
      if(/^Email/){
      ($label2,$carta)=split(/:/,$_);
      last;
      }
}
close(FILE);

#Obtención de la 'region' y 'nombre de usuario' del email que nos ha
#rebotado
($user,$domain)=split(/\@/,$carta);
chop($region);
chop($domain);
if($domain eq ""){
      chop($user);
      }
$region =~ s/ +//;
$user =~ s/ +//;

#Borrado del mensaje del Guestbook correspondiente
if (-e"$directorio/$region.$user.$domain"){
      unlink("$directorio/$region.$user.$domain");
}
}
```

Busqueda de Personas

##
#Nombre del programa: personas.c
#Ubicación: /usr/httpd/cgi-bin (maquina: 192.188.59.2)
#Autor: Servio Lima
#Lenguaje: C (compilador GNIJ gcc para Solaris 2.4)
#Fecha Creación: 1/Dic/95
#Descripción:
personas.c permite realizar la busqueda de usuarios por departamentos en la ESPOL, utilizando funciones propias de C, para abrir el archivo del sistema /etc/passwd.
##

```
#Librerías a ser usadas
#include<ctype.h>
#include<pwd.h>
#include<string.h>
#include <stdio.h>
#ifndef NO_STDLIB_H
#include <stdlib.h>
#else
char *getenv();
#endif

#Definición de constantes y estructuras
#define MAX_DEPT 45

#Definition de un arreglo de estructuras de departamentos
struct departamentos{
      int dept_number;
      struct personas *point;
      } departments[MAX_DEPT];

#Definición de un arreglo de punteros de personas
struct personas{
      char pw_name[30];
      char pw_gecos[501;
      struct personas *sig;
      };
static struct personas *temp[MAX DEPT];
struct personas *getnodo-personas();

struct entry{
    char *name;
    char *val;
};
char servidor[]={"www.espol.edu.ec\0"};
```

```
#Departamentos de la ESPOL
char *string[] ={
      "BIBLIOTECA\0",
      "Biblioteca\0",
      "CEAA\0",
      "CECYP\0",
      "CELEX\0",
      "CENAIM\0",
      "CESERCOMP\0",
      "CETE-FIM\0",
      "CICYT\0",
      "CONTABILIDAD\0",
      "CPS\0",
      "CRECE\0",
      "ESPAE\0",
      "ESPOL\0",
      "ESPOL/UNO\0",
      "FICT\0",
      "FIE\0",
      "FIM\0",
      "FIMCIM\0",
      "FIMCM\0",
      "FISICA\0",
      "HP\0",
      "HTTPD\0",
      "ICF\0",
      "ICHE\0",
      "ICM\0",
      "ICT\0",
      "IETEL-ESPOL\0",
      "MATEMATICAS\0",
      "OTROS\0",
      "PAE\0",
      "PLANIFICACION\0",
      "PROTAL\0",
      "PROTCOM\0",
      "PROTEL\0",
      "PROTEP\0",
      "PROTMEC\0",
      "Protmec\0",
      "QUIMICA\0",
      "RECTORADO\0",
      "RELACIONES EXTERNAS\0",
      "SECRETARIA\0",
      "SYSADMIN\0",
      "UP\0",
      "VYD\0",
      "\0"};


main()
{
    struct entry entries[MAX_DEPT];
```

```
    register int x,m=0;
    int cl=0;
    int dep=0;
    struct passwd *p;
      int flag=0;

#Obtención de los datos enviados por el programa servidor web con el
#método POST
    abrir_body();
    if(strcmp(getenv("REQUEST_METHOD"),"POST"))
      imprimir_error("Este script debe ser referenciado con el metodo
POST");

              if(strcmp(getenv("CONTENT_TYPE"),"application/x-www-form-
ur1encoded"))
      imprimir_error("Este script solo puede ser usado para decodificar
resultados de formas");

#Recepción de los datos a traves de la entrada estandar del programa
#STDIN
    cl = atoi(getenv("CONTENT_LENGTH"));

    for(x=0;cl && (!feof(stdin));x++) {
      m=x;
      entries[x].val = fmakeword(stdin,'&',&cl);
#plustospace convierte los signos + a espacios en blanco
      plustospace(entries[x].val);

#unescape convierte los valores hexadecimales a sus respectivos codigos
#alfabéticos
      unescape_url(entries[x].val);

#makeword separa los pares nombre=valor
      entries[x].name = makeword(entries[x].val,'=');

#1 << es una operacion binaria que sirve para hacer una operacion AND
#de los distintos departamentos escogidos de la lista
#Esto sirve cuando el usuario ha escogido más de un departamento para
#la busqueda.
      if(!strcmp(entries[x].name,"departamentos"))
            dep |= (1 << atoi(entries[x].val));
    }

      if(!dep) imprimir_error("No especifico ningun departamento");
      limpiar();

#Apertura del archivo /etc/passwd (setpwent) y busqueda de la cadena de
#caracteres solicitada (val), linea por linea (getpwent)
      for(x=0;x<=m;x++){
            setpwent();
            while((p=getpwent()) !=NULL){
                  if(buscar_string(string[atoi(entries[x].val)],p-
>pw_gecos))
```

```
#impresión de los resultados de la busqueda
                        imprimir(p);
                    }
                }

cerrar_body();
exit(0);
1

#cerrar_body permite cerrar el cuerpo de un documento HTML
cerrar_body()
{
printf("<A HREF = \"/ESPOL/personas.html\"><IMG SRC =
\"/GRAFICOS/CATALOGO/back.gif\">REGRESAR</IMG></A><HR>\n");
printf("<A HREF =\"/index.html\"><IMG SRC =
\"/GRAFICOS/CATALOGO/home.gif\">HOMEPAGE</IMG></A>\n");
printf("</body>\n");
printf("</html>\n");
}

#abrir_body permite abrir el cuerpo de un documento HTML
abrir_body()
{
printf("Content-type: text/html\n\n");
printf("<html>\n<head>\n<title>Busqueda de
Personas</title>\n</head>\n");
printf("<body>\n<h1>Busqueda de Personas</h1>\n");
printf("<IMG SRC = \"/GRAFICOS/CATALOGO/line.gif\" ALT=\"Grafico:Linea
dorada\" ></IMG><P>\n");
printf("<h3>Usuarios del Servidor %s</h3><p>",servidor);
printf("<h4>Ud. puede enviarles mail a las siguientes
personas:</h4><p>\n");
}

#imprimir_error permite imprimir un mensaje de error (razon) dado
imprimir_error(razon)
char *razon;
{

    printf("<TITLE>Busca Personas no inicializado</TITLE>\n");
    printf("<H1>Busca Personas no inicializado</H1>\n");
      printf("El Busca personas no pudo inicializarse debido a que:
<h4>%s</h4><P>\n",razon);
    cerrar_body();
    exit(1);
}

#buscar_string permite buscar una cadena de caracteres (pointer) dentro
de otra (cadena)
buscar_string(cadena,pointer)
char *cadena;
char *pointer;
```

```
{
int i=0,j=0;
#ingreso al lazo de busqueda
do{

#conversion a mayúsculas de todo caracter
cadena[i] = toupper(cadena[i]);
pointer[j] = toupper(pointer[j]);

#Comparación caracter por caracter de pointer y cadena
while(!strncmp(&cadena[i],&pointer[j],1) && strcmp(&cadena[i],"\0")){
      i++;
      j++;
cadena[i] = toupper(cadena[i]);
pointer[j] = toupper(pointer[j]);
      }
#Si no se encontro coincidencias, retornar
if(cadena[i]=='\0') break;
i=0;
j++;

#Salir cuando pointer sea igual a null
}while(pointer[j]!='\0');

#Retornar un valor booleano 1 o 0 de acuerdo a si se encontro
#coincidencias o no se encontro
return(cadena[i]=='\0' ? 1 : 0);
}

#almacenar permite guardar los miembros de un departamento en un
#arreglo de punteros a la información de tales miembros
almacenar(y,q)
int y;
struct passwd *q;
{

#Creation de un elemento más en la lista de personas
      if(departments[y].point==NULL){
            temp[y]=getnodo-personas();
            departments[y].point=temp[y];
            }
      else{
            temp[y]->sig=getnodo_personas();
            temp[y]=temp[y]->sig;
           }

#Copiar los datos temporales a sus lugares respectivos
      strcpy(temp[y]->pw_name,q->pw_name);
      strcpy(temp[y]->pw_gecos,q->pw_gecos);
      temp[y]->sig =NULL;

}
```

```
#Obtención de espacio de memoria dinámicamente para el almacenamiento
#de información acerca de las personas
struct personas *getnodo_personas0
{
return((struct personas *)malloc(sizeof(struct personas)));
}

#Limpieza del arreglo de departamentos
limpiar()
{
int i;
for (i=0;i<MAX_DEPT;i++){
      departments[i].point=NULL;
      temp[i]=NULL;
      }
}

#Impresión de los resultados de la búsqueda
imprimir(p)
struct passwd *p;
{
int i;

#Impresión del campo email y comentarios del usuario, sacados del
#archivo /etc/passwd
printf("<LI>Correo:                    <a                    href=\"/cgi-
bin/correo.p1?%s\"><b>2s</b></a><p>\n",p->pw_name,p->pw_name);
printf("<i>%s</i><p>\n",p->pw_gecos);

#Por cada usuario crear una forma que llame al programa datos.pl para
#obtener datos adicionales de dicho usuario (utilizando finger)
printf("<form      method=POST      action=\"/cgi-bin/datos.pl?%s\">",p-
>pw name);
printf("<input type=submit value=\"Datos Adicionales\"></form>");
printf("<hr>");
}
```

Busqueda de teléfonos

###
#Nombre del prograrna: telefonos.pl
#Ubicación: /usr/httpd/cgi-bin (maquina: 192.188.59.2)
#Autor: Servio Lima
#Lenguaje: Perl (interpretador ver 4.0 o superior)
#Fecha Creación: 1/Dic/96
#Descripción:
telefonos.pl permite buscar los numeros telefonicos en los archivos colocados por el administrador en el directorio /home/httpd/cgi-bin/telefonos. Permite realizar una busqueda AND de tres parametros: nombre del usuario, area y numero telefonico.
###

```
#Linea inicial de todo programa Perl
#!/usr/bin/perl

#Definición de constantes
$dir="/home/httpd/cgi-bin/telefonos";
$fi1e="/home/httpd/cgi-bin/.te1efonos";

#Lectura del directorio donde se almacenan los archivos que contienen
#la información de los números telefónicos,actualizada.
system("ls $dir >$file");

#Uso del metodo POST para la obtencion de los datos
$method = 'POST';
        #
        #chequear que el metodo POST es usado
        #
if ($ENV{'REQUEST_METHOD'} eq $method)
                {
                #
                #El metodo POST determina que obtengamos
                #la entrada de la forma a partir de stdin
                #
                read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
                #
                #Dividir los pares nombre-valor tomando como
                #referencia '&'
                #
                @pairs = split(/&/, $buffer);
                #
                #Ir a traves de los pares y determinar el
                #nombre y valor para cada variable.
                #
                foreach $pair (@pairs)
```

```
                {
                ($name, $value) = split(/=/, $pair);
                $value =- tr/+/ /;
            $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;

                $FORM{$name} = $value;
             }
  }

#Banderas necesarias para la busqueda
$band=0;
$flag1=1;
$flag2=1;
$flag3=1;
$flag1=0 unless $FORM{nombre};
$flag2=0 unless $FORM{area};
$flag3=0 unless $FORM{telefono};

#Se realiza una operación binaria entre las banderas para darle un
#valor a la busqueda
$search= $flag1 | $flag2 | $flag3;

#Conversión de las variables a mayúsculas
$nombre1= "\U$FORM{nombre}";
$area1= "\U$FORM{area}";
$telefono1= "\U$FORM{telefono}";

#Se ingresa si se han dado los datos para la busqueda
if($flag1 || $flag2 || $flag3){
if (-z $file){
        print "<h2>No existe la base de datos de Telefonos</h2>";
        }
else{


&html header("Resultados de la b&uacute;squeda");
open(FILE,$file);
      $flag=0;
while(<FILE>){
        $linea =$_;
        chop($linea);
        open(FILE3,"$dir/$linea");
       while(<FILE3>){

#A partir de la siguiente línea, se consideran todos los valores
#posibles de las banderas $flag1, $flag2, $flag3, y se imprimen los
#datos de la busqueda con imprimir_datos();
#Uso del operador // para realizar las busquedas respectivas
      if($flag1){
            if($flag2){
                  if($flag3){
                        #$band="111";
                        if(/$nombre1/ && /$area1/ && /$telefono1/){
                              $flag=1;
```

```
                        &imprimir_datos($_);
                    }

                else{
                    #$band="110";
                    if(/$nombre1/ && /$areal/){
                        $flag=1;
                        &imprimir_datos($_);
                        }
                    }
            }else{ if($flag3){
                        #$band="101";
                    if(/$nombre1/ && /$telefono1/){
                        $flag=1;
                        &imprimir_datos($_);
                    }
                    }
                    else{
                    #     #$band="100";
                    if(/$nombre1/){
                        $flag=1;
                        &imprimir_datos($_);
                        }
                    }
            I
}else{
                if($flag2){
                    if($flag3){
                        #$band="011";
                        if(/$areal/ && /$telefono1/){
                            $flag=1;
                            &imprimir_datos($_);
                        }

                        I
                        else{
                        #$band="010";
                        if(/$areal/){
                            $flag=1;
                            &imprimir_datos($_);
                        }
                    }
                }else{
                    if($flag3){
                        #$band="001";
                        if(/$telefono1/){
                        $flag=1;
                        &imprimir_datos($_);
                        }
                    }
                    else{
                        $band="000";
                        print "No se ingresaron datos<p>";
                    }
```

```
                    }
                }

    }#fin de while(file3)
  }#fin de while (file1)
      if (!$flag){
      print "<h1>No se encontraron registros coincidentes</h1><p>\n";
      }
 }#fin de else -z
}#fin de $flag1 | $flag2 | $flag3
else{
      &html_header("No se ha ingresado ning&uacute;n registro<p>");
}
&html_trailer();
exit;


sub imprimir_datos
{
#Impresión de los datos hallados
local($datos) = @_;
($nombre2,$area2,$telefono2)= split(/:/,$datos);
            print "<b>Nombre</b>: $nombre2\n<dd>";
               print "<b>Area</b>: $area2\n<dd>";
               print "<b>Tel&eacute;fono</b>: $telefono2\n<p>";

}

#html_header imprime la parte inicial de un documento html
sub html_header
{
        $document title = $_[0];
        print "Content-type: text/html\n\n";
        print "<HTML>\n";
        print "<HEAD>\n";
        print "<TITLE>$document_tit1e</TITLE>\n";
        print "</HEAD>\n";
        print "<BODY>\n";
        print "<H1>$document_tit1e</H1>\n";
        print "<hr>\n";
}

#html_trailer imprime la parte final de un documento HTML
sub html_trailer
{
        print"</BODY>\n";
        print"</HTML>\n";
}
```

Lectura de correo en hipertexto

###
#Nombre del programa: mail5.pl
#Ubicación: /usr/httpd/cgi-bin (máquina: 192.188.59.2)
#Autor: Servio Lima
#Lenguaje: Perl (Interpretador ver 4.0 o superior)
#Fecha Creación: 1/Oct/95
#Descripción:
mail5.pl permite leer un archivo texto que constituye el archivo mailbox de un usuario cualquiera, lo transfiere via FTP y lo convierte a hipertexto para ser presentado por pantalla.
###

```
#Linea inicial de todo programa en Perl
#!/usr/bin/perl

#Definición de constantes
$mail-root = "/home/httpd/docs/mail";
$contador = "/home/httpd/cgi-bin/.contador";
$correo = '/usr/bin/cat /var/mail';
$hyper = '/home/httpd/cgi-bin/hypermail/hypermail';
$espol = 'www.espol.edu.ec';
$webmaster = 'httpd@espol.edu.ec';
$secure = "/home/httpd/docs";
$htpasswd ="/home/httpd/cgi-bin/htpasswd";
$lock="/home/httpd/cgi-bin/.lock_mail";

#Utilización del metodo POST para recepción de la información desde el
#servidor web
$method = 'POST';
if ($ENV{'REQUEST_METHOD'} eq $method)
                {
            #
            #El metodo POST determina que obtengamos
            #la entrada de la forma a partir de stdin
            #
            read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
            #
            #Dividir los pares nombre-valor tomando como
            #referencia '&'
            #
            @pairs = split(/&/, $buffer);
            #
            #Ir a traves de los pares y determinar el
            #nombre y valor para cada variable
            #
```

```
        foreach $pair (@pairs)
            {
        ($name, $value) = split(/=/, $pair);
            $value =~ tr/+/ /;
       $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;

        $FORM{$name} = $value;
          }


    #blank-response es invocado si no se ha ingresado el login o la
    #clave del usuario
    &blank response unless $FORM{'login'};
    &blank_response unless $FORM{'clave'};

#Desde while(1) empieza la protección al recurso critico, que en este
#caso es la escritura del archivo .script que contendra comandos para
#realizar el FTP automático al directorio /var/spool donde se almacena
#el mailbox del usuario
while(1){
        if (-e $lock){
        next;
        }
        else{
        open(LOCK,">$lock");

open(ARCHIVO,">.script");
#flock(ARCHIVO,$exclusive_lock);
print ARCHIVO "open $espol\n";
print ARCHIVO "user $FORM{'login'} $FORM{'clave'}\n";
print ARCHIVO "cd /var/mail\n";
print ARCHIVO "prompt off\n";
print ARCHIVO "asc\n";
print ARCHIVO "get $FORM{'login'}\n";
print ARCHIVO "quit\n";
#flock(ARCHIVO,$unlock_lock);
close(ARCHIVO);

#Apertura del proceso FTP, que lee el archivo de comandos .script y lo
#ejecuta; cualquier mensaje de error resultante sera almacenado en
#output.txt
open (FTP, "|ftp -in < .script > output.txt") || die "Content-type:
text/html\n\nError al ejecutar ftp!\n";
close(FTP);

#Borrado del archivo .script, por razones de seguridad
unlink(".script");

#Asignación y actualización de un número secuencial a la variable
#$numero leido del archivo %CONTADOR. Este nlimero servira para nombrar
#al directorio donde se almacenará los archivos de hipertexto resultado
#de la conversion del mailbox del usuario a HTML
$numero = 0;
```

```
 if (-e "$contador.dir" || -e "$contador.pag"){
        dbmopen(%CONTADOR,Scontador,0600);
        $numero = $CONTADOR{"num"} + 1;
        delete $CONTADOR{"num"};
        $CONTADOR{"num"} = $numero;
        dbmclose(%CONTADOR);
 }
 else {
 dbmopen(BCONTADOR, "$contador",0600);
 $CONTADOR{"num"} = "1";
 $numero = "1";
 dbmclose(%CONTADOR);
}

#Definición del nombre del directorio a ser creado, donde se
#almacenarán los archivos de correo en HTML
$directorio = "${mail_root}${numero}";

   close(LOCK);
        unlink($lock);
        last;
        }
} #fin de la sección crítica

#Creación del directorio $directorio
mkdir("${mail_root}${numero}",0700)     ||     die     "Content-type:
text/html\n\n<h1>No se
 pudo crear mail$numero directory<h1>";

#Apertura del programa HIPERMAIL, que permitira convertir el email del
#usuario a hipertexto
open(MAIL, "|$hyper -x -m \"/home/httpd/cgi-bin/$FORM{'login'}\" -d
${directorio}");

#Negar permisos de escritura al directorio (6) o permisos de lectura de
cualquier usuario diferente al administrador del servidor web (00)
open(CHMOD,"|/usr/bin/chmod -R 600 $directorio/*\n");
close(CKMOD);
close(MAIL);

#Borrado del archivo de correo del usuario, transmitido via ftp
unlink("/home/httpd/cgi-bin/$FORM{'login'}");

#Envio del URL al servidor web para su publicación
print "Location: http://www.espol.edu.ec/mail${numero}/index.html\n\n";

}
else
{
      #En caso de que no se haya usado el método POST

      &html_header("Error en la forma de comentarios");
      print "<HR><P>\n";
      print "La Forma no fue procesada. Por favor envie";
```

```
      print "sus comentarios a $webmaster\n";
      &html_trailer;
}

#html_header coloca la parte inicial de un documento HTML
sub html_header
{
      $document_title = $_[0];
      print "Content-type: text/html\n\n";
      print "<HTML>\n";
      print "<HEAD>\n";
      print "<TITLE>$document_title</TITLE>\n";
      print "</HEAD>\n";
      print "<BODY>\n";
      print "<H1>$document_title</H1>\n";
      print "<P>\n";
}

#html_trailer coloca la parte final al documento HTML
sub html_trailer
{
      print"</BODY>\n";
      print"</HTML>\n";
}

#blank-response imprime un mensaje de error si los datos no han sido
#ingresados
sub blank_response
{
    &html_header("Datos en blanco");
    print "<HR>";
    print "Sus comentarios o direccion email  aparece(n) en blanco, por
lo que no han ";
    print "sido enviados a $FORM{'correo'}. Por favor re-ingrese sus ";
    print "datos, o regrese a nuestro <i>HOMEPAGE</i><p>";
     print "<HR><A HREF = \"/mail.correo.pl?$FORM{'correo'}><IMG SRC =
\"/GRAFICOS/CATALOGO/back.gif\">Enviar nuevamente</IMG></A><HR>\n";
            print   "<A   HREF   =   \"/index.html\"><IMG   SRC   =
\"/GRAFICOS/CATALOGO/home.gif\">HOMEPAGE</IMG></A>\n";
    &html_trailer;
    exit;
}
```

Interfaz unificada de administración

###
#Nombre del programa: conjunto de programas desarrollados en Perl
#Ubicación: /usr/httpd/cgi-bin (maquina: 192.188.59.2)
#Autor: Servio Lima
#Lenguaje: Perl (Interpretador ver 4.0 o superior)
#Fecha Creación: 1/Nov/96
#Descripción :
Las herramientas de administración del sistema la conforman un conjunto de programas CGI que permiten realizar tareas simples como: borrado de archivos de bloqueo, borrado de colas de mensajes, etc. Todos los programas presentan el mismo formato de codigo, tal como se ve a continuación.
###

```
#Borrado de archivos de bloqueo
#Linea inicial de todo programa Perl
#!/usr/bin/perl
$dir="/home/httpd/cgi-bin";

#Separación del proceso padre e hijo
unless(fork){

#Proceso hijo
#Borrado de archivos .lock generados por ciertos programas CGI
exec("/bin/rm -f $dir/.lock*");
}
#Proceso padre espera terminación de proceso hijo
wait;

#Mensaje enviado al usuario administrador
print 'Icontent-type: text/html\n\n";
print "<html><head><title>Borrado de archivos .lock</title></head>\n";
print "<body background=/GRAFICOS/CATALOGO/espol6.gif>\n";
print "<h1>Borrado de archivos .lock</h1>\n";
print "<hr>Borrado Finalizado\n";
print "</body></html>\n";
```

Guestbook

###
#Nombre del programa: guestbookpl
#Ubicación: /usr/httpd/cgi-bin (maquina: 192.188.59.2)
#Autor: Servio Lima
#Lenguaje: Perl (Interpretador ver 4.0 o superior)
#Fecha Creación: 1/Jun/95
#Descripción:
guestbook.pl permite guardar en una cola de espera ubicada en el directorio "/home/httpd/docs/ESPOL/GUESTBOOK", los datos ingresados por un visitante, hasta que estos sean confirmados via email (ver programa mailrcpt.pl)
###

```
#Linea inicial de todo programa en Perl
#!/usr/bin/perl

#Definición de constantes del programa
Smailprog = '/usr/lib/sendmail';
$fecha ='date '+%m-%d-%y'`;

#Arreglo de obscenidades
@obscenidades=("...",".....","...","...");
chop($fecha);
Sdirectorio ="/home/httpd/docs/ESPOL/GUESTBOOK";
Swebmaster = 'httpd';
$webmaster_comment ="Administrador HTTPD";
$method = 'POST';

#Utilización del metodo WST para lectura de información enviada por el
#servidor web
if ($ENV{'REQUEST_METHOD'} eq $method)
                {
             #
             #El metodo POST determina que obtengamos
             #la entrada de la forma a partir de stdin
             #
             read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'));
             #
             #Dividir los pares nombre-valor tomando como
             #referencia '&'
             #
             @pairs = split(/&/, $buffer);
             #
             #Ir a traves de los pares y determinar el
             #nombre y valor para cada variable.
             #
```

```
            foreach $pair (@pairs)
                {
            ($name, $value) = split(/=/, $pair);
                $value =~ tr/+/ /;
            $value =~ s/%([a-fA-FO-91[a-fA-F0-9])/pack("C", hex($1))/eg;

            $FORM{$name} = $value;
              }
#blank-response es invocado cuando no se han ingresado uno de los
#siguientes campos: nombre, email, pais, comentario
      &blank response unless $FORM{'nombre'};
      &blank-response unless $FORM{'email'};
      &blank-response unless $FORM{'pais'};
      &blank_response unless $FORM{'comentario'};

#Chequeo del contenido del campo comentarios (verificar que no existan
#palabras obscenas)
for($i=0;$i<=$#obscenidades;$i++){
        if(index($FORM{'comentario'},$obscenidades[$i]) ge 0){
        &error("No se permiten palabras obscenas\n");
        }

}


#Apertura del proceso de email para envio de mensaje de gratitud a la
#persona que ingreso todos los datos
open (MAIL, "|$mailprog $FORM{'email')") || die "No se pudo abrir
$mailprog!\n";

      print MAIL "To: $FORM{'email'}\n";
      print MAIL "From: $webmaster ($webmaster-comment)\n";
      print MAIL "Reply-to: $webmaster ($webmaster_comment)\n";

      #Cuerpo del mensaje de email

      print MAIL "Subject: ESPOL: Gracias por su visita! \n\n";
      print MAIL "Con fecha $fecha, Usted ($FORM{'nombre'}) envio lo
siguiente:\n";
      print MAIL  "-------------------------------------------------
--------\n";
      print MAIL "$FORM{'comentario'}";
      print MAIL "\n-------------------------------------------------
---------\n";
      print MAIL "Seccion: $FORM{'region'}\n";
      print MAIL "Email:   $FORM{'email'}\n";
      print MAIL "-------------------------------------------------
-------\n";
      print MAIL "Gracias por sus comentarios, sugerencias o
criticas.\n";
      print MAIL "Sus comentarios seran publicados en 24 horas,
automaticamente\n";
      print MAIL "Nuestro equipo de trabajo, le queda cordialmente
agradecido.\n\n";
```

```
      print MAIL "Atentamente\n";
      print MAIL "----------\n";
      print MAIL "Grupo de Desarrollo de la ESPOL\n";
      print MAIL "httpd\@espol.edu.ec\n";
      close (MAIL);
      &html_header("Gracias $FORM{'nombre'}");
      print "<HR><P>\n";
      print "Sus comentarios seran anadidos, luego de 24 horas<p>\n";
      print "Usted recibir&aacute; un correo con los datos
ingresados<p>\n";

      print "<HR><A HREF = \"/index.html\"><IMG SRC =
\"/GRAFICOS/CATALOGO/home.gif\">Retornar al HOMEPAGE</IMG></A>\n";
      &html_trailer;
}
else
{
      #En caso de no haberse usado el método POST

      &html_header("Error en la forma de comentarios");
      print "<HR><P>\n";
      print "La Forma no fue procesada. Por favor envie";
      print "sus comentarios a $webmaster\n";
      &html_trailer;
}

#Almacenamiento de los datos ingresados por el visitante en la cola del
#Guestbook, y publicación en 24 horas (mediante el programa CRONTAB del
#sistema). Si se recibe via email un mensaje de error indicando que el
#usuario no existe (ver mailrctp.pl), entonces se borra el mensaje de
#la cola del Guestbook
($user,$domain) =split(/\@/,$FORM{'email'});

open(FILE,">$directorio/$FORM{'region'}.$user.$domain");

print FILE "<b>$FORM{'nombre'} (<a href=\"/cgi-
bin/correo.pl?$FORM{'email'}\">$FORM{'email'}</a>).</b> Fecha
:$fecha<br>\n";
print FILE "<b>Regi&oacute;n:</b> \u$FORM{'region'} <b>Pais:</b>
$FORM{'pais'}<br>";
print FILE "$FORM{'comentario'}<br>";
close(FILE);

#html_trailer cierra el cuerpo de un documento HTML
sub html_trailer
{
      print"</BODY>\n";
      print"</HTML>\n";
}

#blank-response envia un mensaje indicando que los datos no han sido
#ingresados
sub blank_response
{
```

```
    &html_header("Datos en blanco");
    print "<HR>";
    print "Sus comentarios o direccion email aparece(n) en blanco, por
lo que no han ";
    print "sido enviados a $FORM{'correo'}. Por favor re-ingrese sus ";
    print "datos, o regrese a nuestro <i>HOMEPAGE</i><p>";
     print "<HR><A HREF = \"/mail.correo.pl?$FORM{'correo'}><IMG SRC =
\"/GRAFICOS/CATALOGO/back.gif\">Enviar nuevamente</IMG></A><HR>\n";
            print "<A HREF = \"/index.html\"><IMG SRC =
\"/GRAFICOS/CATALOGO/home.gif\">HOMEPAGE</IMG></A>\n";
    &html_trailer;
    exit;
}

#error envia un mensaje de error $status
sub error

    local($status) = @_;
    &html_header($status);
    print "<HR>";
    print "<blink><h3>$status </h3></blink>";
    print "Por favor reintente o regrese a nuestro <i>HOMEPAGE</i><p>";
        print "<HR><A HREF = \"/ESPOL/guestbook.html\"><IMG SRC =
\"/GRAFICOS/CATALOGO
/back.gif\">Regresar</IMG></A><HR>\n";
            print "<A HREF = \"/index.html\"><IMG SRC =
\"/GRAFICOS/CATALOGO/home.gif\">
HOMEPAGE</IMG></A>\n";
    &html_trailer;
    exit;
}



#html_trailer abre un documento HTML
sub html_header
{
        $document_title = $_[0];
        print "Content-type: text/html\n\n";
        print "<HTML>\n";
        print "<HEAD>\n";
        print "<TITLE>$document_title</TITLE>\n";
        print "</HEAD>\n";
            print "<BODY BACKGROUND=\"/GRAFICOS/CATALOGO/espol6.gif\"
text=\"000000
\" link=\"0000ff\" alink=\"ff0000\" vlink=\"0000ff\">\n";
        print "<H1>$document_title</H1>\n";
        print "<P>\n";
}
```

BIBLIOGRAFÍA

1. STALLINGS, William. **Data and Computer Communications.** MacMillan Publishing Company, New Jersey-USA. 4ta edicion. 1994.875 pp

2. STEVENS, Richard. **Unix Network Programming.** Prentice Hall, New-Jersey-USA. 1990. 772 pp

3. VALLEY, John. **C Programming for Unix.** Sams Publishing. Indiana-USA. Ira edicion. 1992. 644 pp

4. DECEMBER, John - Mark GINSBURG. **Html & Cgi Unleashed.** Sams.net Publishing. Indianapolis-USA. 1ra edicion. 1995. 830 pp

5. SCHWATZ, Randal. **Learning Perl.** O'Reilly & Associates, Inc. USA. 1ra edicion. 1993. 246 **pp**

6. COMMER, Douglas E. **Internetworking with TCP/IP volume 111,** 1ra edicion, 1991, 498 PP

		postal o texto donde los quiebre de línea son siqnificativos
Imaaenes		
<IMG> </IMG>		Ubica una imagen grafica en un documento en donde se coloque el taq.
	Src=''URL''	Identifica el archivo fuente de la imaqen
	41t=''string''	Un string de caracteres puede ser definido y este sera mostrado en browsers no graficos
	Align=''top|middle|bottom"	Configura la relación de posición entre el grafico y el texto que sigue; los valores pueden ser: top: el texto siguiente al grafico debe estar alineado con el tope del grafico **middle:** el texto que sigue al grafico debe estar alineado con la mitad del grafico **bottom:** el txto que sigue al grafico debe estar alineado con la parte inferior del qrafico
	Ismap	Identifica la imagen como un mapa sensitivo.

Elementos de HTML Nivel I

Formateo de caracteres		
Taq (inicio...fin)	Atributos	Explicación
<CITE>string</CITE>		Delimita una citación
<CODE>string</CODE>		Delimita codigo fuente de software
<EM>string</EM>		Delimita texto enfatizado
<KBD>string</KBD>		Delimita texto que ha sido ingresado como entrada de teclado
<SAMP>string</SAMP>		delimita texto que debe ser mostrado tal como es.
<STRONG>string</STR		Delimita texto con enfasis
<VAR>string</VAR>		Delimita un nombre de variable.
<B>string</B>		Delimita texto en bold
<I>string</I>		Delimita texto en italicas
<TT>string</TT>		Delimita texto de maquina de escribir.

Elementos de FORMAS Nivel 2 HTML

Elementos de FORM		
Taq (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 la forma para su procesamiento
	Method=''get|post''	Indica la variación en el protocolo manejador de la forma que sera usado en el procesamiento del script
	Enctype=''string''	Identifica el tipo de medio (ver RFC1590) que sera usade para la codificación de los pares nombre/valor de los datos de la forma. Esto es necesitado cuando el protocolo identificado en Method no tiene su propic formato.
		Usado para recoger information del usuario
	Align=''top|middle|bottom''	Usado sole con el tipc image. Posibles valores son: top, middle y bottom, y definen la relación de la imagen con el texto siguiente
	Checked	Causa que el estado inicial de un checkbox o radio boton sea el "seleccionado". Sin este stributo, el estado inicial es "no seleccionado''
	Maxlength=''N''	Configura el número maximo de caracteres que un usuario puede ingresar en un campo texto. El valor default es ilimitado

<INPUT> </INPUT>	Name=''string''	Identifica el nombre simbolico que es usado en la transferencia e identificación de la salida de este elemento de la forma.
	Size=''N''	Especifica el ancho del campo que es mostrado al usuario. Si el tamaño es menor que Maxlength, al campo de texto se le pueds hacer un scroll.
	Src=''URL''	El archivo fuente para la imagen usado con el atributo Type.
	Type=''checkbox|hidden|image|password|radio|reset|submit|text"	Identifica el tipo de campo de ingreso: 1. checkbox es usado para recoger datos que pueden tener multiples valores a la vez. 2. Hidden es para valores que son dados por la forma sin ser ingresados por el usuario. 3. Image es una campo imagen usado para someter la forma: cuando el usuario hace click sobre la imagen, la forma es sometida, y las coordenadas x y y de la localización del click son transmitidas con los pares nombre/valor. 4. password: es un campo en el cual el usuario ingresa texto, pero el texto no es presentado (aparece como estrellas) 5. radio es usado para recoger información donde hay uno y solo un posible valor de un conjunto de alternativas. El atributo Checked puede setear el valor inicial de este elemento.

		6. reset: es usado para resetear y limpiar la forma a sus valores default. El atributo Value setea el string mostrado al usuario en este elemento 7. submit: es un botón usado para someter la forma. El atributo Value setea el string mostrado al usuario en este elemento 8. text: es usado para una linea simple de texto; para multiples lineas usar: TEXTAREA
	Text	Identifica la entrada como una area de entrada de texto simple
	value=''string''	setea el valor mostrado inicialmente del campo o el valor del campo cuando es seleccionado.
<SELECT>...</SELECT>		Usado para presentar un usuario con una elección de un conjunto de alternativas. El elemento OPTION es usado para definir cada alternativa.
	Name=''string''	Identifica el nombre lógico que sera sometido y ssociado con los datos como resultado de la selección del usuario
	Multiple	Por default, el usuario puede solo hacer una selección de un grupo en el elemento SELECT. Jsando el atributo Multiple, el usuario puede seleccionar una o mas de las opciones.
	Size=''N''	Especifica el numero de items visibles. Si es mas de uno, la muestra visual derá una lista.

<OPTION>...</OPTION>		Ocurre solo dentro del elemento SELECT y es usado para representar cada opción de SELECT.
	Selected	Indica que esta opcion esta inicialmente seleccionada
	Value=''N	Si esta presente, su valor sera retornado por SELECT. Si esta opcion es elegida; de otra manera, el valor retornado es el seteado por el elemento OPTION
<TEXTAREA>...</TEXTAREA>		Usado para recoger multiples líneas de texto del usuario; al usuario se le muestra un area donde puede escribir.
	Name=''string''	Identifica el nombre lógico que sera asociado con el texto retornado.
	Rows=''N''	El numero de filas de texto que sera mostrado (el usuario puede mostrar mas filas y hacer un scroll de ellas)
	Cols=''N''	El numero de columnas de texto que sera mostrado (el usuario puede usar mas columnas y hacer un scroll hacia la derecha)

Elementos de TABLAS Nivel 3 HTML

Elementos de TABLE		
Taq (inicio...fin)|	Atributos	Explicación
		Delimita el contenido de una tabla
<TABLE> </TABLE>	Align =''bleedleft|left|center|right|bledright|justify"	El alineamiento horizontal de la tabla en la ventana (no el contenido de la tabla). Posibles valores son:
		bleedleft:alinear a la izquierda del borde de la ventana left:a la izquierda del texto center: centrado entre margenes de texto right: a la derecha deltexto bleedright: alineado a la derecha del borde de la ventana justify: tabla debe llenar el espacio entre los marqenes de texto.
	Border	Causa que el browser grafique un borde alrededor de la tabla;
	Width=''N''	Especifica cuan ancha es la tabla. Se lo puede dar como %
	Colspec="LN RN CN''	Especifica la alineacion de los items en las columnas
	Jnits=''	Identifica las unidades a ser usadas en medidas: el default es "en" (1/2 punto)
<CAPTION>string</CAPTION>		Jsado para etiquetar la tabla
	align=''top|bottom|left|right''	Identifica la posición de la etiqueta relativa a la tabla o figura

<TH>string</TH> <TD>string</TD>		TH usado para etiquetar un encabezado en la tabla; TD etiqueta los datos en la tabla
	Align=‘‘left|center|right|justify|decimal’’	Identifica el alineamiento horizontal de los items en una fila
	Valign=‘‘top|middle|bottom|baseline”	Identifica el alineamiento vertical de los items en una celda
	Colspan=‘‘N’’	Identifica el numero de columnas que la celda *se* expande
	Rowspan=‘‘N’	Identifica el numero de filas que la celda se expande
	Nowrap	Previene al browser de hacer un wrap del contenido de la celda

Elementos HTML: Extensiones de Netscape Navigator

Elementos de BODY		
Taq (inicio...fin)	Atributos	Explicación
		Delimita el contenido del BODY
<BODY>...</BODY>	BACKGROUND=''URL	Identifica el URL del grafico que sera repetido como el fondo de la pagina. El usuario no verá este grafico en browsers que no lo permitan o si ··image loading" esta apagado, o *si* el usuario ha sobrepuesto sus propios fondos.
	Bcolor=''#RRGGBB	Especifica un color de background sólido. El color es especificado usando un codigo color decimal para rojo, verde y azul
	Text=··#RRGGBB'·	Especifica el color del texto del documento
	Link=''#RRGGBB''	Especifica el color de los enlaces del documento
	Vlink=''#RRGGBB	Especifica el color de los enlaces visitados del documento
	Alink=''#RRGGBB ''	Especifica el color de los enlaces activos del documento
<HR>...</HR>		Crea una regla horizontal
	Size=''N''	Identifica el grosor de la linea
	Noshade	Apaga el sombreado para crear una barra solida
	Width=''N|N%''	Identifica el ancho de la linea, expresado en pixeles o relativo a un porcentaje de la pantalla actual.
	Align=''left|right|center"	Especifica la alineación de las lineas horizontales que son menores que el ancho total de la pagina

		Cambia el tipo de letra actual
<BASEFONT>...</BASEFONT>	Size=''N''	especifica el tamaño de tipo de letra rango: 1-7
<BLINK>string</BLINK>		Crea texto centelleante
<CENTER>...</CENTER>		Centra el texto
<IMG>...</IMG>		Adiciones al elemento IMG
	Align=' left|right''top texttop|middle|absmiddle|baseline|absbottom"	Especifica el lugar de la imagen relativa al texto que la sigue
<TABLE>...</TABLE>		Adiciones al elemento TABLE
	Width='N'	Especifica el grosor del borde de la tabla
	Cellspacing=''N''	especifica el espacio alrededor de las celdas de la tabla
	Cellpadding=''N''	especifica el espaciado de los datos en la celda

Códigos caracteres especiales en los URL

Caracter	Código de Escape
SPACE	%20
<	%3C
>	%3E
#	%23
%	%25
{	%7b
}	%7d
|	%7C
\	%74
^	%5E
~	%7E
[	%5b
]	%5D
`	%60
;	%3B
/	%2F
?	%3A
@	%40
=	%3D
&	%26

Cuadro de los tipos MIME más usados

Tipo	Subtipo	Extensiones de archivo típicas
text	enriched	
	html htm	html
	plain	txt
	tab-separated-values	
	richtext	
multipart	alternative	
	appledouble	
	diaest	
	header-set	
	mixed	
	parallel	
nessage	external-body	
	news	
	partial	
	rfc822	

Tipo	Subtipo	Extensiones típicas de archivo
application	activemessaqe	
	andrew-inset	
	applefile	
	atomicmail	
	commonqround	
	cybercash	
	dca-rft	
	dec-dx	
	eshop	
	iqes	
	mac-binhex40	hax
	macwriteii	
	mathematica	
	msword	doc
	news-messaqe-id	
	news-transmission	
	octet-stream	tar dump readme bin uu exe
	oda	oda
	pdf	pdf

	postscript	ps eps ai
	\remote-printinq	
	riscos	
	rtf	rtf
	slate	
	wita	
	wordperfect5.1	
	x-dvi	dvi
	x-pdf	pdf
	x-tar	tar
	x-tex	tex
	x-www-form-urlencoded	
	x-www-pqp-request	
	x-www-pqp-reply	
	x-www-local-exec	
	zip	zip
image		qif
		ief
	jpeq	jpeg jpg jpe
	rqb	rqb
	tiff	tiff tif
	xbm	xbm
	xpm	xpm
	x-xwindowsdump	xwd
	x-pict	pict
audio	basic	au snd
	x-aiff	aif aiff aifc
video	x-mpeq	mpeq mpq mpe
	quicktime	qt mov
	x-msvideo	avi
	x-sqi-movie	movie

Diferentes Formatos de Imágenes

Extensión de Archivo	Tipo de Archivo
bmp	Microsoft Windows bitmap file
cur	Microsoft Windows cursor file
eps	Encapsulated Postscript
gif	Compuserver qraphics imaqe format file
hdf	Hierarchical data format file
ico	Microsoft Windows icon file
icon	Sun icon and cursor file
mpnt	MacPaint file
pbm	Portable bitmap file
pgm	Portable grayscapel map file
pic	PIXAR picture file
pict	Macintosh QuickDraw/pict file
pict	Softimage pict file
pix	Alias pixel image file
pnm	portable any map file
ppm	portable pixel map file
ps	Postcript
ras	Sun RASterfile
rgb	Silicon Graphics RGB image file
rla	Wavefront raster image file
rpbm	Raw portable bitmap file
rpgm	Raw portable grayscale map file
rpnm	Raw portable any map file
rppm	Raw portable pixel map file
synu	Synu image file
tga	Truevision targa image file
tiff	Tagged image file
viff	Khoros visualization image file format
xbm	X11 bitmap file
xwd	X window dump image file
png	Portable network qraphic

APÉNDICE C

Codigo fuente de los CGI implementados

Directorio de Homepages

```
##########################################################################
#Nombre del programa: enlaces.pl
#Ubicación: /usr/httpd/cgi-bin (maquina: 192.188.59.2)
#Autor: Servio Lima
#Lenguaje: Perl  (Interpretador ver 4.0 o superior)
#Fecha Creación: 1/Nov/95
#Descripción :
Programa que permite adicionar un usuario al Directorio de Homepages de la ESPOL,
una vez que se han verificado TODOS sus datos
##########################################################################
```

```
# Linea inicial de todo programa en Perl
#! /usr/bin/perl

# Definición de variables
$filename = "/home/httpd/docs/ESPOL/HOMEPAGES/letra";
$filename-url = "http://www.espol.edu.ec/ESPOL/HOMEPAGES/letra";
$nombre_espol ="espol.edu.ec";
#Archivos de bloqueo
$lock="/home/httpd/cgi-bin/.lock_enlaces";
$lock2="/home/httpd/cgi-bin/.lock_enlaces2";

#Grupos de la ESPOL de acuerdo al archivo /etc/groups
$profesores = 100;
$estudiantes = 110;
$administrativos =120;

# Posibles mensajes devueltos por el programa servidor web
%OkStatusMsgs = (
  200, "OK 200",
  201, "CREATED 201",
  202, "Accepted 202",
  203, "Partial Information 203",
  204, "No Response 204",
  302, "Found",
```

```
);

%FailStatusMsgs = (
  -1,  "Could not lookup server",
  -2,  "Could not open socket",
  -3,  "Could not bind socket",
  -4,  "Could not connect",
  301, "Posibles causas: 1. Falta / al final del enlace; 2. Permisos de
lectura no validos del directorio de usuario, directorio public-html o
del archivo index.html; 3. Ingreso duplicado de la Informaci&oacute;n",
  303, "Metodo",
  304, "No Modificado",
  400, "Ma1 requerimiento",
  401, "No autorizado",
  402, "Pago Requerido",
  403, "Negado",
  404, "No encontrado",
  500, "Error Interno ",
  501, "No implementado",
  502, "Temporal sobrecarga del servicio ",
  503, "Gateway timeout ",
  600, "Ma1 requerimiento",
  601, "No implementado",
  602, "Conexion fallida (host no encontrado?)",
  603, "Timed out",
);


#Obtención de datos enviados por el servidor web a traves del STDIN del
programa
read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});

# Descomponer los datos recibidos en pares nombre=valor
@pairs = split(/&/, $buffer);

#Almacenamiento de los pares nombre=valor en el arreglo FORM
foreach $pair (@pairs)
{
    ($name, $value) = split(/=/, $pair);

    # Convertir los + a espacio en blanco y los códigos hexadecimales a
      su caracter respectivo
    $value =~ tr/+/ /;
    $value =~ s/%([a-fA-FO-91[a-fA-FO-91)/pack("C", hex($1))/eg;

    #Almacenamiento de los pares nombre=valor en el arreglo FORM
    $FORM{$name} = $value;
}

#Ingreso de las secciones del Directorio de Homepages en un arreglo
asociativo
%sections=("a1um","Alumnos","prof","Profesores","admi","Administrativos
","visi","Visitantes");
```

```
#blank response se ejecuta si no se ha llenado el campo respectivo
&blank-response unless $FORM{'email'};
&blank-response unless $FORM{'url'};
&blank_response unless $FORM{'title'};

#Chequeo de que el campo 'url' sea un URL válido
if($FORM{'url'} !~ /^http:\/\//){
      &URL response3("Enlace No v&aacute;lido\n");
      exitj
      }

#Llamada a la subrutina Check_External_URLs que chequea si el URL
existe
$retorno = &Check_External_URLs($FORM{'url'});
if($retorno){
      &URL response($retorno);
      exitj
      }

#Division del campo 'email' en user y dominio
if ($FORM{'email'} =~ /@/&& $FORM{'email'} !~ /;/){
($user,$domain) =split(/@/,$FORM{'email'});
}
else{
      &URL response3("Formato invalido de email");
      exitj
}

#Chequeo de que el usuario pertenezca a la sección que dice pertenecer
if ($FORM{'section'} eq "Alumnos" || $FORM{'section'} eq "Profesores"
|| $FORM{'section'} eq "Administrativos"){
      if ($domain eq $nombre_espol){
      ($dato)=(getpwnam($user))[3];
      if ($dato ne Sprofesores && $dato ne $estudiantes && $dato ne
      $administrativos) {
        &URL response3("Ud. no esta registrado en $FORM{'section'}");
        exit;
        }
      if ($dato ne Sprofesores && $FORM{'section'} eq "Profesores"){
      &URL response3("PROFESOR no registrado en $nombre_espol.");
      exitj
      }
      else{
            if($dato ne $administrativos && $FORM{'section'} eq
      "Administrativos"){
      &URL_response3("ADMINISTRATIVO no registrado en $nombre_espol.");
      }
      if($dato eq Sprofesores && $FORM{'section'} ne "Profesores"){
      &URL response3("Ud. debe ingresarse en la seccion Profesores");
      exit;
      }
    }
}
else{
```

```
        &URL_response3("Ud. no est&aacute; registrado en la ESPOL");
        exit;
    }
}


#Chequeo de que el URL no ha sido registrado previamente
dbmopen(SLINKS,"links",0644);
        if($LINKS{"$FORM{'email'}") eq $FORM{'url'}){
        URL_response2($FORM{'email'});
        dbmclose(BLINKS);
        exit;
        }
#Chequeo de que la dirección email no ha sido registrada previamente
        else{
        while(($key,$value)=each(BLINKS)){
        if($key eq $FORM{'email'}){
                URL_response3("Registrado previamente");
                dbmclose(%LINKS);
                exit;
            }
        }
        #Se adiciona el usuario a la base de datos del Directorio
        Telefonico
        $LINKS{"$FORM{'email'}"}= "$FORM{'url'}";
}
dbmclose(BLINKS);


# Respaldo del archivo que contendra el hiperenlace hacia el usuario
open(COPIA,"|cp                                $filename$FORM{'letter'}.html
$filename$FORM{'letter'}.bak");
close(COPIA);

#A partir de while(1) se inicia una sección crítica del programa. En
este caso se protege la lectura archivo que contendra el hiperenlace
hacia el usuario.
while(1){
          if (-e $lock2){
          next;
          }
          else{
          open(LOCK,">$lock2");
           #apertura en modo lectura del archivo que contendra el
        hiperenlace hacia el usuario
          open (FILE,"$filename$FORM{'letter'}.html");
          while (<FILE>){
          #almacenamiento del archivo que contendra el hiperenlace hacia
              el usuario en una variable ($raw-data)
              $raw-data .= $_;
          }
          close(FILE);
          close(LOCK);
```

```
        unlink($lock2);
        last;
        }
}
unlink($lock2);
#Fin del bloqueo y de seccion critica



undef $/;

#a partir de while(1) se inicia una sección critica en la que protege
la escritura del archivo (FILE) donde se colocara el hiperenlace del
usuario.
while(1){
        if (-e $lock2){
        next;
        }
        else{
        open(LOCK2,">$lock2");
        #Almacenamiento de cada linea del archivo (FILE) representado
      por la variable raw-data en el arreglo @proc_data.
        @proc_data = split(/\n/,$raw_data);

      #Apertura del archivo donde se colocara el hiperenlace del
      usuario. Se destruye el anterior archivo y se almacena el
      contenido del arreglo @proc_data en el nuevo archivo, conteniendo
      el hiperenlace del usuario.
        open (FILE,">$filename$FORM{'letter'}.html");
      foreach $line (@proc_data)  # For every line in our data
      I
        print FILE "$line\n";
        #Se inserta el hiperenlace del usuario en la sección
      correspondiente (keys(%sections))
        foreach $tag (keys(%sections))
        {
                if ( ($FORM{section} eq $sections{$tag}) &&
                     ($line =~ /<META $tag>/) )
                        I
                        print FILE "<img
                        src=\"/GRAFICOS/CATALOG0/b1ueba11.gif\"><a
                        href=$FORM{url}>$FORM{title}</a><br>\n"

                        }
        }
}
close (FILE);
   close(LOCK2);
        unlink($lock2);
        last;
        }
} #Fin de bloqueo y de seccion critica
```

```
unlink($lock2);
```

#Retorno del hiperenlace el archivo que contiene el nuevo hiperenlace, hacia el browser del usuario

```
print "Location: $filename_url".$FORM{'letter'}.".html\n\n";

#=========================================================================
# html_trailer termina el cuerpo de un documento HTML
#=========================================================================
sub html_trailer
{
        print"</BODY>\n";
        print"</HTML>\n";
}
#=========================================================================
#html_header empieza el cuerpo de un documento HTML
#=========================================================================
sub html_header
{
        $document_title = $_[0];
        print "Content-type: text/html\n\n";
        print "<HTML>\n";
        print "<HEAD>\n";
        print "<TITLE>$document_title</TITLE>\n";
        print "</HEAD>\n";
        print "<BODY>\n";
        print "<H1>$document_title</H1>\n";
        print "<P>\n";
}

#=========================================================================
#blank-response envia mensajes de error en caso de campos incompletos
sub blank-response
{
    &html header("Datos en blanco");
    print-"<HR>";
    print "<blink><h3>Sus datos  aparecen en blanco</h3></blink>";
    print "Por favor reintente o regrese a nuestro <i>HOMEPAGE</i><p>";
      print   "<HR><A   HREF   =   \"/ESPOL/alumnos.html\"><IMG   SRC   =
\"/GRAFICOS/CATALOGO/back.gif\">Regresar</IMG></A><HR>\n";
        print    "<A    HREF    _    \"/index.html\"><IMG    SRC    =
\"/GRAFICOS/CATALOGO/home.gif\">
HOMEPAGE</IMG></A>\n";
    &html_trailer;
    exit;
}
```

```
#=============================================================================
#URL_response envía el código de error dado por el servidor web, en
#formato HTML
#=============================================================================
sub URL_response
{
    local($status) = @_;
    &html_header("Error en el Enlace");
    print "<HR>";
    print "<blink><h3>Error en el Enlace</h3></blink>";
    print "<p> xx Falla: $FailStatusMsgs{$status}<p>\n";
    print "Por favor reintente o regrese a nuestro <i>HOMEPAGE</i><p>";
        print "<HR><A HREF = \"/ESPOL/alumnos.html\"><IMG SRC =
\"/GRAFICOS/CATALOGO
/back.gif\">Regresar</IMG></A><HR>\n";
            print "<A HREF = \"/index.html\"><IMG SRC =
\"/GRAFICOS/CATALOGO/home.gif\">
HOMEPAGE</IMG></A>\n";
    &html_trailer;
    exit;
}


#=============================================================================
#URL_response envía el mensaje de error en HTML cuando el usuario o el
#URL ya existen.
#=============================================================================
sub URL_response2
{
    local($status) = @_;
    &html_header("Error en el Enlace");
    print "<HR>";
    print "<blink><h3>Error en el Enlace</h3></blink>";
        print "<p> xx Falla: Enlace ya ha sido registrado por
$status<p>\n";
    print "Por favor reintente o regrese a nuestro <i>HOMEPAGE</i><p>";
        print "<HR><A HREF = \"/ESPOL/alumnos.html\"><IMG SRC =
\"/GRAFICOS/CATALOGO
/back.gif\">Regresar</IMG></A><HR>\n";
            print "<A HREF = \"/index.html\"><IMG SRC =
\"/GRAFICOS/CATALOGO/home.gif\">
HOMEPAGE</IMG></A>\n";
    &html_trailer;
    exit;
}

#=============================================================================
#URL_response imprime el mensaje recibido en la variable $status
#=============================================================================
sub URL_response3
{
    local($status) = @_;
    &html_header($status);
    print "<HR>";
```

```
    print "<blink><h3>$status </h3></blink>";
    print "Por favor reintente o regrese a nuestro <i>HOMEPAGE</i><p>";
        print "<HR><A HREF = \"/ESPOL/alumnos.html\"><IMG SRC =
\"/GRAFICOS/CATALOGO
/back.gif\">Regresar</IMG></A><HR>\n";
            print "<A HREF = \"/index.html\"><IMG SRC =
\"/GRAFICOS/CATALOGO/home.gif\">
HOMEPAGE</IMG></A>\n";
    &html_trailer;
    exit;
}

#=========================================================================
#Check_External_URLs invoca a Check_URL para chequear la validez de un
#hiperenlace y retorna un código de error $rcode
#=========================================================================
sub Check_External-URLs {

   local($URL) = @_;

   #invocation de la función Check_URL();
   $rcode = &Check_URL($URL);

#Se pregunta si el codigo de regreso del servidor web (rcode) se
encuentra dentro del arreglo OkStatusMsgs, lo cual indica que el URL
esta correcto
  if (defined($OkStatusMsgs{$rcode})) {
        return 0;
  }
  else {
#Si el URL esta incorrecto, se verifica que el código de retorno del
servidor web (rcode) se encuentre definido en el arreglo FailStatusMsgs
    if (defined($FailStatusMsgs{$rcode})) {
        return $rcode;
    }
    else {
#Si no se encuentra en ninguno de los arreglos mencionados, simplemente
se devuelve el codigo de retorno del servidor web (rcode)
        return $rcode;
    }
  }
}

#=========================================================================
#Check_URL chequea la validez de un hiperenlace comunicándose con el
#puerto 80 de cualquier servidor web.
#=========================================================================
sub Check_URL {


local($URL) = @_;

#Chequeo de que el URL tenga al menos "http://"
```

```
if ($URL !- m#^http://.*#i) {
  print "Formato incorrecto http\n";
  return;
}
else {
#Se separan los elementos del URL en host, puerto y path.
if ($URL =~ m#^http://([\w-\.]+):?(\d*)($|/(.*))#) {
   $host = $1;
    $port = $2;
    $path = $3;
  }
#Se completan los datos del path y el puerto, en caso de que esten
incompletos.
  if ($path eq "") { $path = '/'; }
  if ($port eq "") { $port = 80; }

#Se borra del path todo elemento que sea un enlace hacia otra sección
del mismo documento HTML (anchors)
  $path =- s/#.*//;
#Se borran parámetros enviados despues del signo de interrogación ?, en
caso de que se trate de accesar un URL que envíe parámetros a un
programa CGI.
  #$path =- s/\?.*//;

}


#Definición de los parámetros necesarios para la conexión
cliente/servidor a través de sockets
$AF_INET = 2;
$SOCK-STREAM = 1;
$sockaddr = 'S n a4 x8';

#Definición del nombre del host local, el nombre del protocolo a
usar(proto), el nombre del servicio a usar (port).
chop($hostname = 'hostname');
($name,$aliases,$proto) = getprotobyname('tcp');
($name,$aliases,$port) = getservbyname($port,'tcp') unless $port =~
/^\d+$/;
($name,$aliases,$type,$len,$thisaddr) = gethostbyname($hostname);
if (!(($name,$aliases,$type,$len,$thataddr) = gethostbyname($host))) {
  return -1;
}

$this = pack($sockaddr, $AF_TNET, 0, $thisaddr);
$that = pack($sockaddr, $AF_INET, Sport, $thataddr);

#Apertura del socket de comunicacion
if (!(socket(S, $AF_INET, $SOCK-STREAM, $proto))) {
  $SOCK-STREAM = 2;
  if (!(socket(S, $AF_TNET, $SOCK-STREAM, $proto))) { return -2; }
}
```

```
# Se le da al socket una dirección
if (!(bind(S, $this))) {
  return -3;
}

#Se establece la conexión cliente/servidor
if (!(connect(S,$that))) {
  return -4;
}

#Se define como via de comunicación a la salida estandar STDOUT
select(S); $| = 1; select(STDOUT);

#Se envia un comando HTTP/1.0 a traves del socket (comando HEAD)
print S "HEAD $path HTTP/1.0\n\n";

#Se recibe la respuesta del servidor
$response = <S>;
($protocol,$status) = split(/ /, $response);
close(S);
return $status;

}
```