UNIVERSIDAD DON BOSCO
FACULTAD DE INGENIERIA
ESCUELA DE INGENIERIA EN COMPUTACION

DISEÑO Y ELABORACIÓN DE UN SOFTWARE PROTOTIPO PARA LA CREACIÓN DE UN SERVIDOR WEB, ORIENTADO A LA MEDIANA EMPRESA DE EL SALVADOR

TRABAJO DE GRADUACION PARA OPTAR AL GRADO DE
INGENIERO EN CIENCIAS DE LA COMPUTACION

PRESENTADO POR:
BR. DAYSI ELIZABETH MINEROS VALENCIA

CIUDADELA DON BOSCO DICIEMBRE 2003

UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERIA

RECTOR

ING. FEDERICO MIGUEL HUGUET RIVERA

VICERECTOR

PRESBÍTERO VICTOR BERMÚDEZ

SECRETARIO GENERAL

LIC. MARIO RAFAEL OLMOS ARGUETA

DECANO DE LA FACULTAD DE INGENIERIA

ING. CARLOS GUILLERMO BRAN

UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERIA

ESCUELA DE INGENIERIA EN COMPUTACION

JURADO EVALUADOR DEL TRABAJO DE GRADUACIÓN:

DISEÑO Y ELABORACIÓN DE UN SOFTWARE PROTOTIPO PARA LA CREACIÓN DE UN SERVIDOR WEB, ORIENTADO A LA MEDIANA EMPRESA DE EL SALVADOR

Lic. Santiago Abarca Fuentes	Ing. Carlos Alberto Reyes Quintero
JURADO	**JURADO**
Lic. Fidias Edgardo Alfaro Arévalo	Ing. Guillermo Antonio Valencia
JURADO	**ASESOR**

AGRADECIMIENTOS

A Dios Padre Todopoderoso; por haberme brindado la confianza, el conocimiento y la fortaleza para culminar mi trabajo de graduación y mi carrera universitaria, gracias por no haberme abandonado durante los momentos difíciles y haber escuchado todas mis oraciones.

A mis Padres y Hermanos; por el apoyo, la confianza, las palabras de aliento, la paciencia que tuvieron en los momentos difíciles, por sus oraciones y consejos, en fin por toda la ayuda que me brindaron de una u otra forma les agradezco de todo corazón.

A mis amigos especiales, por su apoyo incondicional, su ayuda desinteresada, por sus palabras de aliento y su plena confianza en mi capacidad.

Un agradecimiento especial a todas las personas que de alguna manera me proporcionaron su ayuda y su apoyo durante mi carrera universitaria y durante el proceso de graduación.

Daysi Elizabeth Mineros Valencia.

INDICE

INTRODUCCIÓN

Vivimos en una era de globalización y la tendencia mundial es la de abrir mercados en países lejanos. Dentro de este proceso, los medios masivos de comunicación juegan un papel crucial. El Internet cabe dentro de estos medios, pero ofrece ventajas que los demás no tienen; reducción de costos y una cobertura mundial las 24 horas del día y los 365 días del año.

Internet cobra una gran importancia dentro de las empresas. Por un lado se ofrece una imagen adaptada a las nuevas tecnologías, necesaria en la sociedad de la información, y por otro le supone ahorro en costos de comunicaciones y de tiempos de desarrollo, producción y disposición del producto.

Un gran número de comercios y empresas, conocedoras de la relevancia que día a día asume la importancia de contar con un Sitio Web, no se han aventurado en este proceso, aun sabiendo que traería importantes beneficios para sus negocios; lo anterior debido a que consideran que este tipo de servicios utilizan tecnologías avanzadas pudieran resultar caras, y a que no cuentan con una idea clara y concreta de como iniciarse en el proceso.

La mayor parte del sector empresarial salvadoreño, en especial las Medianas Empresas, no esta integrado a Internet, debido a que la mayoría de las alternativas actuales no son completamente compatibles con su capacidad de inversión y mantenimiento, sobrepasando el valor de su presupuesto; o porque simplemente, no conocen los beneficios que pueden obtener.

Por estas razones, se plantea la necesidad de proporcionar alternativas orientadas a construir o mejorar la infraestructura tecnológica del sector empresarial de El

Salvador. En el presente documento se estructura una propuesta de solución al problema.

En el Capitulo I, se plantean la problemática acontecida y la situación actual, así como se destaca la importancia de realizar esta investigación, identificando los sectores objetivo hacia los cuales se enfoca el desarrollo de la misma. Por otra parte, se establece la finalidad del estudio a través de la definición de objetivos, alcances y limitaciones, así como se define, el marco teórico relacionado a la problemática establecida.

En el Capitulo II, se demuestra mediante una investigación de campo, las causas y efectos ocasionados por la problemática planteada a través de la presentación y el análisis de los resultados obtenidos. De igual manera, se detallan los procedimientos involucrados en dicha investigación, haciendo referencia a las técnicas de ingeniería utilizadas.

En el Capitulo III, se presenta el Diseño del Prototipo de Software de Servidor Web elaborado, describiendo la estructura y funcionamiento del mismo, así como la herramienta para gestionar su administración y control.

Finalmente, en el Capitulo IV, se describe el Desarrollo del Prototipo de Software de Servidor Web, utilizando los diagramas de flujo, para describir cada proceso elaborado.

CAPITULO I. PLANTEAMIENTO DEL PROBLEMA

1.1 ANTECEDENTES

Internet es un moderno y versátil medio de comunicación que conecta a millones de personas en todo el mundo, a través de sus computadoras permitiéndoles intercambiar mensajes, accesar a bases de datos, efectuar consultas y transacciones de todo tipo.

El origen de Internet se remonta hacia mediados de la década de los 60 y se basó en una red de comunicaciones creada por el ministerio de defensa de los Estados Unidos, durante la guerra fría contra la antigua Unión Soviética. En 1969 se consiguió por primera vez, conectar cuatro computadoras mediante líneas de alta velocidad, es el nacimiento de ARPANET (Advanced Research Project Agency Network).

Para 1972, aproximadamente 40 universidades ya formaban parte de la ARPANET, y sus computadoras tenían la capacidad de intercambiar mensajes y archivos, además de controlar computadoras a distancia.

En 1984, quince años después de su aparición, la ARPANET contaba con sólo 1024 servidores. A finales de 1989 eran 160.000 los servidores, en 1992 eran ya 1.000.000 y más de 8.000 las redes que funcionaban en el mundo.

En 1990 se creó el World Wide Web por un grupo de investigadores bajo la dirección de Tim Berners-Lee en el Laboratorio Europeo de Física de Partículas, CERN, situado en Suiza. La primera demostración pública del WWW se realiza en diciembre de 1991 en "Hypertext'91" en San Antonio, Texas, con un navegador instalado en un IBM con sistema VM/CMS. En 1992 aparece uno de los primeros navegadores (browsers) de Web en el CERN, se llamaba *viola* y funcionaba en

modo texto. A principios del 1993, Marc Andreessen de la universidad de Illinois, junto a otros desarrolladores, trabajó en un proyecto cuyo propósito era crear un navegador capaz de leer las Páginas Web pero no en modo texto, sino en forma gráfica, utilizando las capacidades de hipertexto e hipermedia, creando uno de los navegadores más conocidos el *NCSA Mosaic*, precursor de otros como Netscape, Internet Explorer.

El Mosaic apareció para el público en Abril de 1993, y en octubre de ese mismo año existían 200 Servidores Web utilizando el protocolo HTTP. A partir de 1994 Internet empezó a convertirse en lo que ahora conocemos: una red que permite la transferencia de todo tipo de información: texto, imágenes, dibujos, vídeo, sonido, animaciones creadas por computadora. A partir de 1995 se produce el “Boom Internet” que es considerado como el nacimiento de la Internet comercial. Desde ese momento el crecimiento de la red ha superado todas las expectativas. Este hecho se produce porque es en este año cuando la WWW supera a ftp-data, transformándose en el servicio más popular de la red.

Durante un año y medio, hasta la mitad del 2000, el ámbito económico occidental se vio sacudido por la fiebre de las empresas puntocom, por las inversiones apabullantes en start-up, por el nacimiento de millonarias y fugaces iniciativas virtuales, y por la consideración global de la red como el escenario principal donde la nueva economía interpretaría su papel estelar en los mercados globalizados del nuevo siglo.

Durante los últimos años, Internet ha revolucionado el campo de la informática como un tema de interés nacional. Lo que solía ser una red informática reservada para investigadores, gobiernos e instituciones educacionales se ha convertido en

algo disponible para las empresas e incluso para los usuarios particulares. En el siguiente cuadro[1] se detallan las etapas de la evolución de Internet.

CUADRO N° 1. Etapas de evolución de Internet.

	1975-1992	1993-1997	1998 a la fecha
No. Usuarios (Mill)	< 0.1	Decenas	Centenas
Servicios	Mensajería, FTP	Web	Interactividad, Videoconferencia
Media	Texto	Hipertexto, Audio, Fotos, Multimedia	Vídeo, Animaciones, Realidad Virtual
Velocidad	.4 – 14.4 Kbps	28.8 –128 Kbps	> 1Mbps
Entorno de acceso mayoritario	UNÍX	PC's	Terminal Internet (TV, Consola de Juegos, Agendas de Bolsillo.)
Uso Principal	Académico	Comercial y fuente de información	Relación interpersonal, información, entretenimiento
Implantación geográfica	Norteamérica y Europa	América, Europa, Australia, Sudeste asiático y países en desarrollo	Mundial

Internet es literalmente la red de redes que actualmente enlaza computadoras de todo el mundo. Todas las computadoras están interconectadas de forma que pueden compartir entre ellas archivos de datos, programas. Normalmente, estas redes disponen de una o varias computadoras llamadas servidores que almacenan la información y controlan los periféricos a compartir, y otras llamadas clientes disfrutan de estos servicios.

Un Servidor Web es un programa que corre sobre el servidor que escucha las peticiones HTTP que le llegan y las satisface. Dependiendo del tipo de petición, el Servidor Web buscará una Página Web o bien ejecutará un programa en el servidor. De cualquier modo, siempre devolverá algún tipo de resultado HTML al cliente o navegador que realiza la petición.

[1] Fuente: Documento Tecnologías de Información Turística. http://www.turismo.uma.es

Hoy en día, por muy pequeña que sea una empresa, la mejor forma de relacionarse y promocionar sus productos o servicios, con sus clientes o proveedores es disponer de un Sitio Web o portal Intranet / Internet. En muchos casos las empresas contratan los servicios de otras empresas dedicadas a proveer servicios de esta naturaleza, desde el diseño del sitio, hasta proporcionar el Hosting en un Servidor Web. Otras empresas, desarrollan sus propios sitios y el software adecuado para servir sus páginas, y como es normal, genera un costo adicional a la empresa. Para el desarrollo de un proyecto, siempre se buscan los métodos que permitan obtener la máxima funcionalidad posible por el menor precio, manteniendo además un elevado nivel de calidad.

A la hora de sumar los costos de las computadoras, las licencias de los productos y los desarrollos a medida, y querer encajar los cálculos con el presupuesto disponible puede resultar una inversión un poco elevada. El software libre o de fuentes abiertas, permite reducir los costos, porque eliminan el gasto de las licencias y en muchas ocasiones proporciona un grado de calidad mayor. La prueba es el Servidor Web Apache, que dispone de una cuota de mercado del 60% y ha sido integrado dentro de diversos paquetes comerciales. El problema muchas veces con el software libre es el soporte técnico, ya que por su naturaleza gratuita, no se cuenta con la garantía que ofrecen los productos no gratuitos. Se puede encontrar el soporte adecuado en el medio, pero siempre incurre en gastos que van acorde a la complejidad del proyecto.

1.2 SITUACION PROBLEMÁTICA

La incorporación de las tecnologías de información y de las comunicaciones en las empresas y en la sociedad esta influyendo de forma significativa en el entorno competitivo actual. El impacto de las nuevas tecnologías cada vez va adquiriendo mayor importancia y se integra en todos los aspectos que involucra el desempeño de las labores de una empresa. Como consecuencia de esto, será indispensable que estás extiendan sus operaciones a nuevos mercados y que busquen nuevas estrategias para aumentar su competitividad. Internet es una de las alternativas para conseguir estas metas.

Internet es un medio de comunicación que influye en gran manera en el modo de vida de las personas y las sociedades, esto como consecuencia de un fuerte proceso de desarrollo tecnológico que esta cambiando el panorama global de las comunicaciones, debido a las grandes redes mundiales de datos que pretenden integrar el teléfono, computadoras, televisión, vídeo. Esto abre un mundo con enormes posibilidades y utilidades, para las personas que desean obtener información desde sus hogares, oficinas o Universidades así como a las empresas que buscan completar su ciclo de ventas.

Conscientes de esto, algunas empresas se han integrado al mundo de Internet instalando Servidores Web para publicar Páginas Web, ya sea para publicidad o para gestión de ventas y servicios. Sin embargo, la mayoría de Empresas Salvadoreñas, no cuentan con un espacio en el Internet, debido a que no conocen los beneficios y ventajas que este proporciona, o porque no cuentan con alternativas acordes a sus posibilidades económicas y tecnológicas. Esto disminuye su competitividad en el mercado y su desarrollo económico en gran medida, impidiendo que puedan expandir sus operaciones, dejando fuera de su alcance un mercado potencial que podría ubicarse tanto dentro como fuera del país.

En este sentido es necesario proporcionar alternativas, encaminadas a construir o mejorar la infraestructura tecnológica de las empresas de El Salvador, para que a través de un análisis y una evaluación de las mismas, puedan elegir la que más se adapte a sus necesidades informáticas, y a sus capacidades de inversión y mantenimiento. De este modo se contribuirá a aumentar su competitividad en el mercado, impulsando así el desarrollo económico, social y tecnológico del país.

Por esta razón se concibe la idea de diseñar y elaborar software para servir páginas en Internet, a través de una estructura física como un servidor conectado a Internet, de tal manera que la mediana empresa salvadoreña, pueda tener una amplia gama de alternativas, entre las cuales pueda seleccionar la que más se adecue a sus capacidades de inversión y a su presupuesto.

1.3 JUSTIFICACIÓN

La publicidad en Internet a través del mundo es una de las herramientas de mayor difusión y utilidad, ya que permite la publicación de cantidades de información que en otros medios, como la radio, televisión o periódico, no sería posible, por el alto costo de publicación. A continuación se enumeran algunos beneficios que las empresas adquieren al tener un sitio en Internet.

- Interactividad
- Marketing y Publicidad
- Reducción de Costos
- Ampliación de Mercado
- Información Actualizada

En el cuadro N° 2[2] se observan estadísticas sobre las razones por las cuales los empresarios deciden hacer presencia en Internet.

CUADRO N° 2. Razones para tener presencia en Internet.

Con el propósito de hacer publicidad y mercadeo	65 %
Ser percibido como líder	62%
Mejorar el Servicio al Cliente	53%
Proveer métodos alternativos de compra	41 %
Mantenerse competitivo	30%
Aumentar las ganancias	20%
Reducir el tiempo en el proceso de ordenes	9%
Vender productos directamente a través de Internet	8%
Empezar a mejorar el proceso de ordenes a través de e-mail	7%
Rebajar el costo promedio de venta	4%
Reclutar empleados	1%

[2] http://www.ExitoExportador.com, tomado de The Grant Thornton Survey of American Manufaturers

Internet y las tecnologías de información son las nuevas opciones generadoras de eficiencia en la introducción al mercado global, al incrementar los ingresos y potenciar la satisfacción de las personas con el acceso a mejores comunicaciones, bienes y servicios. Por la importancia que tiene la economía de Red para América Latina, y para El Salvador se dan a conocer las posibilidades estratégicas y las realidades comerciales de ese sector.

En el cuadro N° 3 se destacan las Estadísticas del Uso de Internet en América Central y El Salvador. Esto informa sobre las expectativas levantadas por el medio, dando una idea de la fortaleza de los diversos sectores informáticos y de telecomunicaciones.

CUADRO N° 3. Usuarios de Internet en América Central.

AMERICA CENTRAL	(%)	Población Actual (2002)	Usuarios, dato más reciente	Usuarios, año 2000	Crecimiento (2000-2002)	Penetración (% usuarios)
Belice	2.2 %	251,600	18,000	15,000	20.0 %	7.2 %
Costa Rica	46.2 %	4,050,100	384,000	250,000	53.6 %	9.5 %
El Salvador	6.0 %	6,076,800	50,000	40,000	25.0 %	0.8 %
Guatemala	24.0 %	13,730,900	200,000	65,000	207.7 %	1.5 %
Honduras	4.8 %	6,438,100	40,000	40,000	0.0 %	0.6 %
Nicaragua	6.0 %	5,579,400	50,000	50,000	0.0 %	0.9 %
Panamá	10.8 %	2,939,800	90,000	45,000	100.0 %	3.1 %
Tot. Am. Central	**100 %**	**39,066,700**	**832,000**	**505,000**	**64.8 %**	**2.1 %**

[3]Datos actuales a julio 31 del 2,002. Datos actuales de Gazetteer.de se tomaron como base para las cifras de población actual. Diversas fuentes locales y datos de Nielsen-NetRatings corresponden a los datos más recientes del número de usuarios. La determinación de las cifras de crecimiento se hizo comparando el

[3] Fuente: http://www.ExitoExportador.com

dato actual de usuarios con el dato del año 2,000, tomado de las estadísticas del ITU.

Como se puede observar el Internet en El Salvador ha mostrado ritmos de crecimiento aceptables, que han llevado a la red a constituirse en una oportunidad de negocios, para que las empresas puedan desplegar sus fortalezas competitivas, incrementar su capacidad de producción y satisfacer sus necesidades.

Las empresas Salvadoreñas tienen posibilidades estratégicas para gozar de los beneficios que el Internet proporciona a través de los Sitios Web, si se proveen alternativas funcionales y que reduzcan los costos a la hora de su implementación, es aquí donde tienen el papel principal los Servidores Web. Los Servidores Web permiten a los clientes compartir datos, documentos y multimedia en formato web para uso tanto de una Intranet, como en Internet.

En este sentido, con el desarrollo de este proyecto, se presenta una alternativa más para las empresas salvadoreñas, en especial para la Mediana Empresa, que garantice su presencia en Internet y que le permita reducir sus costos de inversión, ofreciendo un Prototipo de Software para servir páginas en Internet, acorde a las necesidades que una empresa de este tipo pueda tener.

1.4 ENUNCIADO DEL PROBLEMA

La falta de alternativas de software, acordes a las capacidades tecnológicas y económicas, de las Medianas Empresas de El Salvador que les impide tener presencia en Internet e innovar sus procesos a través de Sitios Web.

1.5 DELIMITACION

1.5.1 DELIMITACION ESPACIAL

Se tomarán para este estudio, todas aquellas empresas categorizadas como Medianas Empresas en El Salvador, que se encuentren legalmente registradas en "La Dirección General de Estadísticas y Censos" (DIGESTYC) hasta el año 2,000.

1.5.2 DELIMITACION GEOGRAFICA

Para la investigación se visitaran las Medianas Empresas ubicadas en la zona paracentral, específicamente los departamentos de San Salvador y La Libertad.

1.5.3 DELIMITACION ESPECIFICA

Las empresas que serán objeto de estudio se encuentran ubicadas en la Zona Industrial de Merliot, Zona Industrial de Santa Elena, Zona Industrial Plan de la Laguna y Empresas ubicadas sobre la Avenida Roosvelt.

1.5.4 DELIMITACION TEMPORAL

El tiempo estimado para la realización del proyecto es de nueve meses, comenzando en el mes de Marzo y terminando en Diciembre de 2003.

1.6 OBJETIVOS

1.6.1 OBJETIVO GENERAL

Diseñar y elaborar un Software prototipo para la creación de un Servidor Web, orientado a la Mediana empresa de El Salvador, el cuál proporcione una herramienta útil para la gestión de sus operaciones.

1.6.2 OBJETIVOS ESPECÍFICOS

1. Establecer el marco teórico conceptual relacionado a la problemática ocasionada por la falta de utilización de tecnología apropiada de informática, para la utilización de Internet como medio de publicidad y herramienta en el desarrollo de actividades laborales de la Mediana Empresa en El Salvador.

2. Determinar mediante una investigación de campo, la situación tecnológica actual de las Medianas Empresas de El Salvador, para conocer sus tendencias y sus capacidades informáticas.

3. Diseñar y Elaborar el Software Prototipo para la creación de un Servidor Web, orientado a la Mediana Empresa de El Salvador.

4. Diseñar un mecanismo por medio del cual, el Software Prototipo, pueda autorecuperarse en caso de fallas debidas a factores externos.

1.7 ALCANCES Y LIMITACIONES

1.7.1 ALCANCES

1. Manejo de requerimientos sobre el protocolo HTTP, a través de una Línea dedicada.
2. Se incluye una gestión de autenticación para impedir el acceso al servidor de usuarios no deseados, y la opción de sólo atender peticiones de usuarios que estén registrados. Los usuarios registrados tienen un nombre de usuario y una contraseña y se almacenan en un fichero o directorio especifico.
3. El Software del Servidor incluye un archivo HTML que se visualizará en el navegador en caso de error.
4. El Servidor Web manejará de manera eficiente la presentación de Páginas Web con imágenes y texto.
5. Diseño de un prototipo de un Sitio Web y Una Página Web para evaluar y demostrar la operabilidad del software del Servidor. El Prototipo del sitio ejemplo para efectos de demostración, está orientado a una Empresa de ventas de suministros para computadoras.
6. En cuanto a la seguridad del tráfico de información que entra y sale de la red, únicamente se contempla la inclusión de Firewall basado en las herramientas de administración de Firewalls del Sistema Operativo a utilizar (Linux), a nivel teórico. Entre la documentación se incluirá un apartado para explicar la configuración del mismo. Por otra parte, algunos Proveedores de Servicios de Internet (ISP) proporcionan el servicio de Firewall básico para redes, esto se toma en cuenta a la hora de recomendar uno de ellos para la adquisición de la línea dedicada.
7. Diseño a nivel teórico, de un mecanismo de recuperación del Servidor en caso de fallas debidas a factores externos.

1.7.2 LIMITACIONES

1. El Diseño del Servidor esta orientado a cubrir los requerimientos de una Mediana Empresa, en lo que se refiere a cantidad de accesos y cantidad de información, por lo que su funcionamiento no será óptimo para empresas que tienen un flujo de accesos mayor.
2. No se profundiza sobre el soporte para software de seguridad que encripte los mensajes que se envían por la red, tales como los protocolos SSH (Secure Shell), SSL (Secure Socket Layer) , TLS(Transport Layer Secure), Kerberos, sin embargo entre la documentación se incluirán recomendaciones para su implementación.
3. El Diseño del software no incluye soporte para CGI, o para interpretes de otros lenguajes de programación diferentes a Perl.
4. El Diseño del servidor está hecho para trabajar sólo bajo la plataforma Linux, distribución Red Hat 8.0 (Servidor).
5. El mecanismo mediante el cuál el Servidor Web podrá recuperarse por fallas debidas a factores externos, se realizará a nivel de diseño. Existen limitantes económicas y de tiempo, para llevar a cabo la implementación del mecanismo a diseñar, ya que se requiere una inversión en hardware que para el desarrollo del presente proyecto, no esta contemplada en el presupuesto del trabajo de graduación. Por otra parte, el tema de la autorecuperación, suele ser bastante amplio, por lo que no se profundizará en el mismo ya que no se cuenta con el tiempo necesario para realizar las implementaciones y las pruebas correspondientes. Sin embargo, el diseño podrá servir como pauta para que otros grupos de trabajo, lo puedan retomar y realizar las pruebas correspondientes, además de que podrán ampliarlo o mejorarlo.
6. Existe poca información técnica disponible, sobre la construcción o el diseño de Servidores Web, la información disponible es más que todo de tipo comercial.

1.8 BENEFICIOS

Con el Diseño del Software Prototipo para la creación de un Servidor Web, se proporcionará una alternativa más a las Empresas Salvadoreñas, en especial a la Mediana empresa, que garantice su presencia en Internet y que les permita gozar de los beneficios que Internet proporciona a través de la publicidad, marketing, gestión de ventas y servicios, ofreciendo un Software para servir páginas en Internet, que les ayude a reducir sus costos de inversión a través del uso de herramientas de libre distribución, y que satisfaga los requerimientos y las necesidades informáticas que las empresa de este tipo puedan tener. A continuación se enumeran los Beneficios que se esperan obtener con el desarrollo de este proyecto.

a) Bajo Costo de Instalación y Adquisición

El diseño del software estará hecho para trabajar bajo una plataforma de libre distribución, ahorrando los costos de inversión en licencias, así mismo, el software podrá ser comercializado de manera accesible, a través de la universidad. Por otra parte el Sistema Operativo, incorpora una lista de servicios que se pueden configurar para aumentar la eficiencia de el Servidor mismo, entre ellas la de utilidad para el proyecto es la herramienta iptables[4], que permite agregar la funcionalidad de contrafuegos o firewall a un servidor. De igual manera este servicio contribuye a reducir los costos de instalación y adquisición, ya que no será necesario invertir en software especializado y de alto costo para el filtrado de paquetes.

b) Reducción del Costo de Mantenimiento

El diseño del software del servidor será lo más flexible posible, enfocándose en hacer un servidor pequeño pero eficiente que pueda autorecuperarse en caso de

[4] Es una Herramienta de Linux, para la Administración de Firewalls y filtros de paquetes de IP

fallas debidas a factores externos. Esto minimizará la necesidad de contratar personal adicional o innecesario para vigilancia y mantenimiento del mismo.

c) Facilidad de operación y configuración

El Software del Servidor Web incluirá una interfaz amigable, que facilitará la interacción con el usuario o la persona que lo administre. Los procesos que incluirá serán diseñados lo más eficiente posible, de manera que el usuario tenga que complicarse lo menos posible a la hora de interactuar con el mismo. Esto ayudara a reducir los costos de mantenimiento, ya que no se requerirá personal especializado para la operación o gestión del servidor. Por otra parte, se incluirá entre la documentación una guía que cubra todos los aspectos relacionados con la configuración del software y para la configuración del Firewall.

d) Independencia de un proveedor para futuras adaptaciones o mejoras

Esto gracias, a que el diseño del software del servidor estará bajo el esquema de código abierto y a la comercialización del software de manera accesible. La Persona o Empresa que lo adquiera, podrá adaptarlo, actualizarlo o modificarlo, según sus necesidades informáticas.

e) Manejo de Requerimientos sobre el Protocolo HTTP

El Software de Servidor Web manejará eficientemente la presentación de Páginas Web con imágenes y texto, así como Páginas Web que representen formularios con información determinada.

f) Servidor Seguro

Gracias a que el servidor incorporará la opción de atender sólo las peticiones de usuarios que estén registrados se obtendrá un servidor seguro. De la misma forma se impedirá el acceso a usuarios no deseados. Por otra parte se podrán recomendar el uso e implementación de software adecuado para el transporte

seguro de los datos por la red, tal como los protocolos SSL, SSH, Kerberos. De igual manera se proporcionará la documentación y recomendaciones necesarias para la configuración del Firewall.

1.9. MARCO TEÓRICO

1.9.1 INTERNET

1.9.1.1 CONCEPTO

Internet es una Red de Redes porque está hecha a base de unir muchas redes locales de computadoras, es decir de unas pocas computadoras en un mismo edificio o empresa. Además, ésta es La Red de Redes porque es la más grande. Entre las funciones más relevantes del Internet se tienen: Compartir información, comercio electrónico, publicidad, entretenimiento, búsqueda de información.

1.9.1.2 ESTRUCTURA DE LA RED INTERNET

Internet es una red de alcance mundial que une una gran cantidad de redes grandes de computadoras. Esto afecta al usuario de Internet, puesto que le permite contactar con gente y computadoras de todo el mundo desde cualquier lugar donde él se encuentre. Internet funciona bajo el esquema Cliente/Servidor, lo que significa que en la Red hay computadoras Servidores que dan una información concreta en el momento que se solicite, y por otro lado están las computadoras que piden dicha información, los llamados Clientes. Existe una gran variedad de lenguajes que usan las computadoras para comunicarse por Internet. Estos lenguajes se llaman Protocolos. Se ha establecido que en Internet, toda la información ha de ser transmitida mediante el Protocolo TCP/IP.

1.9.1.3 PROTOCOLO TCP/IP

TCP/IP son las siglas de "Transfer Control Protocol / Internet Protocol". Éste es el lenguaje establecido para la Red Internet. Antes de su creación, este protocolo tuvo mucho éxito en el campo de las grandes computadoras máquinas UNIX. El protocolo TCP/IP se ha establecido como estándar en la red Internet debido a las ventajas y características que presenta.

La principal característica del TCP/IP es que establece la comunicación por medio de paquetes de información. Cuando una computadora quiere enviar un fichero de datos, lo primero que hace es partirlo en trozos pequeños (alrededor de unos 4 Kb) y posteriormente envía cada trozo por separado. Cada paquete de información contiene la dirección en la Red donde ha de llegar, y también la dirección de remite, por si hay que recibir respuesta. Los paquetes viajan por la Red de forma independiente. Entre dos puntos de la Red suele haber muchos caminos posibles. Cada paquete escoge uno dependiendo de factores como saturación de las rutas o posibles atascos.

Otra notable y muy positiva consecuencia del uso del TCP/IP es que admite la posibilidad de que algún paquete de información se pierda por el camino. Al transmitir el protocolo TCP, compensa la perdida de paquetes con un esquema de retransmisión. Cuando TCP recibe datos, devuelve un acuse de recibo (ACK) al transmisor. Cuando los envía, inicia un cronometro, si este expira antes de llegar el acuse de recibo, envía de nuevo los datos. De esta manera se garantiza que no existirá la duplicación ni la perdida de paquetes.

1.9.1.4 DIRECCIONES IP Y NOMBRES DE DOMINIO

Cada computadora que se conecta a Internet se identifica por medio de una dirección IP. Ésta se compone de 4 números comprendidos entre el 0 y el 255 ambos inclusive y separados por puntos. Así, por ejemplo una dirección IP podría ser: 156.215.11.35. No está permitido que coexistan en la Red dos computadoras distintas con la misma dirección, puesto que de ser así, la información solicitada por una de las computadoras no sabría a cual de ellos dirigirse. Cada número de la dirección IP indica una subred de Internet. Hay 4 números en la dirección, lo que quiere decir que hay 4 niveles de profundidad en la distribución jerárquica de la Red Internet.

Esta distribución jerárquica de la Red Internet, permite enviar y recibir rápidamente paquetes de información entre dos computadoras conectadas en cualquier parte del Mundo a Internet, y desde cualquier subred a la que pertenezcan.

Un usuario de Internet, no necesita conocer ninguna de estas direcciones IP. Las manejan las computadoras en sus comunicaciones por medio del Protocolo TCP/IP de manera invisible para el usuario. Sin embargo, se necesita nombrar de alguna manera las computadoras de Internet, para poder elegir a cual pedir información. Esto se logra por medio de los Nombres de Dominio.

Los nombres de dominio, son la traducción para las personas de las direcciones IP, las cuales son útiles sólo para las computadoras. Así por ejemplo, yahoo.com es un nombre de dominio. Como se puede ver, los nombres de dominio son palabras separadas por puntos, en vez de números en el caso de las direcciones IP. Estas palabras pueden darnos idea de la computadora a la que nos estamos refiriendo. Por medio de lo que se llaman, "Servidores de Nombres de Dominio (DNS)", Internet es capaz de averiguar la dirección IP de una computadora a partir de su nombre de dominio.

1.9.1.5 CONEXIÓN A LA RED

Para poderse conectar a Internet se necesitan cuatro cosas: una computadora, un módem (Modulador – Demodulador) o DTU(Digital Terminal Unit), un programa que efectúe el enlace con el proveedor, y un programa para navegar por la red.

1.9.1.6 PROVEEDORES DE INTERNET

Un Proveedor Internet permite conectar una computadora a la Red Internet. No se puede conectar directamente, puesto que las líneas de comunicaciones que forman Internet en sí, sólo las pueden manejar las grandes empresas de las

telecomunicaciones a nivel Mundial: Telefónica, British Telecom. Un dato importante a tener en cuenta es lo que se llama el Ancho de Banda del Proveedor. El ancho de banda mide la capacidad de transmitir datos entre Internet y los usuarios. Es importante que un Proveedor tenga el máximo ancho de banda posible para que un módem alcance la máxima velocidad posible, y sea posible recoger la información de Internet solicitada en el menor tiempo posible, y de esta manera ahorrar en tiempo de operación.

1.9.1.7 SEGURIDAD EN INTERNET Y FIREWALL

La seguridad ha sido el principal aspecto a tratar cuando una organización desea conectar su red privada al Internet. Sin tomar en cuenta el tipo de negocio, se ha incrementado el número de usuarios de redes privadas por la demanda del acceso a los servicios de Internet tal es el caso del World Wide Web (WWW), Internet Mail (e-mail), Telnet, y File Transfer Protocol (FTP). Adicionalmente los corporativos buscan las ventajas que ofrecen las páginas en el WWW y los servidores FTP de acceso publico en el Internet.

Los administradores de red tienen que incrementar todo lo relacionado a la seguridad de sus sistemas, debido a que se expone la organización privada de los datos así como la infraestructura de la red a los Expertos de Internet (Internet Crakers). Para superar estos temores y proveer el nivel de protección requerida, la organización necesita seguir una política de seguridad para prevenir el acceso no autorizado de usuarios a los recursos propios de la red privada, y protegerse contra la exportación privada de información. Todavía, aun si una organización no esta conectada al Internet, esta debería establecer una política de seguridad interna para administrar el acceso de usuarios a porciones de red y proteger sensitivamente la información secreta.

Un Firewall en Internet es un sistema o grupo de sistemas que impone una política de seguridad entre la organización de red privada y el Internet. El firewall determina cual de los servicios de red pueden ser accesados dentro de esta por los que están fuera, es decir quien puede entrar para utilizar los recursos de red pertenecientes a la organización. Para que un firewall sea efectivo, todo trafico de información a través del Internet deberá pasar a través del mismo donde podrá ser inspeccionada la información. El firewall podrá únicamente autorizar el paso del trafico, y el mismo podrá ser inmune a la penetración. desafortunadamente, este sistema no puede ofrecer protección alguna una vez que el agresor lo traspasa o permanece entorno a este.

1.9.1.8 PÁGINAS WEB

Es un documento o archivo, al igual que un archivo de Word, Excel o PowerPoint. La diferencia es que utiliza un lenguaje llamado Hiper Texto el cual le permite a estos archivos contener otros, como los antes mencionados. La idea central de las Páginas Web consiste en que un grupo de computadoras están conectadas y pueden intercambiar archivos específicos con las demás computadoras. Pero la red es tan grande que sería muy difícil conectar a todas las computadoras de todo el mundo directamente. Por esto existen los servidores.

Un grupo de computadoras están conectadas a un servidor y éste a el resto de servidores de todo el mundo. Las Páginas Web son archivos almacenados en la memoria de estos servidores (la mayoría de los dueños de los servidores cobran por esto, pero existen algunos que dan éste servicio gratis como geocities, angelfire, xoom, go.to, latinmail). El servidor además de almacenar las páginas administra el acceso a ellas.

1.9.1.9 HTTP (HYPER TEXT TRANSFER PROTOCOL)

El Protocolo de Transferencia de HiperTexto (Hypertext Transfer Protocol) es un protocolo cliente-servidor que se basa en sencillas operaciones de solicitud/respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado. Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto Web (documento HTML, fichero multimedia o aplicación CGI) es conocido por su URL.

HTTP se soporta sobre TCP, que es el protocolo de transporte de TCP/IP para comunicaciones orientadas a conexión y utiliza el esquema cliente-servidor. El servidor HTTP permanece activo esperando conexiones de los clientes HTTP. Una vez solicitada una conexión, el servidor responde dejando la posibilidad de que clientes logren conexiones simultáneamente. Las principales características del protocolo HTTP son:

- Toda la comunicación entre los clientes y servidores se realiza a partir de caracteres de 8 bits. De esta forma, se puede transmitir cualquier tipo de documento, respetando su formato original.
- Permite la transferencia de objetos multimedia. El contenido de cada objeto intercambiado está identificado por su clasificación MIME.
- Existen tres verbos básicos que un cliente puede utilizar para dialogar con el servidor: GET, para recoger un objeto, POST, para enviar información al servidor y HEAD, para solicitar las características de un objeto.
- Cada operación HTTP implica una conexión con el servidor, que es liberada al término de la misma. Es decir, en una operación se puede recoger un único objeto.

- No mantiene estado. Cada petición de un cliente a un servidor no es influida por las transacciones anteriores. El servidor trata cada petición como una operación totalmente independiente del resto.
- Cada objeto al que se aplican los verbos del protocolo está identificado a través de la información de situación del final de la URL.

Estructura de los mensajes HTTP

El diálogo con los servidores HTTP se establece a través de mensajes formados por líneas de texto, cada una de las cuales contiene los diferentes comandos y opciones del protocolo. Sólo existen dos tipos de mensajes, uno para realizar peticiones y otro para devolver la correspondiente respuesta. La estructura general de los dos tipos de mensajes se puede ver en el siguiente esquema.

Esquema N° 1. Estructura general de los dos tipos de mensajes HTTP.

Mensaje de solicitud	Mensaje de respuesta
Comando HTTP + parámetros	Resultado de la solicitud
Cabeceras del requerimiento	Cabeceras de la respuesta
(línea en blanco)	(línea en blanco)
Información opcional	Información opcional

La primera línea del mensaje de solicitud contiene el comando que se solicita al servidor HTTP, mientras que en la respuesta contiene el resultado de la operación, un código numérico que permite conocer el éxito o fracaso de la operación. Después aparece, para ambos tipos de mensajes, un conjunto de cabeceras (unas obligatorias y otras opcionales), que condicionan y matizan el funcionamiento del protocolo.

La separación entre cada línea del mensaje se realiza con un par CR-LF (retorno de carro más nueva línea). El final de las cabeceras se indica con una línea en blanco, tras la cual se pueden incluir los datos transportados por el protocolo, por ejemplo, el documento HTML que devuelve un servidor o el contenido de un formulario que envía un cliente .

Comandos del Protocolo HTTP

Los comandos o verbos de HTTP representan las diferentes operaciones que se pueden solicitar a un servidor HTTP. El formato general de un comando es el siguiente:

Nombre del comando	Objeto sobre el que se aplica	Versión de HTTP utilizada

Cada comando actúa sobre un objeto del servidor, normalmente un fichero o aplicación, que se toma de la URL de activación. La última parte de esta URL, que representa la dirección de un objeto dentro de un servidor HTTP, es el parámetro sobre el que se aplica el comando. Se compone de una serie de nombres de directorios y ficheros, además de parámetros opcionales para las aplicaciones CGI. El estándar HTTP/1.0 recoge únicamente tres comandos, que representan las operaciones de recepción y envío de información y chequeo de estado:

- GET. Recoge cualquier tipo de información del servidor. Se utiliza siempre que se pulsa sobre un enlace o se teclea directamente a una URL. Como resultado, el servidor HTTP envía el documento correspondiente a la URL seleccionada, o bien activa un módulo CGI, que generará a su vez la información de retorno.
- HEAD. Solicita información sobre un objeto (fichero): tamaño, tipo, fecha de modificación. Es utilizado por los gestores de cachés de páginas o los servidores proxy, para conocer cuándo es necesario actualizar la copia que se mantiene de un fichero.

- POST. Sirve para enviar información al servidor, por ejemplo los datos contenidos en un formulario. El servidor pasará esta información a un proceso encargado de su tratamiento. La operación que se realiza con la información proporcionada depende de la URL utilizada. Se utiliza, sobre todo, en los formularios.

Un cliente Web selecciona automáticamente los comandos HTTP necesarios para recoger la información requerida por el usuario. Así, ante la activación de un enlace, siempre se ejecuta una operación GET para recoger el documento correspondiente. El envío del contenido de un formulario utiliza GET o POST, en función del atributo de *form method*. Además, si el cliente Web tiene un caché de páginas recientemente visitadas, puede utilizar HEAD para comprobar la última fecha de modificación de un fichero, antes de traer una nueva copia del mismo.

Las cabeceras HTTP

Son un conjunto de variables que se incluyen en los mensajes HTTP, para modificar su comportamiento o incluir información de interés. En función de su nombre, pueden aparecer en los requerimientos de un cliente, en las respuestas del servidor o en ambos tipos de mensajes. El formato general de una cabecera es:

Nombre de la variable : Cadena ASCII con su valor

Los nombres de variables se pueden escribir con cualquier combinación de mayúsculas y minúsculas. Además, se debe incluir un espacio en blanco entre el signo : y su valor. En caso de que el valor de una variable ocupe varias líneas, éstas deberán comenzar, al menos, con un espacio en blanco o un tabulador.

Cabeceras comunes para peticiones y respuestas

- Content-Type: descripción MIME de la información contenida en este mensaje. Es la referencia que utilizan las aplicaciones Web para dar el correcto tratamiento a los datos que reciben.
- Content-Length: longitud en bytes de los datos enviados, expresado en base decimal.
- Content-Encoding: formato de codificación de los datos enviados en este mensaje. Sirve, por ejemplo, para enviar datos comprimidos (x-gzip o x-compress) o encriptados.
- Date: fecha local de la operación. Las fechas deben incluir la zona horaria en que reside el sistema que genera la operación. No existe un formato único en las fechas; incluso es posible encontrar casos en los que no se dispone de la zona horaria correspondiente, con los problemas de sincronización que esto produce. Los formatos de fecha a emplear están recogidos en los RFC 1036 y 1123.
- Pragma: permite incluir información variada relacionada con el protocolo HTTP en el requerimiento o respuesta que se está realizando. Por ejemplo, un cliente envía un Pragma: no-cache para informar de que desea una copia nueva del recurso especificado.

Cabeceras sólo para peticiones del cliente

- Accept: campo opcional que contiene una lista de tipos MIME aceptados por el cliente. Se pueden utilizar * para indicar rangos de tipos de datos; tipo/* indica todos los subtipos de un determinado medio, mientras que */* representa a cualquier tipo de dato disponible.
- Authorization: clave de acceso que envía un cliente para acceder a un recurso de uso protegido o limitado. La información incluye el formato de autorización

empleado, seguido de la clave de acceso propiamente dicha. La explicación se incluye más adelante.

- From: campo opcional que contiene la dirección de correo electrónico del usuario del cliente Web que realiza el acceso.
- If-Modified-Since: permite realizar operaciones GET condicionales, en función de si la fecha de modificación del objeto requerido es anterior o posterior a la fecha proporcionada. Puede ser utilizada por los sistemas de almacenamiento temporal de páginas. Es equivalente a realizar un HEAD seguido de un GET normal.
- Referer: contiene la URL del documento desde donde se ha activado este enlace. De esta forma, un servidor puede informar al creador de ese documento de cambios o actualizaciones en los enlaces que contiene. No todos los clientes lo envían.
- User-agent: cadena que identifica el tipo y versión del cliente que realiza la petición. Por ejemplo, los navegadores de Netscape envían cadenas del tipo User-Agent: Mozilla/3.0 (WinNT; I).

Cabeceras sólo para respuestas del servidor HTTP

- Allow: informa de los comandos HTTP opcionales que se pueden aplicar sobre el objeto al que se refiere esta respuesta. Por ejemplo, Allow: GET, POST.
- Expires: fecha de expiración del objeto enviado. Los sistemas de cache deben descartar las posibles copias del objeto pasada esta fecha.
- Last-modified: fecha local de modificación del objeto devuelto.
- Location: informa sobre la dirección exacta del recurso al que se ha accedido. Cuando el servidor proporciona un código de respuesta de la serie 3xx, este parámetro contiene la URL necesaria para accesos posteriores a este recurso.
- Server: cadena que identifica el tipo y versión del servidor HTTP. Por ejemplo, Server: NCSA 1.4.

- WWW-Autenticate: cuando se accede a un recurso protegido o de acceso restringido, el servidor devuelve un código de estado 401, y utiliza este campo para informar de los modelos de autenticación válidos para acceder a este recurso.

Códigos de estado del servidor

Ante cada transacción con un servidor HTTP, éste devuelve un código numérico que informa sobre el resultado de la operación, como primera línea del mensaje de respuesta. Estos códigos aparecen en algunos casos en la pantalla del cliente, cuando se produce un error. El formato de la línea de estado es:

Versión de protocolo HTTP utilizada	Código numérico de estado (tres dígitos)	Descripción del código numérico

Existen cinco categorías de mensajes de estado, organizadas por el primer dígito del código numérico de la respuesta:

- 1xx : mensajes informativos. Por ahora (en HTTP/1.0) no se utilizan, y están reservados para un futuro uso.
- 2xx : mensajes asociados con operaciones realizadas correctamente.
- 3xx : mensajes de redirección, que informan de operaciones complementarias que se deben realizar para finalizar la operación.
- 4xx : errores del cliente; el requerimiento contiene algún error, o no puede ser realizado.
- 5xx : errores del servidor, que no ha podido llevar a cabo una solicitud.

1.9.1.10 TIPOS MIME

El protocolo HTML fue diseñado para transportar por red ficheros en formato ASCII, compuestos por texto plano. Con el progreso de las tecnologías y con la inclusión de diferentes tipos de ficheros no ASCII en las aplicaciones por Internet, surgió la necesidad de transformar estos formatos a tipo ASCII, para su correcta recepción en el navegador web. Este problema se produjo inicialmente en las aplicaciones de correo electrónico, cuando se necesitó enviar por mail ficheros no formados por texto plano, y por tanto, no compatibles con los juegos de caracteres permitidos.

Para solucionar este problema el Internet Engineering Task Force (IETF) creó en 1992 los tipos Mime (Multipurpose Internet Mail Extensions), especificaciones para dar formato a mensajes no-ASCII, de forma que pudieran ser enviados por Internet e interpretados correctamente por los programas de correo locales.

Fue tan importante la ampliación que se surgió con los tipos Mime al correo, que pronto se aplicaron también a los documentos web, lo que permitió incluir en las páginas HTML ficheros varios, que dieron nueva vida a la web.

Los tipos MIME especifican los tipos de datos que los archivos contienen. MIME adjunta a cada fichero un archivo de cabecera donde se indica el tipo y el subtipo del contenido de los datos del mismo. Gracias a esta información, tanto el servidor como el navegador pueden manejar y presentar los archivos correctamente.

Los navegadores web traen por defecto configurados una serie de tipos Mime, de tal forma que sabe cómo interpretar y ejecutar los ficheros definidos mediante estos tipos, asociando en una base de datos interna los tríos extensión fichero, tipo Mime, aplicación necesaria. Este es el motivo por el que no es necesario declarar manualmente el tipo Mime asociado a una imagen GIF, ya que el

navegador viene configurado para conocer ese tipo de ficheros y saber cómo abrirlo.

1.9.2 SERVIDOR

1.9.2.1 CONCEPTO

Un servidor es una computadora con grandes capacidades físicas y una capacidad de almacenamiento de datos alta, que contiene información que puede ser consultada por usuarios. Por el contrario, un cliente es una computadora que no está presentando información, sino que la va buscando; es decir, las computadoras cliente se conectan a los servidores para obtener información. Los servidores han de estar conectados permanentemente a Internet, pues en caso contrario, alguien intentaría acceder a ellos y no los encontraría.

1.9.2.2 TIPOS DE SERVIDORES

Existen muchos tipos de servidores, cada uno dedicado a funciones diferentes y cada uno de los cuales es capaz de proporcionar un determinado servicio. Los más importantes son:

- *Servidor de correo*. Una computadora donde se guardan todos los mensajes de correo, en espera de que se conecte el usuario al que van dirigidos y que los recoja.

- *Servidor de news*. Una computadora que contiene las news, es decir, los mensajes de los grupos de noticias, para que una persona pueda conectarse y leerlos. Se suele denominar servidor NNTP.

- *Servidor Web*. Una computadora que presenta información según el estándar Web (WWW). Una persona ejecuta un programa web, se conecta a un Servidor

Web y lee su contenido en forma de páginas con colores, texto, fotografías, sonidos, vídeo, animaciones.

- *Servidor FTP*. Una computadora que contiene ficheros que se puede recoger.

- *Servidor IRC-Chat*. Una computadora encargado de permitir a los usuarios mantener conversaciones en tiempo real.

- *Servidor DNS*. Un servidor de nombres de dominio.

Todas las acciones que se realizan sobre Internet se reducen siempre a una sola; conectarse a un servidor y examinar la información que contiene. Internet se puede definir como un conjunto de servidores que ofrecen información a computadoras clientes de todo el mundo.

1.9.2.3 SERVIDOR WEB

Un Servidor Web es un programa que funciona sobre una máquina en red, esperando conexiones del mundo exterior para servir ciertos documentos pedidos a través de un navegador.

Para comunicarse, el servidor y el navegador usan un método de comunicación asíncrono llamado HTTP (Protocolo de Hipertexto), que funciona de la forma siguiente:

1. El usuario arranca el navegador y escribe una dirección URL .
2. El navegador se conecta a un servidor determinado y pide ese documento.
3. El servidor de web maneja la petición y manda la respuesta.
4. Si el documento existe, el servidor lo envía.

5. Si no existe o su acceso no esta permitido, el servidor devuelve un documento con un mensaje de error.
6. El documento enviado como respuesta a esta petición puede contener objetos incrustados. Estos pueden ser URL apuntando a recursos, documentos, imágenes, applets, cadenas de audio/ vídeo, es decir cualquier objeto que se pueda direccionar por HTML.
7. El navegador entonces pide todos aquellos objetos incluidos en la página del servidor, usando los métodos 2 y 3 anteriores, antes de mostrar el contenido de la página.

El método de comunicación se llama asíncrono, porque el navegador envía las diferentes peticiones a la vez, utilizando diferentes canales de comunicación.

1.9.2.4 PLATAFORMAS DE SERVIDORES WEB

Los Servidores Web existen casi para cualquier plataforma computacional en uso hoy en día. Las plataformas más comunes son: Unix, Windows, Linux, Windows NT, MAC. Los primeros Servidores Web se escribieron para las máquinas UNIX. A continuación se describen algunos de ellos.

- WINDOWS 3.1

Este resulta útil al correr un Servidor Web. Por el lado negativo, tiene muchas limitaciones dolorosas como Servidor Web, en primer lugar, se tiene que adquirir un paquete TCP/IP y luego buscar un paquete Servidor Web. Bajo cargas pesadas, en especial con scripts CGI, el servidor se desempeñará como deficiente. Esta plataforma sólo funciona para un Servidor Web de bajo volumen.

- OS/2

OS/2 de IBM es un sistema operativo tipo Windows que puede correr un Servidor Web. En muchos aspectos es similar al Windows NT.

- WINDOWS 95

Este acepta nombre de archivos largo. Si se utiliza únicamente aplicaciones de 32 bits, cuando se emplean aplicaciones de 16 bits, los trabajos multitareas funcionan de manera muy similar a aquélla de Windows 3.1.

- WINDOWS NT – 2000 SERVER

NT es el siguiente escalón de Windows 95, este es aun más robusto, tiene más capacidad multitareas.

- MACINTOSH

Es una plataforma única para un Servidor Web, ya que es la única sin línea de comandos. Los servidores para Mac por lo general son fáciles de instalar.

- LINUX

Hoy en día, la aplicación más frecuente de Linux es como servidor de Internet. Además se pueden configuran servicios necesarios para el correcto funcionamiento del servidor así como servidores de nombres y firewall.

1.9.2.5 TIPOS DE SERVIDORES WEB

1. EMPRESARIAL O CORPORATIVO

Es el conjunto de información, servicios y herramientas que define la presencia de la empresa en Internet. Es un medio de comunicación con sus clientes y prospectos. También puede convertirse en un canal comercial a través del cual proporcionar productos y servicios.

2. TECNOLÓGICO

Es un conjunto de páginas HTML (hypertext Markup Lenguaje) cuyos contenidos están relacionados entre sí, mediante enlaces (hiperenlaces o vínculos). Estas

páginas son almacenadas en un equipo servidor, gestionadas por un software especializado que atiende las peticiones de los usuarios y envía las páginas solicitadas.

1.9.2.6 FUNCIONAMIENTO DE LOS SERVIDORES WEB

El proceso básico: El navegador genera una conexión a un Servidor Web, solicitará una página y la recibe. Esto se ilustra en la siguiente figura.

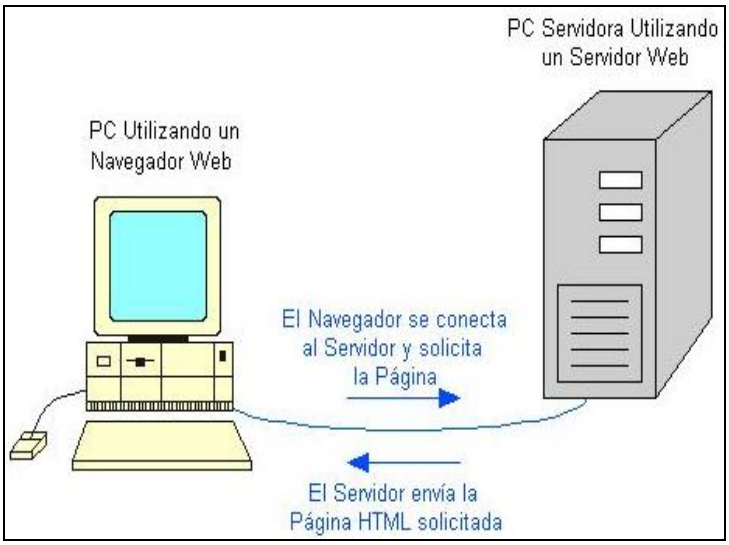


FIGURA N° 1. Proceso básico de un Servidor Web.

Cada vez que un cliente realiza una petición a un servidor, un usuario accede a una URL, seleccionando un enlace de un documento HTML o introduciéndola directamente en el campo *Location* del cliente Web. El cliente Web descodifica la URL, separando sus diferentes partes. Así identifica el protocolo de acceso, la dirección DNS o IP del servidor, el posible puerto opcional (el valor por defecto es 80) y el objeto requerido del servidor. Se abre una conexión TCP/IP con el servidor, llamando al puerto TCP correspondiente. Se realiza la petición. Para ello, se envía el comando necesario (GET, POST, HEAD) la dirección del objeto requerido (el contenido de la URL que sigue a la dirección del servidor), la versión

del protocolo HTTP empleada y un conjunto variable de información, que incluye datos sobre las capacidades del navegador, datos opcionales para el servidor. El servidor devuelve la respuesta al cliente. Consiste en un código de estado y el tipo de dato MIME de la información de retorno, seguido de la propia información. Finalmente, se cierra la conexión TCP.

Este proceso se repite en cada acceso al servidor HTTP. Por ejemplo, si se recoge un documento HTML en cuyo interior están insertadas cuatro imágenes, el proceso anterior se repite cinco veces, una para el documento HTML y cuatro para las imágenes.

1.9.2.7 SERVIDORES WEB MÁS POPULARES

a) NCSA HTTPD: Tiene un buen desempeño y un agradable conjunto de características, su falla principal es la falta para protocolos de encriptación Web.

b) APACHE: es un derivado muy popular del servidor NCSA, una de las diferencias importantes es que hay parches disponibles para soportar SSl (Secure Socket layer).

c) NETSCAPE COMMUNICATIONS SERVER: Este es un servidor cargado de características de alto desempeño, sin embargo es caro. (Unix,Windows NT).

d) NETSCAPE COMERSE SERVER: Este producto es el mismo que Netscape Communications Server, con la excepción de que soporta SSL, y su precio es más alto. (Unix,Windows NT).

e) WEBSITE: Un Servidor Web muy poderoso y popular, pero no intuitivo, es un producto comercial. (Windows NT, Windows 95).

f) AOLServer: Es un producto open-source que fue diseñado conociendo varias deficiencias que existían en el modelo inicial utilizado por Apache. Esta desarrollado con Threading ,es decir, comparte la memoria del Proceso general en varios sub-procesos o Threads.

g) Zope: Es un servidor de páginas open-source que utiliza Python como su "Scripting Language" y es capaz de accesar un gran número de bases de datos.

h) Internet Information Server (IIS): Es el servidor de páginas desarrollado por Microsoft para Windows NT/2000, a diferencia de los dos servidores de páginas mencionados anteriormente, IIS solo puede operar en plataformas Windows. El punto más favorable de este servidor son ASP's que facilitan el desarrollo de aplicaciones y la sencillez de instalación.

1.9.2.8 WEB HOSTING

Web Hosting significa el servicio de alojamiento de las Páginas Web que forman un sitio, en un Servidor Web, asignándole una ubicación (dirección) para que pueda ser accesada por todos los usuarios de Internet. Cuando una empresa o persona crea una Página Web, los archivos de la página deben encontrarse en el servidor con una conexión directa hacia Internet, servicio de nombre de dominio, y otras configuraciones técnicas para que el sitio se encuentre accesible en la World Wide Web.

1.9.3 SISTEMAS DE INFORMACIÓN (SI)

1.9.3.1 DEFINICIÓN

Es el medio por el cual los datos fluyen entre personas o departamentos y puede ser, desde la comunicación interna entre los diferentes componentes de la organización y líneas telefónicas, hasta sistemas de cómputo que generan reportes periódicos para varios usuarios.

1.9.3.2 OBJETIVOS GENERALES DE LOS SI

- La principal función de un SI es proporcionar a los encargados de la toma de decisiones, datos oportunos y exactos que les permitan tomar y aplicar las decisiones necesarias que mejoren al máximo la relación que existe entre los recursos de la empresa.
- Este sistema tiene el propósito general de ayudar a los gerentes en la planeación, control y toma de decisiones.
- Asegurar que la información exacta y confiable esté disponible cuando se necesite y que se le presente en forma fácilmente aprovechable.
- Incrementar la productividad operacional.
- Hacer que el proceso de información deje de ser información fragmentada, conjeturas inspiradas en la intuición y solución de problemas aislados.

1.9.4 DISEÑO DE SISTEMAS

1.9.4.1 DEFINICIÓN

El diseño de sistemas es la evaluación de las distintas soluciones alternativas y la especificación de una solución detallada de tipo informático. También se conoce como diseño físico. Mientras que el análisis de sistemas concentra principalmente su interés en los aspectos lógicos, independientes de la implantación, de un sistema (las necesidades), el diseño de sistemas trata los aspectos físicos o

dependientes de la implantación del sistema (las especificaciones técnicas de dicho sistema).

1.9.4.2 FASES DEL DISEÑO DE SISTEMAS

1. Selección

Una vez obtenidas las necesidades de la empresa con respecto a la elaboración de un sistema de información mejorado, se puede planear el modo en que funcionará dicho sistema. Durante la fase de selección, es imperativo identificar y analizar las diversas opciones posibles, y solo entonces proponer las soluciones más viables sobre la base del análisis realizado.

2. Adquisición

La adquisición de software y hardware (equipo Informático) no es necesario en todos los nuevos sistemas. Por otra parte, cuando se precisa un nuevo software o hardware, la selección de los productos apropiados entraña con frecuencia una cierta dificultad. Las decisiones se complican por consideraciones técnicas, económicas y políticas. Una decisión deficiente puede arruinar cualquier tarea de análisis y diseño que, en caso contrario, habría logrado sus objetivos.

3. Diseño e Integración

Dadas las necesidades de diseño y las necesidades de integración del sistema objeto, esta fase implica el desarrollo de las especificaciones técnicas de diseño.

1.9.5 TECNICAS DE PROGRAMACION

Para la construcción de programas, es tan importante el conocimiento de las técnicas de programación, como de los conceptos básicos, sin los cuales la creación de los programas es muy difícil o casi imposible. Las nuevas teorías de

la programación se centran en las técnicas de programación modular y programación estructurada. El diseño de un programa entraña la descomposición del problema en módulos o partes independientes, es decir, programación modular y la programación de cada módulo mediante métodos estructurados o programación estructurada y su unión posterior.

1.9.5.1 PROGRAMACION ESTRUCTURADA

Se refiere a un conjunto de técnicas que han evolucionado. Estas técnicas aumentan considerablemente la productividad del programa reduciendo el tiempo requerido para escribir, verificar, depurar y mantener los programas. La programación estructurada utiliza un número limitado de estructuras de control que minimizan la complejidad de los problemas y que reducen los errores. Esta incorpora: diseño descendente, recursos abstractos y estructuras básicas. La programación estructurada permite la escritura de programas fáciles de leer y modificar. En un programa estructurado el flujo lógico se gobierna por las estructuras de control básicas: secuenciales, repetitivas y selectivas.

1.9.5.2 PROGRAMACION MODULAR

Es uno de los métodos de diseño más flexibles y potentes para mejorar la productividad de un programa. El programa se divide en módulos, cada uno de los cuales ejecuta una única actividad o tarea y se codifican independientemente de los demás módulos. Cada uno de estos módulos se analizan, codifican y ponen a punto por separado. Cada programa contiene un módulo llamado programa principal que controla todo lo que sucede; se transfiere el control a submódulos de modo que ellos puedan ejecutar sus funciones. Los módulos son independientes en el sentido en el que ningún módulo puede tener acceso directo a los demás, excepto el módulo al que llama y sus propios submódulos. El esquema de la Programación Modular se ilustra en la Figura N° 2.

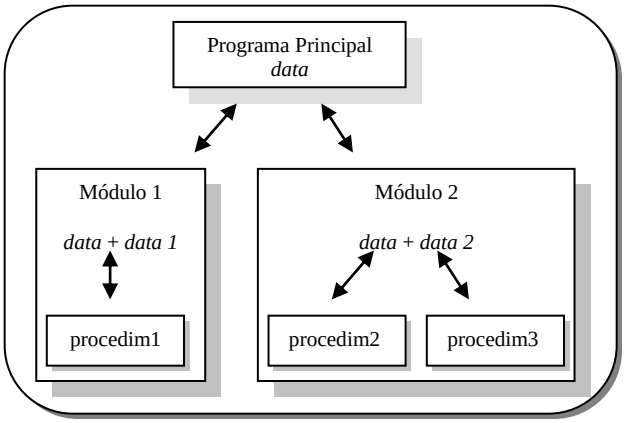


FIGURA N° 2. Programación Modular.

Dado que los módulos son independientes, diferentes programadores pueden trabajar simultáneamente en diferentes partes del mismo programa, de ésta manera se reduce gran tiempo de diseño del algoritmo y la codificación, además una modificación radical dentro de un módulo no afectará a los demás.

1.9.6 PROTOTIPOS

1.9.6.1 DEFINICIÓN

Un prototipo es un modelo (representación, demostración o simulación) fácilmente ampliable y modificable de un sistema planificado, probablemente incluyendo su interfaz y su funcionalidad de entradas y salidas. Es un modelo original que sirve de patrón y/o un primer ejemplar a escala real y usualmente funcional de un nuevo tipo o diseño.

1.9.6.2 DISEÑO MEDIANTE PROTOTIPOS

En la actualidad, muchos analistas aplican la técnica del uso de prototipos, un moderno método basado en ingeniería. La estrategia de diseño mediante prototipos esta siendo utilizada cada vez por más empresas. El método requerirá en cierta medida un planteamiento diferente al del ciclo de vida tradicional.

La fase de estudio es crucial, con independencia del método de diseño que se aplique. Deben identificarse, analizarse y comprenderse los problemas y las oportunidades para poder establecer los objetivos del nuevo sistema. Aunque la fase de definición puede simplificarse en los casos en que se use prototipos, deberían especificarse algunos requisitos generales antes de generar los prototipos.

Es importante completar la fase de selección antes de hacer los prototipos, de modo que se garantice que el sistema objeto sea la solución más viable para los problemas y necesidades expresados por los usuarios. Dada la solución más viable, puede iniciarse la elaboración de prototipos.

La elaboración de prototipos ha surgido como estrategia idónea para el diseño físico. Los prototipos son modelos de trabajo de un sistema. Los analistas pueden construir rápidamente prototipos mediante el empleo de lenguajes de cuarta generación y generadores de aplicaciones. La estrategia del uso de prototipos no es un sustituto del ciclo de vida.

Todas las fases del ciclo de vida son esenciales para realizar con éxito el desarrollo de sistemas. Sin embargo, la elaboración de prototipos permite agrupar en una etapa porciones de las fases de definición, diseño físico y construcción del ciclo de vida tradicional. En consecuencia, los prototipos, aumentan la productividad.

Ventajas del diseño mediante prototipos

- Los prototipos alientan y requieren la participación activa de los usuario finales. Ello sirve para elevar la moral de los usuario finales y aumentar su apoyo al proyecto.

- La interacción y el cambio son consecuencias naturales del desarrollo de sistemas, es decir, los usuarios suelen cambiar de opinión. La elaboración de prototipos se ajusta mejor a este enfoque, ya que supone que un prototipo puede evolucionar mediante sucesivas iteraciones hasta convertirse en el sistema requerido.
- Se ha dicho a menudo que los usuarios finales no conocen completamente sus necesidades hasta que no las ven implementadas. Siendo así, los prototipos se adecuan a esa filosofía.
- Los prototipos son modelos activos, no pasivos, que los usuarios finales pueden ver, tocar y sentir, y con los que pueden practicar.
- Un prototipo aprobado equivale a las especificaciones de diseño en papel, con la ventaja de que los errores pueden ser detectados mucho antes.
- La elaboración de prototipos puede estimular la creatividad, dado que permite al usuario manifestar su opinión con mayor rapidez, lo que se traduce en mejores soluciones.

1.9.7 MEDIANA EMPRESA

A continuación se detallan aspectos relacionados con el concepto de Mediana Empresa, se hablará en general de la Pequeña y Mediana Empresa (PYME), ya que el concepto de Mediana Empresa esta íntimamente ligado al concepto de Pequeña Empresa.

1.9.7.1 DEFINICIÓN

En El Salvador no existe una definición única del significado de la PYME. Las definiciones y criterios utilizados varían entre las distintas instituciones públicas o privadas que están relacionadas con el desarrollo de este sector empresarial. Estas tienden a utilizar los criterios de clasificación que más se adaptan a la

disponibilidad de información estadística generada por las instituciones encargadas.

Mediana Empresa es toda unidad económica que emplea entre 50 y 99 trabajadores[5].

1.9.7.2 CARACTERÍSTICAS

Según estudios de FUNDAPYME, las PYME presentan algunas de las siguientes características:

- Las PYME tienen como principal cliente a los consumidores finales. El segmento de mayor importancia para el 54% de las empresas lo constituye el consumidor final.
- La mayoría de las PYME son empresas maduras. El 55% de las empresas pequeñas y medianas tienen más de diez años de haber iniciado actividades.
- Los empresarios de las PYME tienen una experiencia empresarial relativamente grande. La mayor experiencia empresarial está presente en el sector de la mediana empresa, en donde los propietarios y gerentes propietarios tienen más de 10 años de experiencia.
- Las PYME salvadoreñas destinan su oferta fundamentalmente al mercado nacional.
- La incidencia de empresas que exportan actualmente es apenas del 14%, y el porcentaje de las que no exportan actualmente, pero tienen interés en hacerlo en el futuro es de apenas el 8%.

[5] **FUENTE**: FUNDAPYME, sobre la base de entrevistas realizadas a las instituciones.

1.10 TEORIA ADOPTADA

El desarrollo del proyecto se realiza bajo El Sistema Operativo Linux en su distribución Red Hat 8.0. Este es un Sistema Operativo bastante robusto, multiplataforma, multiusuario y que además, es de libre distribución por lo que proporciona la ventaja de no requerir ningún costo de licencias, para su uso. Esto es de gran utilidad en el desarrollo del proyecto, ya que ayuda en gran manera a reducir los costos de instalación y operación.

En cuanto a las Herramientas de desarrollo se ha considerado la utilización del lenguaje de programación Perl[6], debido a que este incorpora librerías con las que se pueden implementar operaciones con los puertos o programación con sockets, y que al igual que el Sistema Operativo, es de libre distribución. Existen varias librerías de comunicaciones en Perl, el paquete IO::Socket[7], específicamente, es una de ellas. Esta será una herramienta bastante útil para el desarrollo del proyecto. Además, se ha considerado el uso de la Programación Modular para el desarrollo del software, debido a las ventajas que esta proporciona, ya que la división en módulos hará mas eficiente y organizado el diseño por lo que las modificaciones y expansiones del mismo serán mucho más fáciles.

Por otra parte, se elaborará un prototipo de un Sitio Web para evaluar la funcionalidad y operabilidad del software.

---

[6] Perl que significa "Practical Extraction and Report Language" es un lenguaje de programación que se utiliza principalmente para labores de procesamiento de texto, programación de software de sistemas y como lenguaje para programar aplicaciones para WWW.

[7] IO::Socket es una librería de comunicación en Perl que proporciona una interfaz sencilla para crear y operar sockets o puertos.

CAPITULO II. INVESTIGACIÓN DE CAMPO

2.1 OBJETIVO

Realizar una investigación de campo en las Medianas Empresas Salvadoreñas para cuantificar e interpretar los datos obtenidos de cada una de ellas con el fin analizar el impacto de Internet y los beneficios que ofrece para dichas empresas, así como determinar las tendencias en cuanto al tipo de tecnología que utilizan para el mantenimiento de sus Sitios Web y sus capacidades de inversión.

2.2 METODOLOGIA DE INVESTIGACION

2.2.1 METODO DE INVESTIGACIÓN

Es necesario utilizar en toda investigación un método que sirva de camino para lograr el conocimiento. En el presente documento se utiliza el método inductivo ya que de acuerdo a la naturaleza de la investigación es el que más se adapta, debido a que este comienza de lo particular a lo general; en este caso se estudia una muestra representativa de las Medianas Empresas establecidas y registradas legalmente, de acuerdo a la "Dirección General de Estadísticas y Censos" (DIGESTYC) y la "Comisión Nacional de la Micro y Pequeña Empresa" (CONAMYPE), posteriormente se generalizan los resultados del análisis a todas las Medianas Empresas Salvadoreñas. Una de las ventajas de este método es que existe contacto directo con el objeto de investigación, en este caso las Medianas Empresas de El Salvador.

2.2.2 TIPO DE ESTUDIO

El desarrollo de la Investigación se caracteriza de acuerdo a los siguientes criterios:

Exploratorio: Por la Necesidad esencial de familiarizarse con el tema. Este es novedoso y poco estudiado en nuestro medio. Además es el punto de partida para estudios posteriores de mayor profundidad.

Descriptivo: Debido que se analiza e interpreta el funcionamiento de Los Servidores Web en cuanto a software se refiere, y se exponen sus características.

De Campo: Ya que este estudia el fenómeno en el escenario donde se manifiestan los hechos, se visitaron Medianas Empresas legalmente establecidas y registradas en El Salvador de acuerdo a DIGESTYC, siendo esta una muestra representativa.

2.2.3 TÉCNICAS DE INVESTIGACIÓN

2.2.3.1 RECOPILACIÓN DE DATOS

Para el desarrollo del proyecto se realizó una investigación tanto documental como de campo. La información de carácter documental amplió el conocimiento acerca del funcionamiento y el diseño del Software para servir páginas en Internet y las herramientas que se utilizan para su desarrollo. Así mismo, fue de utilidad para obtener información sobre proyectos similares. Las fuentes de consulta fueron de dos tipos:

a. Bibliográfica

Que incluyó libros de texto sobre Servidores Web, Manuales Técnicos sobre las herramientas utilizadas para el desarrollo de este tipo de proyectos, Manuales Técnicos de Servidores Web, Tesis relacionadas con el tema.

b. Internet

Fue una de las principales fuentes de información, ya que de ella se pudo obtener información descriptiva sobre Los Servidores Web y el software que utilizan,

Estadísticas de su Uso, estadísticas de publicidad y usos de Internet de las Empresas, Manuales sobre las tecnologías de desarrollo e información sobre proyectos similares de otras Universidades.

En la investigación de campo se aplicaron las siguientes técnicas de recopilación de información:

a. Entrevistas

La entrevista consiste en una conversación entre dos o más personas, sobre un tema determinado de acuerdo a ciertos esquemas o pautas determinadas. Mediante las entrevistas se obtuvo información referente a la situación de las Medianas Empresas en cuanto a usos de Internet y tecnología informática. De igual manera se obtuvo información sobre aspectos relacionados al diseño y desarrollo de software para servir páginas en Internet. Las entrevistas realizadas fueron de dos tipos.

a.1 Expertos en el tema

Se entrevistó a personas que por su trabajo o experiencia, conocen sobre tecnologías o herramientas de desarrollo, así como detalles de implementación que contribuyeron al desarrollo del proyecto.

a.2. Empresas

De la misma manera, se realizaron entrevistas con el personal de las Medianas Empresas para recolectar información sobre el uso y administración de Sitios Web y de Servidores Web. La información obtenida de estás fue recopilada en las encuestas.

b. Observación

Consiste básicamente en utilizar los sentidos para observar los hechos, realidades sociales y a las personas en su contexto cotidiano con el objetivo de obtener información que es difícil obtener a través de una entrevista o encuesta.

Se visitaron de forma presencial, Medianas Empresas Salvadoreñas, de esta manera se conocieron los usos de Internet y las tendencias de administración de los Servidores Web y el mantenimiento de dichos Sitios.

c. Encuestas

Es una metodología que satisface todas las exigencias de investigación, para su realización se necesita una planificación y su objetivo es investigar acontecimientos presentes ocurridos y de opinión; el instrumento utilizado fue el cuestionario, ya que es la traducción de los objetivos de la investigación a preguntas especificas y operacionaliza problemas de investigación. El cuestionario que se utilizó es de carácter genérico, conteniendo una diversidad de preguntas las cuales son de tipo abiertas, cerradas y categorizadas.

Se realizaron las encuestas con el objetivo de conocer de manera directa, la forma en que las Medianas Empresas hacen uso de Internet, las tecnologías que utilizan y sus necesidades de uso en cuanto a Sitios Web y Servidores Web. Así mismo, las encuestas permitieron determinar la situación actual de dichas empresas en cuanto a tecnología se refiere. (Ver anexo II, en él se presentan los Modelos Propuestos para la Encuestas, una para el Usuario Técnico y otra para el Usuario Administrativo).

2.2.3.2 ANÁLISIS DE LA INFORMACIÓN

Utilizando la información obtenida en las encuestas, se determinó la medida en que las Medianas empresas hacen uso de Internet y de Servidores Web, así como se determinó la situación tecnológica actual de las mismas. Se realizó una tabulación de los datos para estimar los beneficios que dichas empresas pueden obtener por medio de la mayor publicidad y la gestión de labores o servicios desde Internet, así como sus necesidades de uso de tipo informáticas.

2.2.3.3 DISEÑO PROPUESTO

Se realizó el diseño del software prototipo para la creación del Servidor Web, atendiendo los requerimientos planteados para una Mediana Empresa.

2.2.3.4 DOCUMENTACIÓN

La funcionalidad y operabilidad del software del Servidor Web están incluidos en el manual del usuario y del programador. El manual del programador describe los procesos y todo lo referente al servidor en cuanto a programación se refiere. El manual del usuario sirve como una guía detallada para configurar y operar el Software del Servidor, así mismo, se incluirá en dicho manual, un apartado para la instalación del software y la configuración del Firewall.

2.2.4 MARCO MUESTRAL

El marco muestral esta determinado por toda la población de la cuál se extrae la muestra en la que se realiza la investigación. Para el análisis de la muestra se utilizó el método estadístico inferencial.

2.2.4.1 POBLACIÓN

La población constituye la totalidad de un grupo de elementos u objetos que se quiere investigar, es el conjunto de todos los casos que concuerdan con lo que se pretende investigar. La población que forma el universo en la investigación lo constituyen las Medianas Empresas legalmente registradas en El Salvador, ya que

se considera que una empresa de esta categoría, tiene la capacidad de hacer una inversión, de acuerdo a los requerimientos mínimos planteados para el proyecto.

Para la obtención de estos datos se consultó a “La Comisión Nacional de la Micro y Pequeña Empresa” (CONAMYPE), quienes facilitaron información sobre el total de Empresa legalmente registradas en El Salvador clasificadas por su tamaño. Por otra parte, se consultó a “La Dirección General de Estadísticas y Censos” (DIGESTYC) para la obtención de las Empresas Legalmente registradas. Es importante mencionar el hecho de que pueden existir muchas Medianas Empresas que no estén legalmente registradas y para efectos de la investigación no se tomaran en cuenta. A continuación se presenta el cuadro N° 4 con los datos recolectados.

CUADRO N° 4. Total de Empresas en El Salvador.

El Salvador ***2000***	
Tipo	**No. De establecimientos**
Microempresas	512,877
Pequeñas empresas	4,327
Medianas empresas	**502**
Grandes empresas	316

El total de Medianas Empresas legalmente registradas es de 502. Este es el tamaño de la Población a utilizar. La referencia más reciente es del año 2,000.

2.2.4.2. MUESTRA

La muestra es un subconjunto de la población o parte representativa de la misma. Se utilizó la siguiente fórmula[8] para obtener el tamaño de la muestra; siendo está una recomendación de los métodos estadísticos probabilísticos para población finita.

[8] Obtenida de documento en http://www.mrbit.es/hsa/vai/muestreo/

$$n = \frac{N (z^2 \times P \times Q)}{E^2 (N-1) + (z^2 \times P \times Q)}$$

donde:

N = Tamaño del universo

z = Nivel de Confianza

P x Q = Factores de Variabilidad de fenómeno

E = Error

n = Tamaño de la muestra

operando se tiene:

N = 502

Z = Se utilizará el nivel de confianza de 0.95% por lo tanto: 0.95/2=0.475.

De acuerdo a la tabla de la curva normal se tiene que: Z = 1.96

P = 0.5 Es la probabilidad de éxito

Q = 0.5 Es la probabilidad de fracaso

E =10%

Utilizando la fórmula para calcular la muestra (n) se tiene:

$$n = \frac{502[(1.96)^2 (0.5) (0.5)]}{[(0.1)^2 (502 - 1)] + [(1.96)^2 (0.5) (0.5)]}$$

n = 81

2.3 INTERPRETACION DE LOS RESULTADOS

Después de haber realizado la investigación de campo por medio de las técnicas establecidas, se procesaron los datos recabados en tablas estadísticas, las cuales proporcionan información del objeto en estudio y, están conformadas por títulos, encabezado, concepto o columna matriz y cuerpo.

Los tipos de gráficos que se utilizan son de pastel o circular y el gráfico de barras agrupadas. El gráfico de pastel, consiste en una circunferencia cuya superficie esta dividida en sectores circulares, cada una de las cuales representa la parte proporcional de los datos o la suma de todas las frecuencias. El gráfico de barras agrupadas, consiste en un plano cartesiano en el que se encuentran una serie de barras donde cada una representa una categoría; de tal manera que se pueden comparar los valores entre cada categoría de la serie. La razón por la que se utilizan estos tipos de gráficos, es debido a que por sus características la visualización de los resultados se hace más fácil.

2.3.1 ENTREVISTAS

a. Empresas

Las entrevistas a empresas se realizaron de manera paralela con las encuestas, ya que para lograr obtener los datos solicitados en las mismas, fue necesario concertar entrevistas con el personal involucrado. Los datos obtenidos se recopilaron en las encuestas.

b. Expertos en el tema

En esta etapa se realizaron entrevistas con dos personas que por su experiencia y por su trabajo, tienen conocimiento sobre este tipo de proyectos. Esto con el objetivo de tener una base metodológica sobre el desarrollo e implementación de los Servidores Web.

De esta parte de la Investigación, se obtuvo información de importancia para el desarrollo del software. Las personas entrevistadas brindaron datos sobre el lenguaje de programación Perl, en cuanto a las librerías con las que se pueden implementar operaciones con los puertos o programación con sockets. Existen varias librerías de comunicaciones en Perl, el paquete IO::Socket, específicamente, proporciona una interfaz muy sencilla para crear y operar con sockets.

De igual manera, se obtuvo información acerca de la arquitectura sobre la que trabajan los Servidores Web a nivel de software y recomendaron la programación o el desarrollo modular, de esta manera se tendrá un diseño eficiente, además de que brindará estabilidad, y permitirá la fácil adición de otras funciones al servidor sin que otras dejen de funcionar.

Por otra parte, se coincidió en el hecho de que existe muy poca información técnica sobre el diseñó de Servidores Web, y señalan que la implementación de este tipo de proyectos contribuye en gran medida al desarrollo, tanto económico como tecnológico de nuestro país, dando pautas para que se puedan desarrollar a largo plazo, proyectos a mayor escala.

2.3.2 OBSERVACIÓN

Esta etapa se realizó en paralelo con las entrevistas y encuestas. Se logro obtener información sobre el tipo de tecnologías y equipos que poseen algunas de las empresas entrevistadas y encuestadas. De igual manera se obtuvo información sobre las actitudes, conocimientos y tendencias de las empresas en cuanto a tecnología informática. Al igual que las entrevistas, la información obtenida se recopiló en las encuestas.

2.3.3 ENCUESTAS

2.3.3.1 OBJETIVO GENERAL

Recolectar información que permita analizar el impacto de Internet y los beneficios que ofrece a la Mediana Empresa en El Salvador, así como determinar en que medida las empresas tienen presencia en Internet a través de un Sitio Web, el tipo de tecnología que utilizan para el mantenimiento de los mismos y sus capacidades y disponibilidad de inversión.

2.3.3.2. PRESENTACIÓN DE LOS RESULTADOS

PREGUNTA N° 1

¿En su empresa tienen conexión a Internet?

OBJETIVO:

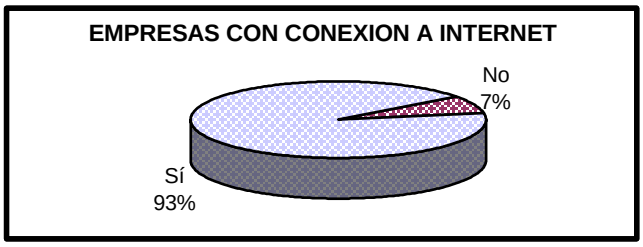
Determinar el porcentaje de Medianas Empresas que tienen acceso a Internet.

RESULTADOS:

TABLA N° 1

ALTERNATIVA		N° DE EMPRESAS	PORCENTAJE
a	Sí	75	93%
b	No	6	7%
Total		**81**	**100%**

GRAFICA N° 1



INTERPRETACIÓN DE LOS RESULTADOS:

De las 81 empresas encuestadas 93% de éstas afirma que poseen conexión a Internet. Esto nos indica que la mayoría de las empresas encuestadas poseen conexión a Internet.

PREGUNTA N° 2

¿Cuál es el fin principal de acceso a Internet en su Empresa?

OBJETIVO:

Conocer cuales son los fines de acceso a Internet de las empresas y de esta manera determinar la medida en que ciertas actividades se realizan en las empresas a través de Internet y los beneficios que pueden percibir. (Las empresas pueden seleccionar más de una opción).

RESULTADOS:

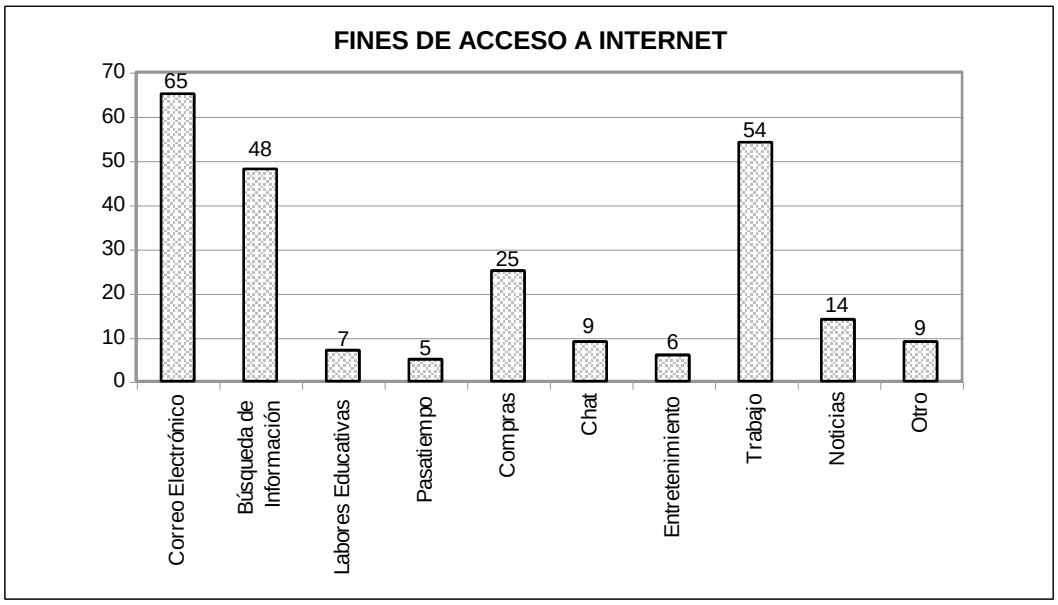
TABLA N° 2

ALTERNATIVA		FRECUENCIA (f) (No.Empresas que seleccionaron c/opción)	FRECUENCIA RELATIVA (fr) ((f/N)*100)[9]
a	Correo Electrónico	65	27%
b	Labores Educativas	7	3%
c	Compras	25	10%
d	Entretenimiento	6	2%
e	Trabajo	54	22%
f	Noticias	14	6%
g	Búsqueda de Información	48	20%
h	Pasatiempo	5	2%
i	Chat	9	4%
j	Otro	9	4%
TOTAL		**N=242 (Σf=N)[10]**	**100%**

[9] El valor del porcentaje es equivalente a la Frecuencia Relativa (f/N)*100, de la Alternativa, siendo f el número de casos por alternativa y N el total de casos obtenidos. Formula tomada del Libro: Estadística I, Gildalberto Bonilla.

[10] La sumatoria de todas las frecuencias o el número de casos por alternativa, es igual al total de casos **(Σf=N)** . Formula tomada del Libro: Estadística I, Gildalberto Bonilla.

GRAFICA N° 2



INTERPRETACIÓN DE LOS RESULTADOS:

De acuerdo a los resultados de la tabla N° 2, de las 75 empresas que poseen conexión a Internet, el 27% manifiestan que accesan a Internet para utilizar Correo Electrónico, siendo este el principal fin de acceso; el 22% lo hace para realizar actividades relacionadas con su trabajo; el 20% para buscar información; el 10% para realizar compras; el 6% para conocer noticias. Estas son las 5 actividades con mayor porcentaje.

Como se puede observar, entre los fines principales de acceso a Internet se encuentra el trabajo, esto indica que Internet es una herramienta importante para las Medianas empresas salvadoreñas, ya que el 20% lo utilizan para el desempeño de sus actividades laborales. La presencia de las Medianas empresas en Internet ayudará a incrementar su desarrollo económico, a través de las ventas en línea, la publicidad, servicio al cliente, ya que de esta manera podrán abarcar nuevos mercados y expandir sus operaciones, haciéndolas más competitivas.

PREGUNTA N° 3

¿Que tipo de información busca regularmente en Internet su empresa?

OBJETIVO:

Determinar que tipo de información buscan las Medianas empresas de El Salvador en Internet, de esta manera poder estimar las tendencias en cuanto a usos de información y el impacto del mismo para dichas empresas. (Las empresas pueden seleccionar más de una opción).

RESULTADOS:

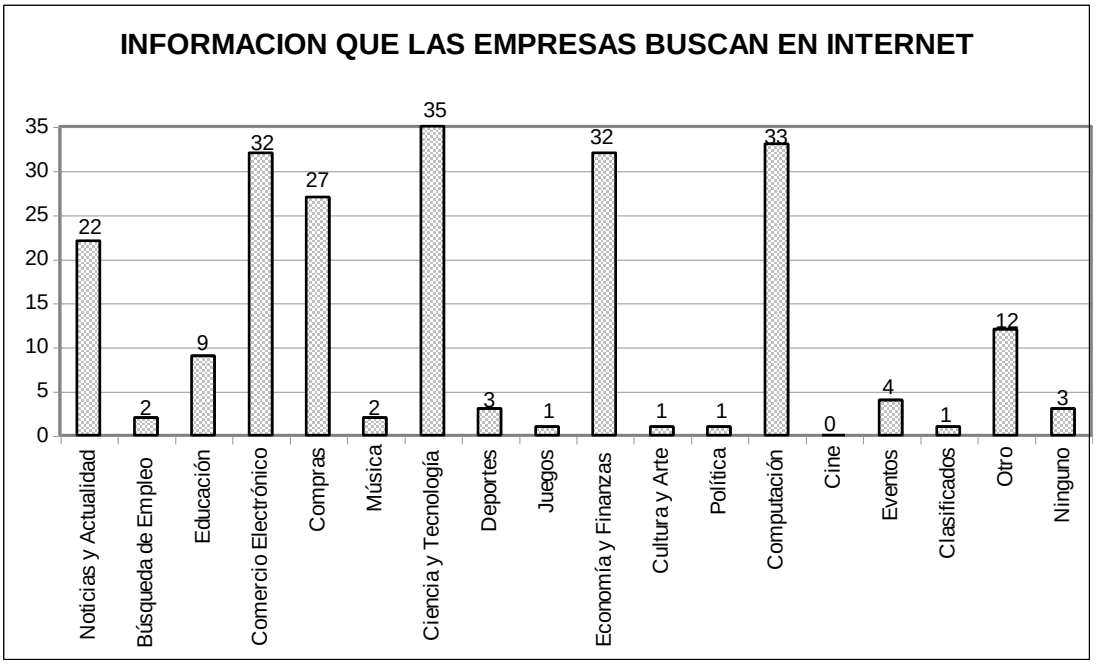
TABLA N° 3

ALTERNATIVA		FRECUENCIA (f) (No.Empresas que seleccionaron c/opción)	FRECUENCIA RELATIVA (fr) ((f/N)*100)[11]
a	Noticias y Actualidad	22	10%
b	Compras	27	12%
c	Deportes	3	1%
d	Cultura y Arte	1	0.45%
e	Cine	0	0%
f	Búsqueda de Empleo	2	1%
g	Música	2	1%
h	Juegos	1	0.45%
i	Política	1	0.45%
j	Eventos	4	2%
k	Educación	9	4%
l	Ciencia y Tecnología	35	16%
m	Economía y Finanzas	32	15%
n	Computación	33	15%
o	Clasificados	1	0.45%
p	Comercio Electrónico	32	15%
q	Otro	12	5%
r	Ninguno	3	1%
TOTAL		**N=220 (Σf=N)[12]**	**100%**

[11] El valor del porcentaje es equivalente a la Frecuencia Relativa (f/N)*100, de la Alternativa, siendo f el número de casos por alternativa y N el total de casos obtenidos. Formula tomada del Libro: Estadística I, Gildalberto Bonilla.

[12] La sumatoria de todas las frecuencias o el número de casos por alternativa, es igual al total de casos **(Σf=N)** . Formula tomada del Libro: Estadística I, Gildalberto Bonilla.

GRAFICA N° 3



INTERPRETACIÓN DE LOS RESULTADOS:

Tal como se observa en la tabla N° 3, entre los tipos de información con mayor porcentaje están Ciencia y Tecnología con el 16%, Computación, Economía y Finanzas y Comercio Electrónico, con el 15%, Compras con el 12%, Noticias y Actualidad con el 10%. De esta manera, es notable el hecho de que las empresas utilizan Internet para la búsqueda de información que tiene relación con su ámbito laboral. Cabe mencionar que entre estos tipos de información están las compras y comercio electrónico, lo que se traduce en un interés de las empresas por utilizar Internet para transacciones en línea.

PREGUNTA N° 4

¿Su empresa Utiliza la Publicidad en Internet, a través de banners u otros métodos, en algún sitio de Internet?

OBJETIVO:

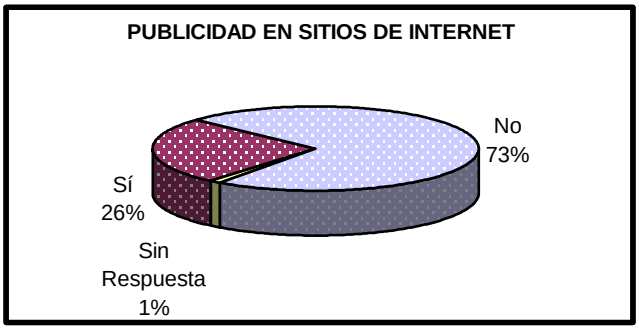
Determinar si las Medianas Empresas de El Salvador utilizan la publicidad en Internet, por algún medio que no sea su propio Sitio o Página Web.

RESULTADOS:

TABLA N° 4

ALTERNATIVA		No DE EMPRESAS	PORCENTAJE
a	Sí	21	26%
b	No	59	73%
	Sin Respuesta	1	1%
Total		**81**	**100%**

GRAFICA N° 4



INTERPRETACIÓN DE LOS RESULTADOS:

La gráfica N° 4 muestra que el 26% de las empresas utilizan publicidad en Sitios no propios en Internet, a través de banners o links a sus Sitios; el 73% no poseen y el 1% no respondió a la pregunta. Esto indica que la mayoría de Medianas empresas de El Salvador no utilizan este tipo de métodos de publicidad en otros sitios de Internet diferentes al propio.

PREGUNTA N° 5

¿Posee su empresa Página Web o Sitio Web?

OBJETIVO:

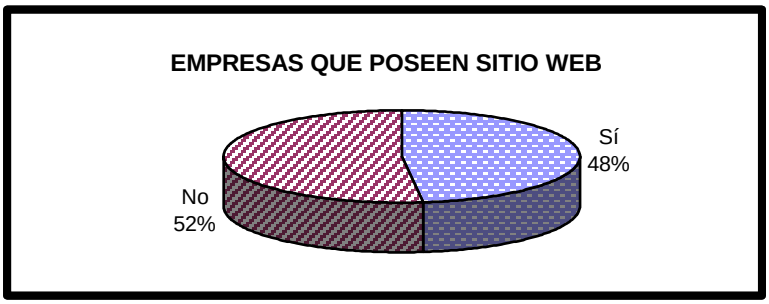
Conocer el Porcentaje de Medianas Empresas Salvadoreñas que tienen presencia en Internet a través de un Sitio o Página Web.

RESULTADOS:

TABLA N° 5

ALTERNATIVA		No DE EMPRESAS	PORCENTAJE
a	Sí	39	48%
b	No	42	52%
Total		**81**	**100%**

GRAFICA N° 5



INTERPRETACIÓN DE LOS RESULTADOS:

El 52% de las empresas encuestadas no poseen Sitio Web, el 48% si posee. De acuerdo con esta información, la mayoría de las Medianas Empresas salvadoreñas no tienen presencia en Internet, esto indica que existe un amplio mercado potencial al cual dotar de Servicios de Internet para que puedan gozar de los beneficios que esto les puede proporcionar, abriéndoles las puertas a nuevas oportunidades que les permitirán aumentar sus ingresos y reducir sus costos, contribuyendo a su crecimiento económico y tecnológico.

PREGUNTA N° 6

¿Por qué razón su empresa tiene presencia en Internet?

OBJETIVO:

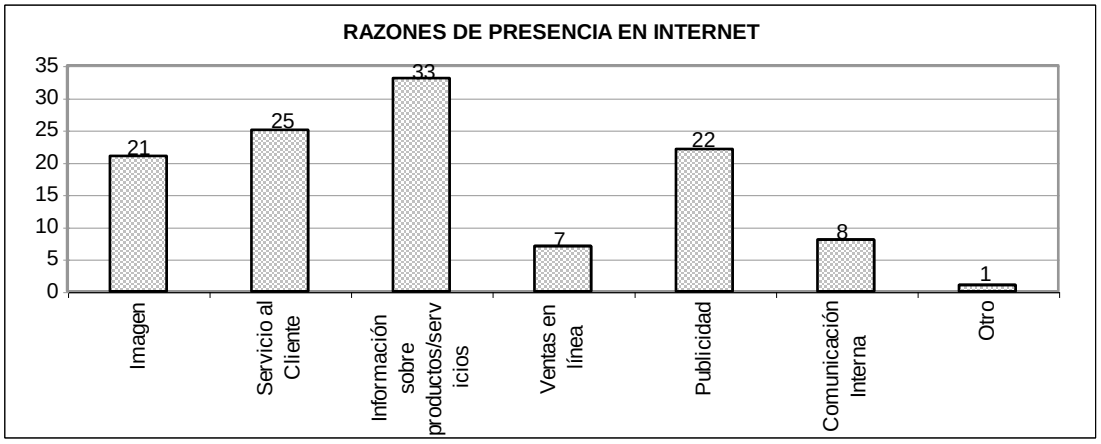
Identificar las razones por las que las medianas empresas salvadoreñas tienen presencia en Internet. (Las empresas pueden seleccionar más de una opción).

RESULTADOS:

TABLA N° 6

ALTERNATIVA		FRECUENCIA (f) (No.Empresas que seleccionaron c/opción)	FRECUENCIA RELATIVA (fr) ((f/N)*100)[13]
a	Imagen	21	18%
b	Información sobre productos/servicios	33	21%
c	Publicidad	22	28%
d	Servicio al Cliente	25	6%
e	Ventas en línea	7	19%
f	Comunicación Interna	8	7%
g	Otro	1	1%
TOTAL		**N=117 (Σf=N)[14]**	**100%**

GRAFICA N° 6



INTERPRETACIÓN DE LOS RESULTADOS:

La tabla N° 6 muestra que de las empresas encuestadas, que poseen Sitio Web; el 18% tienen presencia en Internet por Imagen; el 21% por información sobre productos/servicios; el 28% por publicidad; el 6% por Servicio al Cliente; 19% por ventas en línea; el 7% por comunicación interna y el 1% por otras razones. Como se puede observar entre las 3 razones principales están la información sobre

[13] El valor del porcentaje es equivalente a la Frecuencia Relativa (f/N)*100, de la Alternativa, siendo f el número de casos por alternativa y N el total de casos obtenidos. Formula tomada del Libro: Estadística I, Gildalberto Bonilla.
[14] La sumatoria de todas las frecuencias o el número de casos por alternativa, es igual al total de casos (Σf=N) . Formula tomada del Libro: Estadística I, Gildalberto Bonilla.

productos/servicios, servicio al cliente y la publicidad; esto indica que el mayor interés de las empresas salvadoreñas esta orientado al cliente y su satisfacción. Las ventas en línea cuentan con un 6%, que puede aumentar si se cuentan con alternativas que ayude a dichas empresas, a optimizar e implementar servicios de Internet a un costo accesible, aumentando su eficiencia y por lo tanto la satisfacción de sus clientes.

PREGUNTA N° 7

¿Cómo mantiene su Sitio Web o su Página Web?

OBJETIVO:

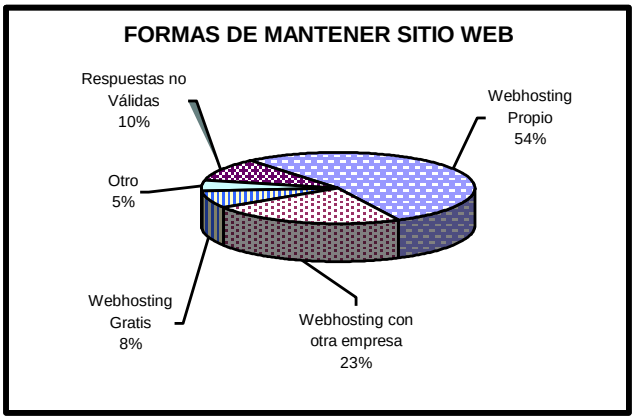
Determinar las formas en que las medianas empresas salvadoreñas que poseen Sitio Web, los mantienen.

RESULTADOS:

TABLA N° 7

ALTERNATIVA		No DE EMPRESAS	PORCENTAJE	PORCENTAJE TOTAL
a	Web Hosting Propio	21	54%	26%
b	Web Hosting con otra empresa	9	23%	11%
c	Web Hosting Gratis	3	8%	4%
d	Otro	2	5%	2%
	Respuestas no Válidas	4	10%	5%
Total		**39**	**100%**	**48%**

GRAFICA N° 7



INTERPRETACIÓN DE LOS RESULTADOS:

El 54% de empresas que poseen un Sitio Web, es decir el 48% del total, utiliza un Web Hosting propio para el mantenimiento de los mismos, esto representa el 26% del total de empresas encuestadas. El 23% utiliza Web Hosting con otra empresa que representa el 11% del total. El 8% poseen Web Hosting gratuito, es decir el 4% del total encuestado. El 5% utiliza otros mecanismos y el 10% fueron respuestas no válidas. Esto representa el 2% y el 5% del total encuestado respectivamente.

Como se puede observar, la mayoría de las empresas que poseen un Sitio Web cuentan con una infraestructura tecnológica que va desde una conexión dedicada a Internet hasta el software para servir sus página, esto representa un mercado potencial que en un futuro dado podría optar por buscar nuevas alternativas de equipo y software que les permita hacer más eficientes sus operaciones e innovar sus procesos a un costo accesible, ya que tienen una inversión realizada.

PREGUNTA N° 8

¿Que tipo de Servidor Web Utiliza?

OBJETIVO:

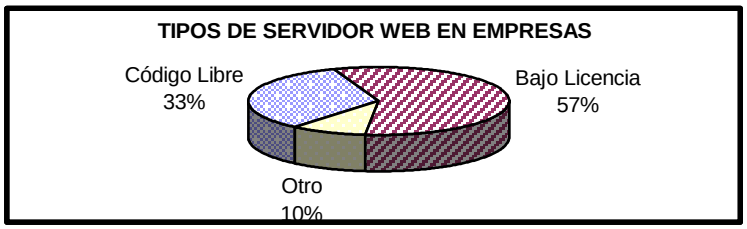
Identificar los tipos de software que las medianas empresas que poseen un Sitio Web, utilizan para servir sus páginas en Internet.

RESULTADOS:

TABLA N° 8

ALTERNATIVA		No DE EMPRESAS	PORCENTAJE	PORCENTAJE TOTAL
a	Código Libre	7	33%	9%
b	Bajo Licencia	12	57%	15%
c	Otro	2	10%	2%
Total		**21**	**100%**	**26%**

GRAFICA N° 8



INTERPRETACIÓN DE LOS RESULTADOS:

Del 26% de las empresas que poseen Web Hosting propio, el 33% utilizan un Servidor Web de Código Libre, esto es el 9% del total. El 57% utiliza un Servidor Web bajo licencia y representa el 15% del total. El 10% utiliza otro tipo siendo el 2% del total encuestado. De esta manera, los productos de código libre abarcan un nivel aceptable dentro del mercado nacional. Este podría aumentar en gran medida, a través de una buena publicidad de los beneficios y de la arquitectura de implementación utilizando este tipo de productos, ya que en el medio no existe mucha información o una orientación de calidad, de los mismos.

PREGUNTA N° 9

¿El Servidor Web que utiliza cumple con sus expectativas de funcionalidad?

OBJETIVO:

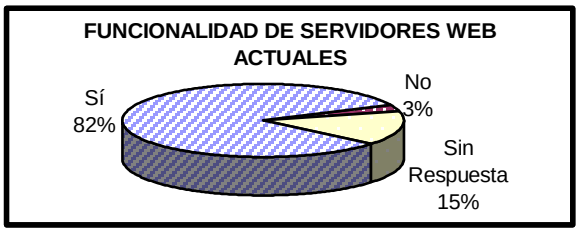
Determinar si las medianas empresas se encuentran satisfechas con el Software que utilizan para servir sus páginas en Internet.

RESULTADOS:

TABLA N° 9

ALTERNATIVA		No DE EMPRESAS	PORCENTAJE	PORCENTAJE TOTAL
A	Sí	32	82%	40%
B	No	1	3%	1%
	Sin Respuesta	6	15%	7%
Total		**39**	**100%**	**48%**

GRAFICA N° 9



INTERPRETACIÓN DE LOS RESULTADOS:

Del total de empresas que poseen un Sitio Web (48%), el 82% respondió afirmativamente; el 3% negativamente y el 15% se abstuvo de responder. Esto indica que la mayoría de las encuestadas se encuentran satisfechas, con el Servidor Web que poseen, o al menos cumple con las expectativas mínimas requeridas, hasta la actualidad.

PREGUNTA N° 10

¿Que tipo de problemas son los más frecuentes?

OBJETIVO:

Conocer los problemas que las Mediana Empresas tienen con los Servidores Web que utilizan para servir sus páginas en Internet. (Las empresas pueden seleccionar más de una opción).

RESULTADOS:

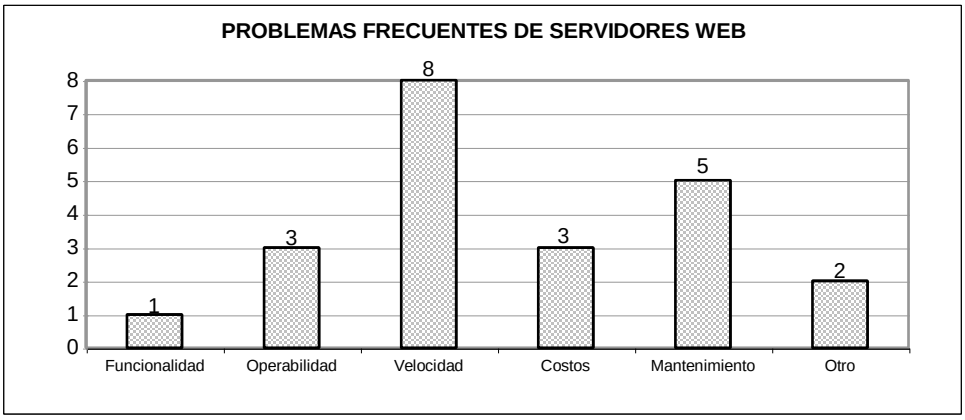
TABLA N° 10

ALTERNATIVA		FRECUENCIA (f) (No.Empresas que seleccionaron c/opción)	FRECUENCIA RELATIVA (fr) ((f/N)*100)[15]
a	Funcionalidad	1	4%
b	Operabilidad	3	14%
c	Velocidad	8	36%
d	Costos	3	14%
e	Mantenimiento	5	23%
f	Otro	2	9%
TOTAL		**N=22 (Σf=N)[16]**	**100%**

[15] El valor del porcentaje es equivalente a la Frecuencia Relativa (f/N)*100, de la Alternativa, siendo f el número de casos por alternativa y N el total de casos obtenidos. Formula tomada del Libro: Estadística I, Gildalberto Bonilla.

[16] La sumatoria de todas las frecuencias o el número de casos por alternativa, es igual al total de casos (Σf=N) . Formula tomada del Libro: Estadística I, Gildalberto Bonilla.

GRAFICA N° 10



INTERPRETACIÓN DE LOS RESULTADOS:

Como muestra la tabla N° 10, de las empresas que poseen Sitio Web el 4% manifiesta que frecuentemente tienen problemas de funcionalidad con sus Servidor Web; el 14% problemas de Operabilidad; el 36% de Velocidad; el 14% dificultades en cuanto a costos; el 23% problemas de Mantenimiento y el 9% otro tipo de problemas como el control de accesos. Esto indica que si bien es cierto, la mayoría de las empresas manifiestan que los Servidores Web con los que cuentan, cumplen con sus expectativas mínimas de funcionalidad, estos presentan al menos un problema que a largo plazo puede disminuir la funcionalidad de los mismos.

PREGUNTA N° 11

¿Su empresa estaría dispuesta a experimentar con otras alternativas tecnológicas en cuanto a lo que a Servidores Web se Refiere? ¿Por qué Razón?

OBJETIVO:

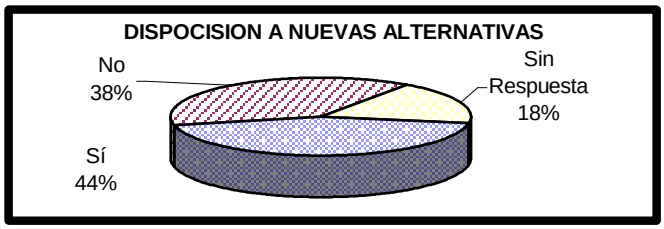
Determinar la disponibilidad, de las Medianas empresas salvadoreñas para experimentar con otras tecnologías en cuanto a Servidores Web, diferentes a las que ya poseen, así como identificar las razones por las que realizarían o no dicho cambio.

RESULTADOS OPCIONES:

TABLA N° 11a

ALTERNATIVA		No DE EMPRESAS	PORCENTAJE	PORCENTAJE TOTAL
a	Sí	17	44%	21%
b	No	15	38%	19%
	Sin Respuesta	7	18%	9%
Total		**39**	**100%**	**48%**

GRAFICA N° 11



RESULTADOS RAZONES:

TABLA N° 11b

RAZONES SI	RAZONES NO
● Adquirir conocimientos nuevos	● El tipo de Negocio necesita tecnología específica
● Conocer beneficios de otras tecnologías	● Estabilidad, buen soporte, confianza y seguridad
● Optimizar servicios y disminuir costos	● Cumple con las expectativas
● Actualizar tecnología	● No se pueden adquirir nuevos Costos
● Escalabilidad	● Planes ya establecidos y una inversión realizada
● Mejorar rendimientos	● Cambios implica pruebas y días fuera de línea
	● No es necesario

INTERPRETACIÓN DE LOS RESULTADOS:

En la gráfica N° 11, el 44% respondió que afirmativamente estaría dispuesta a experimentar con otras tipos de Servidores Web, diferentes al que poseen; el 38% respondió negativamente y el 18% se abstuvieron de responder. De esta manera, se tiene un porcentaje aceptable de empresas que forman parte de un mercado potencial para nuevos productos que se adecuen a las necesidades informáticas y a la capacidad de inversión de dichas empresas. Cabe mencionar, que las

empresas en disponibilidad de experimentar con otras alternativas, representan el 44% de las empresas que poseen un Sitio Web y el 21% de el total de empresas encuestadas. El 38% que respondió que No y el 18% de las que no respondieron, representan el 19% y el 9% respectivamente. En la tabla 11b se detallan las razones por las cuales, las empresas justifican su respuesta, ya sea que respondieron afirmativamente o negativamente.

PREGUNTA N° 12

¿Que tipo de Servidor Web Elegiría?

OBJETIVO:

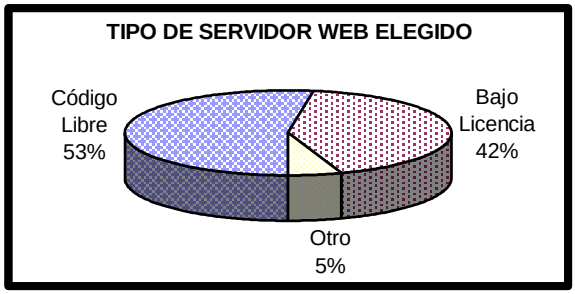
Identificar los tipos de Servidores Web con los que las empresas estarían dispuestas a experimentar.

RESULTADOS:

TABLA N° 12

ALTERNATIVA		No DE EMPRESAS	PORCENTAJE	PORCENTAJE TOTAL
a	Código Libre	10	53%	12%
b	Bajo Licencia	8	42%	10%
c	Otro	1	5%	1%
Total		**19**	**100%**	**23%**

GRAFICA N° 12



INTERPRETACIÓN DE LOS RESULTADOS:

Del 48% de las empresas que poseen un Sitio Web, el 53% elegiría un Servidor Web de Código Libre, esto representa el 12% del total encuestado. El 42% se inclina por un Servidor Web bajo Licencia, siendo el 10% del total. El 5% opta por otro que para el caso, la elección depende de un análisis costo-beneficio, representando el 1% del total. Es importante mencionar que dentro del 53% que se inclina por el servidor de código libre existen dos empresas que respondieron que No a la pregunta de disponibilidad para experimentar con otras tecnologías, pero que sin embargo, si lo hicieran se inclinarían por este tipo de servidores.

PREGUNTA N° 13

¿Esta enterado de las tarifas que cobran las Empresas Salvadoreñas que brindan el servicio de Web Hosting?

OBJETIVO:

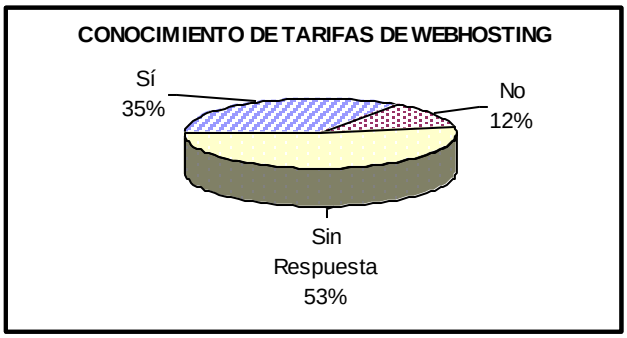
Determinar si las empresas están enteradas de las tarifas que cobran las empresas que se dedican a brindar el servicio de Web Hosting.

RESULTADOS:

TABLA N° 13

ALTERNATIVA		No DE EMPRESAS	PORCENTAJE
a	Sí	28	35%
b	No	10	12%
	Sin Respuesta	43	53%
Total		**81**	**100%**

GRAFICA N° 13



INTERPRETACIÓN DE LOS RESULTADOS:

En la gráfica N° 13 se puede observar que, del 100% de las empresas encuestadas, el 35% conocen las tarifas que cobran las empresas que brindan el servicio de Web Hosting; el 12% no tienen conocimiento y el 53% no respondieron a la pregunta. De este porcentaje de abstención el 37% corresponden a usuarios administrativos y 63% a usuarios técnicos. Esto indica que en general, existe poco conocimiento sobre los precios del servicio de Web Hosting, y no es una alternativa muy utilizada por las empresas, ya que el porcentaje de usuarios técnicos que no contestaron, es bastante alto.

PREGUNTA N° 14

¿Cómo le parecen las tarifas?

OBJETIVO:

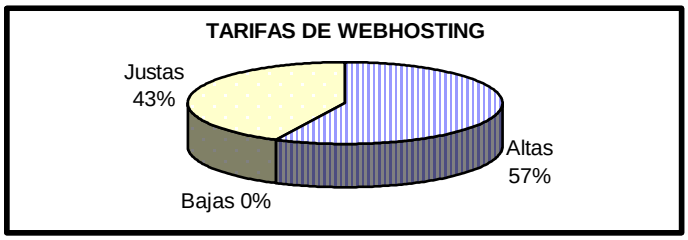
Determinar la forma en que consideran las Medianas Empresas, las tarifas de Web Hosting existentes en el mercado local.

RESULTADOS:

TABLA N° 14

ALTERNATIVA		No DE EMPRESAS	PORCENTAJE	PORCENTAJE TOTAL
a	Altas	16	57%	20%
b	Bajas	0	0%	0%
c	Justas	12	43%	15%
Total		**28**	**100%**	**35%**

GRAFICA N° 14



INTERPRETACIÓN DE LOS RESULTADOS:

Del 35% de las empresas que conocen las tarifas del servicio de Web Hosting; el 57% considera estas tarifas altas; el 43% las considera justas y el 0% las considera bajas. De esta manera se justifica el hecho de que pocas empresas, el 23%, de las que tienen un Sitio Web, lo mantengan a través de un Web Hosting con otra empresa; ya que más de la mitad considera las tarifas altas, además del notable abstencionismo del 53% que se podría traducir en un poco interés por este tipo de métodos; por lo que se puede concluir que las Medianas Empresas salvadoreñas no consideran viable esta alternativa para el mantenimiento de sus Sitios Web.

PREGUNTA N° 15

¿Estaría interesado en tener presencia en Internet? ¿Por qué razón?

OBJETIVO:

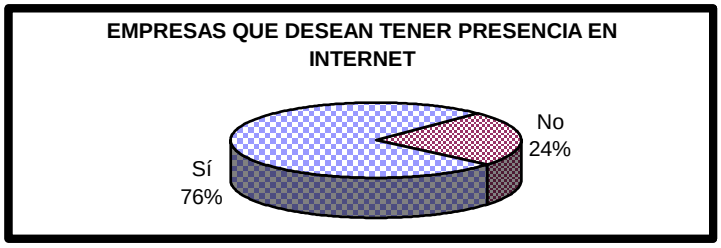
Conocer si las empresas que no poseen un Sitio o Página Web, estarían interesadas en tener presencia en Internet e identificar las razones de dicho interés. (Las empresas pueden seleccionar más de una razón).

RESULTADOS OPCIONES:

TABLA N° 15a

ALTERNATIVA		No DE EMPRESAS	PORCENTAJE	PORCENTAJE TOTAL
a	Sí	32	76%	40%
b	No	10	24%	12%
Total		**42**	**100%**	**52%**

GRAFICA N° 15a

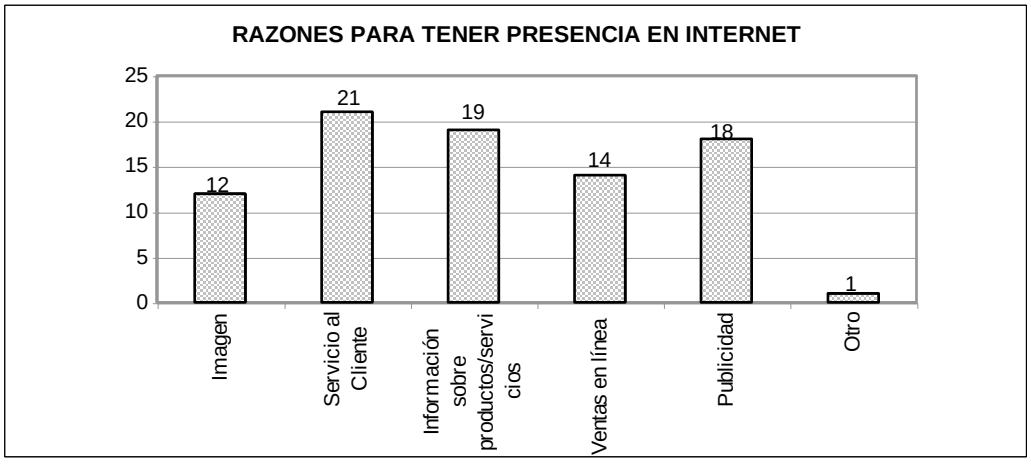


RESULTADOS RAZONES:

TABLA N° 15b

	ALTERNATIVA	FRECUENCIA (f) (No.Empresas que seleccionaron c/opción)	FRECUENCIA RELATIVA (fr) ((f/N)*100)[17]
a	Ventas en línea	14	17%
b	Publicidad	18	21%
c	Información sobre productos o servicios	19	22%
d	Servicio al Cliente	21	25%
e	Imagen	12	14%
f	Otro	1	1%
TOTAL		**N=85 (Σf=N)[18]**	**100%**

GRAFICA N° 15b



[17] El valor del porcentaje es equivalente a la Frecuencia Relativa (f/N)*100, de la Alternativa, siendo f el número de casos por alternativa y N el total de casos obtenidos. Formula tomada del Libro: Estadística I, Gildalberto Bonilla.
[18] La sumatoria de todas las frecuencias o el número de casos por alternativa, es igual al total de casos (Σf=N) . Formula tomada del Libro: Estadística I, Gildalberto Bonilla.

INTERPRETACIÓN DE LOS RESULTADOS:

Del 52% de las empresas encuestadas que no poseen un sitio o Página Web; el 76% manifiesta estar interesado en tener presencia en Internet; el 24% manifiesta que no. Esto indica que existe un mercado potencial, representado por el 40% del total encuestado, que a la hora de elegir un método para mantener su Sitio Web, podría optar por alternativas de código libre. Esto se puede apreciar en la gráfica N° 15a.

Por otra parte, como lo muestra la gráfica N° 15b, de las empresa que no tienen presencia en Internet, el 14% desea tenerla por Imagen; el 25% por Servicio al Cliente; el 22% por Información sobre productos/servicios; el 17% para Ventas en Línea; el 21% por publicidad y el 1% por otras razones. De igual forma que las empresas que poseen un Sitio Web, el mayor interés de las empresa por tener presencia en Internet esta orientado a la satisfacción y el servicio al cliente. Con alternativas que permitan a estas empresas, tener un Sitio Web y les ayude a minimizar los costos de inversión, dichas empresas podrán tener a su disposición nuevos mercados, tanto nacionales como internaciones, esto les permitirá incrementar sus ingresos impulsando su crecimiento económico y tecnológico.

PREGUNTA N° 16

¿Cómo mantendría su Sitio Web o su Página Web?

OBJETIVO:

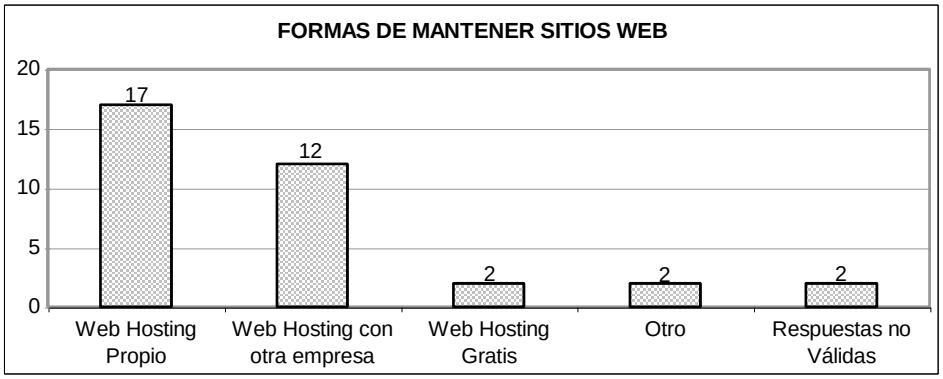
Determinar las formas en que las empresas que no poseen un Sitio Web, mantendrían dichos sitios. (Las empresas pudieron seleccionar más de una opción).

RESULTADOS:

TABLA N° 16

ALTERNATIVA		FRECUENCIA (f) (No.Empresas que seleccionaron c/opción)	FRECUENCIA RELATIVA (fr) ((f/N)*100)[19]
a	Web Hosting Propio	17	48%
b	Web Hosting con otra empresa	12	34%
c	Web Hosting Gratis	2	6%
d	Otro	2	6%
	Respuestas no Válidas	2	6%
TOTAL		**N=35 (Σf=N)[20]**	**100%**

GRAFICA N° 16



INTERPRETACIÓN DE LOS RESULTADOS:

La gráfica N° 16 muestra que el 48% de las empresas optarían por un Web Hosting propio para albergar su Sitio Web; el 34% lo haría contratando los servicios de otra empresa que se dedique a brindar este tipo de servicios; el 6% utilizaría Web Hosting gratuito; el 6% respondieron a la opción de otro; 6% son respuestas no válidas. Como se puede observar más de la mitad de esta empresas tienen la capacidad de invertir en tecnología para albergar su Sitio Web, con equipo propio. Por lo tanto podrían optar por software de código libre para

[19] El valor del porcentaje es equivalente a la Frecuencia Relativa (f/N)*100, de la Alternativa, siendo f el número de casos por alternativa y N el total de casos obtenidos. Formula tomada del Libro: Estadística I, Gildalberto Bonilla.
[20] La sumatoria de todas las frecuencias o el número de casos por alternativa, es igual al total de casos (Σf=N) . Formula tomada del Libro: Estadística I, Gildalberto Bonilla.

servir sus páginas en Internet, esto les ayudaría a minimizar el costo de su inversión.

PREGUNTA N° 17

¿Que tipo de Servidor Web Utilizaría?

OBJETIVO:

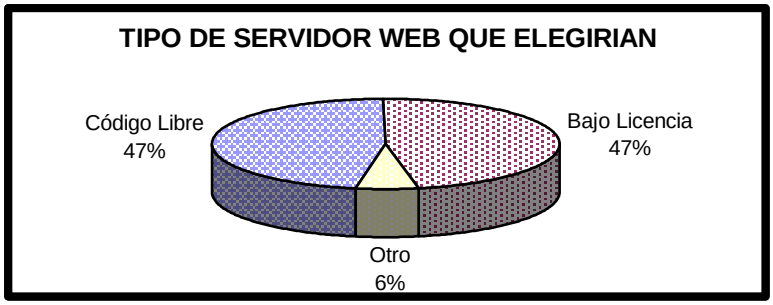
Identificar los tipos de Servidores Web por los que las empresas optarían para servir sus páginas en Internet.

RESULTADOS:

TABLA N° 17

ALTERNATIVA	No DE EMPRESAS	PORCENTAJE	PORCENTAJE TOTAL
Código Libre	8	47%	10%
Bajo Licencia	8	47%	10%
Otro	1	6%	1%
Total	**17**	**100%**	**21%**

GRAFICA N° 17



INTERPRETACIÓN DE LOS RESULTADOS:

Del total de empresas que están interesadas en tener presencia en Internet, el 47% eligieron un Servidor Web de código libre; 47% optaron por un Servidor bajo licencia y el 1% eligieron otra opción. Esto indica que se agrega un 10% de la población total encuestada que se inclina por un Servidor Web de código libre para mantener sus Sitios Web.

2.4 ANÁLISIS DE LOS RESULTADOS

De los resultados obtenidos en la investigación de campo, se determinó que el 93% de las empresas encuestadas poseen una conexión a Internet que utilizan para diversos fines y propósitos, entre los más importantes, con un 67% el Trabajo. Por otra parte, se determinó que el 48% de ellas tienen presencia en Internet a través de un Sitio Web, principalmente para informar acerca de sus productos o servicios. De este porcentaje el 54% posee Web Hosting propio para albergar sus sitios, de las cuales, el 33% utilizan Servidores Web de código libre y el 44% de las misma estaría dispuesto a experimentar con otras alternativas en cuanto a software para servir páginas en Internet se refiere, siendo el 53% de estas el que se inclina por el software de código libre o código abierto.

El 52% de empresas restantes no poseen un sitio en Internet, siendo 76% las que están interesadas en tener presencia en Internet, principalmente para brindar servicio a sus clientes; de ellas el 53% lo haría a través de un Web Hosting propio, y el 47% de este porcentaje, optarían por un Servidor Web de código libre.

El 22% del total de la población encuestada se inclinan por un software de código libre o código abierto para servir páginas en Internet; el 20% por un servidor bajo licencia; el 2% se basarían en un análisis costo-beneficio y el 56% lo conforman las empresas que optarían por otro tipo de método para sus sitios en Internet tal como Web Hosting gratuito y Web Hosting con otra empresa, de igual manera lo conforman las empresas que no están interesadas en tener presencia en Internet y las que no están dispuestas a experimentar con otras alternativas tecnológicas diferentes a las que poseen; finalmente existe un porcentaje de respuestas que se considero no válido, ya que eran incoherentes.

Por otra parte, a través de las entrevistas que se realizaron con el objetivo que las empresas completaran el cuestionario, se pudo observar que existe poco

conocimiento sobre los beneficios de una arquitectura basada en código libre o código abierto tales como menor costo de adquisición, menor costo de mantenimiento, mayor seguridad, tiempos de desarrollo menores. De igual manera, se pudo observar que existe poco conocimiento sobre la implementación de dicha arquitectura y de los productos que se pueden utilizar, de este modo pudo percibirse cierta resistencia al cambio.

Todo esto limita las capacidades de dichas empresas, ya que nuevos productos podrían ayudarles a mejorar el desempeño y la eficiencia de sus actividades laborales reduciendo sus costos de inversión y mantenimiento. Por consiguiente se considera que de alguna manera deberían crearse proyectos encaminados a difundir las ventajas y beneficios de los productos de código abierto o código libre y la forma en que se puede implementar una arquitectura basada en ellos.

Finalmente, se concluye que existe la necesidad urgente de brindar a las Medianas Empresas Salvadoreñas alternativas orientadas a sus capacidades tecnológicas y económicas, a través del diseño y elaboración de un software prototipo para la creación de un Servidor Web, que les permita introducirse en el mundo de Internet para gozar de los beneficios que este y los productos de código abierto proporcionan. De igual manera, beneficiar a aquellas Medianas Empresas que ya tienen una infraestructura tecnológica construida, para que puedan optar por otras alternativas que les ayude a reducir sus costos, permitiéndoles hacer más eficientes sus operaciones y transacciones laborales, así como aumentar la satisfacción de sus clientes, para que de este modo todas ellas puedan aumentar sus ingresos y potenciar su crecimiento económico y tecnológico.

CAPITULO III. DISEÑO DEL PROTOTIPO

3.1 OBJETIVO

Diseño del Software Prototipo para la creación de un Servidor Web, orientado a la Mediana Empresa de El Salvador.

A través de la elaboración de diagramas se describe la estructura interna del software, el funcionamiento del mismo y las interfaces con las que el usuario interactúa.

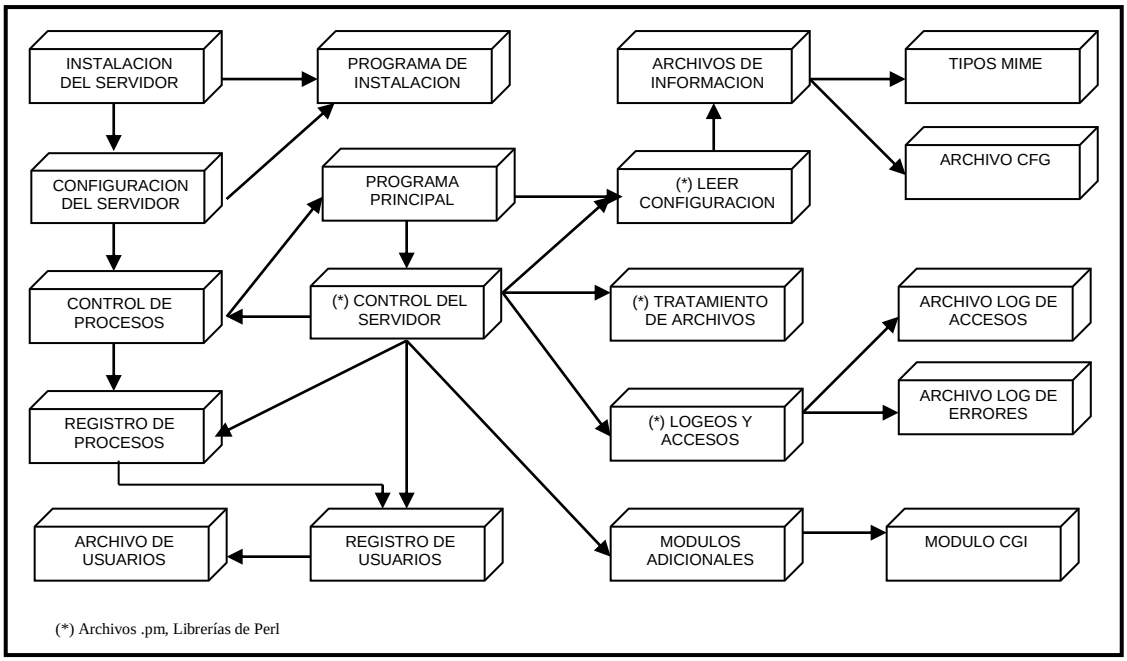
3.2 ESTRUCTURA

El Servidor Web se divide en módulos que realizan una función especifica, de esta manera el software consiste en varias secciones que interactúan a través de llamadas a procedimientos y que integran el servidor en su totalidad.

La programación modular permite agrupar procedimientos, que tienen una funcionalidad común, en módulos separados, lo que permite mantener el orden y el control de los procedimientos que se van ejecutando al igual que facilita la incorporación o adición de nuevas funcionalidades.

Los módulos del servidor los constituyen tanto la parte de instalación, configuración y funcionamiento, así como los módulos para el manejo de funciones o paquetes adicionales del Servidor Web que pueden ser agregados por cualquier otra persona interesada en implementar aspectos no incluidos en este diseño. En la Figura N° 3 se muestra el esquema modular del Servidor Web.

FIGURA N° 3. Esquema modular del Servidor Web.



En la figura anterior se muestra la estructura interna del Software del Servidor Web. En éste se representan los de módulos que interactúan en la instalación, el arranque, el control y la atención de peticiones en el Servidor.

La estructura toma como base las características generales que se consideran en el desarrollo de software de Servidores Web, y que forman parte de la estructura básica de los mismos. De igual manera, se toma en cuenta el aporte de los expertos en el tema entrevistados durante la investigación de campo. Dichas características se han adecuado a las del prototipo que se desarrolla en este capitulo.

3.2.1 DESCRIPCION DE LOS MÓDULOS

1. Instalación del Servidor

Este es el módulo que instala el software para servir las páginas, en el sistema. Consiste de un archivo llamado *setup.pl*, que constituye el programa de instalación, que se ejecuta a través de la opción de Instalar, en la Interfaz gráfica con la que el usuario interactúa, o simplemente se ejecuta desde el directorio donde residen los archivos fuente.

1.1 Programa de Instalación

El programa de instalación consiste en un archivo de código llamado *setup.pl*, elaborado en el lenguaje de Programación Perl. A través de este se realiza la instalación del Servidor Web en el Sistema y la configuración del mismo. Como primer punto el programa solicita al usuario que lo esta ejecutando, valores para establecer rutas y configurar parámetros para el funcionamiento del mismo. Por otra parte, el programa establece valores de configuración a parámetros en los que el usuario no tiene ninguna decisión. Todos estos parámetros se especifican en la sección de funcionamiento del Servidor Web.

Posteriormente, el programa crea los directorios y copia los archivos que el Servidor Web necesita para su funcionamiento en las rutas especificadas por el usuario, al igual que escribe el archivo *server.cfg* en el que se guarda la configuración actual del servidor y que posteriormente se lee para el inicio del mismo.

El programa hace uso de librerías o paquetes incluidos en la distribución de Perl en la que se esta desarrollado el software, tales como *File::Copy* que se utiliza para copiar archivos y manejadores de archivos; *Sys::Hostname* que obtiene el hostname o el nombre de la computadora; *config* para obtener información sobre la configuración del sistema.

2. Configuración del Servidor

Este proceso se realiza a continuación de la Instalación y se lleva a cabo dentro del archivo *setup.pl*. En esta parte es donde se relaciona la información obtenida de la instalación, para crear el archivo *server.cfg*, que almacena toda la información de instalación y de los parámetros configurados que posteriormente son utilizados en el inicio y ejecución de los procesos del Servidor Web. Dicha asociación se realiza a través de variables de configuración definidas en el archivo. Los parámetros que deben ser configurados y que constituyen dichas variables se presentan a continuación.

- La ruta del directorio donde se instala el Servidor Web
- Las rutas de los archivos logs
- El número del puerto de escucha
- La ruta del directorio raíz de archivos
- La ruta de la página de inicio
- Los nombres de los módulos adicionales existentes
- Los manejadores para módulos adicionales (asociación a un tipo MIME)

3. Control de Procesos

Este módulo es el que controla el funcionamiento general del Servidor Web, el inicio, reinicio y finalización del mismo. Esta conformado, básicamente por el programa principal de nombre *sep.pl*, que es un archivo de Perl, dentro del cual se realizan llamadas a librerías, también escritas en Perl y de extensión .pm que es con la cual se guardan los paquetes o librerías en Perl. Dichas librerías leen la configuración actual del servidor, procesan los archivos y atienden las peticiones.

3.1 Programa Principal

Es un archivo elaborado en el lenguaje de programación Perl llamado *sep.pl*, cuya función principal es el control de los procesos que el servidor ejecuta para su funcionamiento. A través de este, se inicia, reinicia y finaliza el servidor.

Al ejecutar el programa, se pasa un argumento, que le indica que proceso es el que debe activar. El programa implementa métodos para ejecutar cada uno de los procesos que controla.

Al inicio del programa, este carga 3 librerías *config.pm, server.pm, server2.pm*; las cuales implementan métodos o funciones para el desarrollo de los procesos en el servidor. A continuación, el programa lee o carga la configuración actual del Servidor Web, a través de la librería *config.pm* para luego proceder a la verificación del argumento recibido desde la interfaz gráfica.

Si dicho argumento es *start*, se ejecuta el inicio del servidor, pasando el control a la librería *server.pm*, que es donde se encuentra la función principal del Servidor Web, la cual crea los procesos e inicia el lazo para la espera de peticiones, inicia las conexiones y envía las respuestas. Durante el proceso, la librería *server.pm*, accesa a los métodos de la librería *server2.pm*, que se encarga de procesar la información para ubicar el archivo solicitado, enviar la respuesta y cerrar la conexión.

Si el argumento que recibe el programa es *restar*, el programa mata o elimina los procesos actuales y procede al inicio del servidor. Si dicho argumento es *stop*, se eliminan los procesos activos y se termina la ejecución del programa, con lo que el servidor queda desactivado.

3.2 Librerías

Perl soporta la Programación Orientada a Objetos (POO). Una clase es una colección de variables y de funciones que acceden a esas variables. Un objeto es una instancia particular de una clase. En Perl, casi todos los módulos o también llamados librerías son, en realidad, objetos que se guardan con la extensión .pm de perl module. Con la distribución de Perl se instalan muchas librerías por default, de igual manera, se pueden crear nuevas según el uso o la función que se pretenda desarrollar.

Para el desarrollo del software del Servidor Web se han implementado 4 librerías: *Config.pm, Server.pm, Server2.pm, Logs.pm*; cada una tiene un propósito especifico, y se complementan entre sí. La programación se basa en los conceptos de utilización e implementaron de clases. Estas librerías se almacenan el directorio *lib*, dentro del directorio que almacena los archivos fuente del software y, se incluyen dentro de un archivo de código a través del uso de la directiva *use nombredepaquete*. Con esto el archivo que se ejecuta puede accesar a cualquier método o función definido en dicha librería. También son de utilidad las directivas *require o include*.

3.2.1 Lectura de Configuración

Esta es la librería *Config.pm* que se ubica en el directorio *lib*, en el directorio de los archivos fuente. Esta desarrollada con el objetivo de cargar la configuración actual del Servidor Web, que se almacena en el archivo *server.cfg* y los tipos MIME que se encuentran especificados en el archivo *mime.types*. Esta información es utilizada por el programa principal *sep.pl* y por la librería de control del servidor *Server.pm*, para el inicio y funcionamiento del servidor. En la librería se han implementado métodos o funciones que desarrollan un proceso especifico como la identificación del nombre del host, identificación de las rutas de los directorios y archivos, identificación de los tipos MIME, la adición e identificación de los

módulos adicionales. Estos métodos se accesan por el programa principal o por las otras librerías, una vez creado el objeto o constructor de la clase, por medio del cual se realiza dicho acceso.

Para leer el archivo de configuración, en el programa se identifican los archivos cuya información se necesita cargar, *server.cgf* y *mime.types*. El acceso a dichos archivos se logra mediante la función *open* incluida en las librerías estándar de Perl, que se utiliza para accesar, ya sea para lectura o escritura, al contenido de archivos. La información leída es almacenada en una variable que pasa su contenido al programa o método que la llamo.

3.2.2 Tratamiento de Archivos

Esta es la librería *Server2.pm*, consiste en un archivo de código escrito en Perl. Es cargada por la librería *Server.pm*, que controla el servidor. La función de esta es procesar la información de la URL para encontrar la ubicación del documento solicitado y enviar la respuesta.

Esta librería implementa dos funciones o métodos que son utilizados por la librería que controla el servidor para realizar el proceso de servir las páginas. La primera función identifica la extensión del documento solicitado y la segunda lo encuentra a partir de información que la URL proporciona. Estas funciones retornan los resultados a la librería de control, para que esta pueda completar el proceso y enviar la respuesta de la solicitud hecha al navegador.

3.2.3 Control del Servidor

Esta es la librería que controla el funcionamiento del servidor. Es un archivo de código llamado *Server.pm*, escrito en Perl y que se ubica dentro del directorio de *lib* en los archivos fuente. Dentro de la misma se cargan las librerías *Logs.pm* y *Server2.pm*, debido a que utiliza métodos definidos en ambas para procesar la

información y enviar la respuesta, y de esta manera las páginas sean desplegadas en el navegador.

El archivo implementa una clase a través de la cual se crean métodos y se llaman a otros definidos en librerías externas, para que el Servidor Web funcione de manera satisfactoria. Los métodos implementados se describen a continuación.

- Método para crear un constructor de la clase, en el que se definen los métodos y se crea el objeto para accesar la clase.
- Método para accesar y cargar el archivo de configuración del servidor, a través de llamadas a los métodos definidos en la librería *Config.pm*, que se carga al inicio, retornando un objeto del tipo de la clase *Config.pm*.
- Se crea un objeto de tipo *HTTP::Response*, que es una librería estándar incluida en la distribución de Perl que consiste en una clase que encapsula respuestas HTTP. Una respuesta HTTP consta de un código de respuesta, las cabeceras y el contenido de la misma. Esta clase posee métodos ya definidos a través de los cuales se devuelve la información de la respuesta, tal como código de respuesta, mensaje de respuesta, protocolo, contenido y la información del header o cabecera del mensaje. El método retorna una variable conteniendo dicha información.
- Método para preparar la respuesta a la solicitud y cargar la página adecuada.
- Método o función principal del Servidor Web que crea el servidor e inicia el lazo para la espera de conexiones. En primer lugar, el método verifica que exista una configuración actual para el servidor, y lo crea a través del uso de la librería incluida en la distribución de Perl, *HTTP::Daemon*, que es una clase que consiste en un servidor HTTP/1.1 que escucha en un puerto por conexiones entrantes. Esta es una subclase de la clase *IO::Socket::INET* con la cual se pueden desarrollar operaciones con los puertos directamente en ellos. Una vez creado el objeto servidor, se accede a los diferentes métodos que ofrece la clase *HTTP::Daemon*,

a través del mismo. A continuación, se espera una conexión entrante, se acepta, se crean los procesos para la conexión, se prepara la respuesta y se encuentra la página apropiada, se envía la respuesta y se guarda la información de logeo ya sea de acceso o errores en el archivo correspondiente. Para llevar a cabo todos estos procedimientos se utilizan métodos, definidos tanto internamente, es decir de la misma clase, como externamente en las librerías que se cargan al inicio de la misma.

- Método para cargar la página solicitada o enviar un mensaje de error en caso de no encontrarla. Este identifica el nombre del directorio raíz y el nombre de la página por default a través de la librería que accesa el archivo de configuración y busca en el directorio raíz, la página solicitada o la página por default. Se verifica la extensión del archivo y el tipo de archivo, se abre y se lee el contenido, se envía la respuesta y finalmente se cierra la conexión.

3.2.4 Logeos y Accesos

Es una librería de código escrita en Perl de nombre *Logs.pm*, cargada desde la librería de control del servidor, con el objetivo de registrar toda la información relacionada con los accesos al servidor y los errores que ocurren durante el proceso. La información se almacena en dos archivos de extensión .log, *access.log* y *error.log* respectivamente. La librería implementa métodos para accesar y escribir en dichos archivos.

4. Registro de Procesos

Este módulo es donde se registran los procesos que el servidor realiza así como los errores ocurridos durante los mismos. Esta información se guarda en dos archivos *access.log* y *error.log*, dentro de la carpeta *logs* en el directorio de los archivos fuente.

El registro de los procesos se realiza cuando el programa principal pasa el control a la librería *Server.pm*. Dentro de esta librería, al ejecutarse un proceso tal como

una petición de una Página Web; al enviar la respuesta se registra en el archivo correspondiente la información sobre la acción ejecutada y los resultados. Este registro se hace a través de funciones definidas en la librería de logeos y accesos, *Logs.pm*, que es un archivo de código escrito en Perl. La librería de control del servidor, *Server.pm*, carga esta librería y accesa a las funciones que leen y escriben en los archivos *logs* registrando la información sobre las peticiones y los errores ocurridos y por otra parte, verifican si el modo de protección esta activado.

5. Registro de Usuarios

En este módulo se realizan todos los procesos para registrar usuarios del Servidor Web y, activa y desactiva el modo de protección contra usuarios no registrados. El registro de los usuarios se realiza desde la opción de Usuarios, en la interfaz gráfica. Simplemente se especifica un nombre para el usuario y un password, esta información se guarda en un archivo de texto plano *usuarios.txt*. Este archivo es accesado por el la librería de *Server.pm*, para verificar si un usuario esta o no registrado y si por lo tanto, tiene acceso al contenido del Servidor Web cuando el modo de protección esta activado.

La activación y desactivaron del modo de protección se lleva a cabo desde las opciones ACTIVAR MODO y DESACTIVAR MODO, en la interfaz gráfica. Al activar cualquiera de las dos opciones, se lee la primera línea del archivo *usuarios.txt*, en la que se encuentra una bandera o variable que indica el estado de dicho modo. Si la bandera es igual a *A* el modo esta desactivado, si tiene el valor de D esta activado.

5.1. Archivo de Usuarios

Es un archivo de texto llamado *usuarios.txt* en el que se almacena la información sobre los usuarios registrados. Consta de dos columnas la primera es el nombre de usuario y la segunda el password. Este archivo posee una bandera que indica

si el modo de protección se encuentra activado; por default el modo esta desactivado y dicha bandera posee el valor de D. Cada vez que se activa o desactiva el modo protección se lee la primera línea de este archivo, para conocer el modo actual o para cambiarlo. El formato del archivo es el siguiente:

```
modo=D              # bandera modo protección
admon  pass         #columna de usuarios y passwords
prueba  pass2
```

6. Archivos de Información

Consisten en archivos en texto plano que almacenan información de configuración del Servidor Web. Estos son accesados por el programa principal y por las librerías para llevar a cabo los procesos que el Servidor Web ejecuta. Son dos archivos, el que almacena los tipos MIME y el que guarda los parámetros de configuración actual del Servidor Web.

6.1 Tipos MIME

Es el archivo *mime.types* y controla que tipos de datos de Internet se envían al cliente para obtener una o varias extensiones de archivo, de esta manera tanto el servidor como el navegador pueden saber como manejar y presentar el contenido del archivo solicitado. Este archivo es accesado por la librería de lectura de configuración *Config.pm*, para asociar una extensión de archivo, a un tipo de archivo y su manejador correspondiente en caso de que sea necesario. El archivo consta de dos columnas la primera es la del tipo MIME y la segunda la extensión asociada al mismo. El formato del archivo se muestra a continuación.

```
# MIME type              Extensión
application/zip          zip
application/x-javascript js
image/gif                gif
image/jpeg               jpeg jpg jpe
text/html                html htm
text/plain               asc txt
```

6.2 Archivo CFG

Este es el archivo *server.cgf* que almacena toda la información relacionada con la configuración actual del Servidor Web. Es un archivo de texto que contiene la siguiente información:

- La ruta de los directorios del servidor
- Las rutas de los archivos logs, acceso.log y error.log
- El número de puerto de escucha del servidor
- El nombre de la computadora o el hostname donde se instala el servidor
- La ruta del directorio raíz de archivos
- La ruta de la página de inicio
- Los módulos adicionales definidos en la carpeta de módulos
- Los manejadores de archivo para los módulos y la asociación a un tipo de extensión o tipo MIME.

Este archivo se lee a través de la librería de lectura de configuración, que es cargada por el programa principal y por la librería de control del servidor, para obtener los parámetros de configuración que ambos utilizan en la ejecución de procesos. El archivo es generado por el programa de instalación. A continuación de la instalación del servidor, el programa toma los parámetros capturados y los asocia con la información que se necesita configurar a través de las variables de configuración establecidas, para posteriormente, abrir y escribir dicho archivo. El formato del archivo es el siguiente.

```
# Archivo de configuración del Servidor Web 1.0
# Los comentarios comienzan con el símbolo de número

# Directorio raíz del servidor
ServerRoot       /usr/bin/swp

# Ruta de archivos de logeo
AccessLog __ServerRoot__\logs\accesso.log
ErrorLog  __ServerRoot__\logs\error.log
```

```
# Número de Puerto del servidor
ServerPort 80

# Hostname o nombre de la computadora
pavilion

# Ruta Directorio raíz de documentos
DocumentRoot __ServerRoot__\html

# Ruta de Archivo de Inicio
DirectoryIndex index.html

#  Agregar módulos bases
AddModule mod_cgi

# Definir Manejadores para archivos de extensión
AddType cgi_script     cgi pl
```

7. Archivos log

Son dos archivos en los que se registra toda la información de logeo y accesos relacionada con los procesos que el Servidor Web ejecuta.

7.1 Archivo log de acceso

Es un archivo en texto plano llamado *access.log*. En este se registra la información de acceso al Servidor Web cuando un cliente solicita un documento o Página Web. Consta de 3 columnas de información y se utiliza una línea por cada acceso al servidor, es decir si un documento posee una página y dos imágenes, en el archivo se guardan 3 líneas. La primera columna de información es la fecha del acceso en el formato *[nombredia, dia mes anio]*, *nombredia* y *mes* están formados por 3 letras, el año de 4 dígitos. La segunda columna almacena el tipo de petición ya sea GET o POST. Finalmente la tercera es la ruta del archivo accesado. El formato de dicho archivo se muestra a continuación.

```
#col1                    col2  col3
[Jue, 21 Ago 2003 ] GET /images/pws_logo.jpg
[Jue, 21 Ago 2003 ] GET /html/readme.html
```

7.2 Archivo log de errores

Al igual que el anterior, es un archivo en texto plano de nombre *error.log*. Este archivo registra la información sobre los errores que ocurren tanto al accesar el servidor, como los ocurridos durante del desarrollo de los procesos de atención de peticiones. Consta de una línea por error, en la que guarda el mensaje de error y la ruta del archivo o documento en el que ocurrió. Al igual que el archivo anterior se utiliza una línea por cada acceso al servidor. El formato del archivo es el siguiente:

Falló al abrir la página: public. html
Error cargando módulo: mod_day no se encuentra el método at (eval 5) line 3.

8. Módulos Adicionales

Estos son módulos que implementan nuevas funciones al Servidor Web, es decir, no están incluidos en la distribución original del mismo. Estos módulos residen en la carpeta de nombre *modulos*, que se ubica dentro de la carpeta *lib*, en la ubicación donde residen los archivos fuente del Servidor Web.

Estos módulos consisten en archivos de código guardados bajo la extensión .pm, es decir son módulos o librerías de Perl, a través de los cuales el Servidor Web puede extender sus funciones abarcando aspectos que no se han considerado en este diseño. Este tipo de diseño modular hace la expansión del servidor fácil y flexible, ya que no se tiene que modificar los programas principales bajo los que el servidor funciona, simplemente se copia el archivo .pm en la carpeta *modulos*, se edita el archivo *server.cfg* en la sección de *modulos* donde se escribe el nombre del módulo agregado. A continuación se vuelve a ejecutar el programa de instalación, para que de esta manera, el servidor reconozca el módulo agregado.

8.1 Módulo CGI

Es un archivo de extensión .pm (perl module) que implementa el manejo de CGI (Common Gateway Interface). Lo que normalmente se conoce por CGI son pequeños códigos de programa que se adaptan al estándar Common Gateway Interface (CGI) mediante el cual se puede acceder a servidores de Internet que envían información a los usuarios.

Perl interactúa con el usuario o con el sistema operativo por medio de entradas salidas que permiten el intercambio de información. Este intercambio de datos se gestiona por medio de operadores específicos que configuran una interfaz entre el script y su entorno. Los archivos se consultan mediante la rutina *open* que admite dos argumentos, un puntero de archivo y un nombre de archivo. La rutina *open* de Perl puede utilizarse también para ejecutar mandatos del shell, modificar sus entradas y recuperar sus salidas.

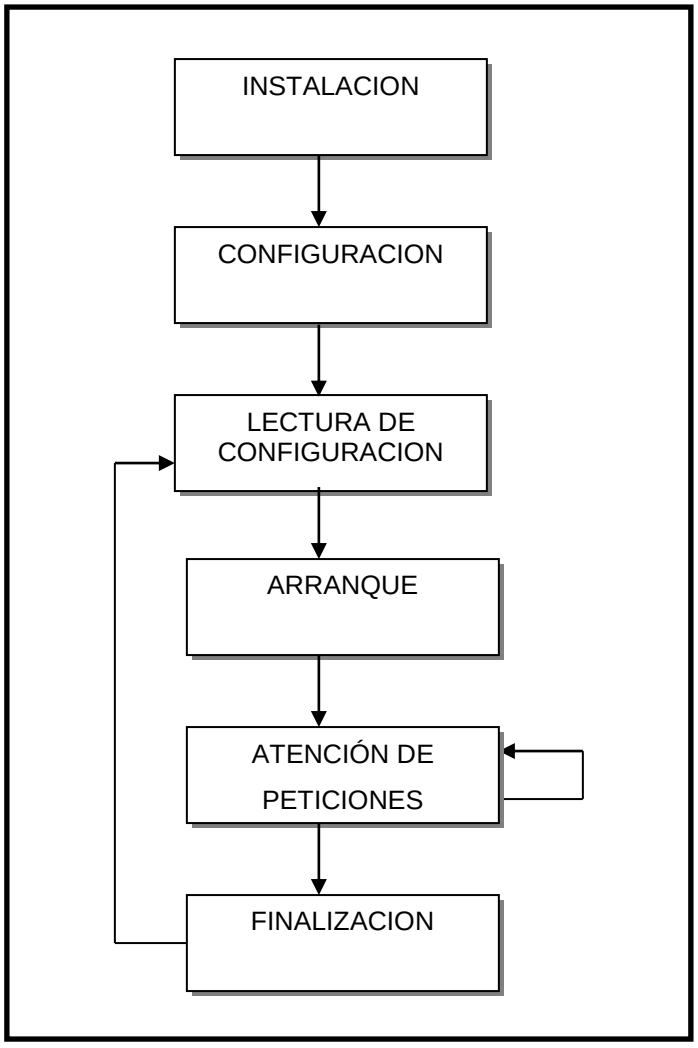
Cuando el nombre del archivo lleva como prefijo el carácter |, este nombre se trata como un mandato. Este mandato se ejecuta y comunica con el script Perl mediante la rutina *print.* Esto es lo que se denomina como *pipe* o *tubería* y es el concepto que se utilizara para ejecutar un script cgi.

Se utiliza el interprete de perl, específicamente la librería estandar de perl IPC::Open3 para ejecutar los script cgi, es decir que el Servidor Web únicamente interpreta los script de extensión .pl y .cgi. Dicha librería es utilizada para abrir canales de comunicación para la ejecución de programas diferentes al actual.

3.3 FUNCIONAMIENTO

El ciclo de vida del servidor esta formado por 6 procesos: Instalación, Configuración, Lectura de configuración, Arranque, Atención de Peticiones, Finalización. Esto se ilustra en la Figura N°4.

FIGURA N° 4. Ciclo de Vida del Servidor Web.



1. INSTALACION

La instalación del Servidor Web es el primer proceso en su ciclo de vida; se lleva a cabo a través del programa de instalación *setup.pl*. Este programa se ejecuta desde una ventana de consola. Al ejecutar el archivo, el usuario tiene que introducir valores o la información que configura los parámetros necesarios para que el servidor funcione correctamente. Por default el programa sugiere valores, sin embargo, el usuario puede introducir los que crea conveniente.

Por otra parte, el programa configura o setea parámetros necesarios de manera transparente al usuario. La información que se configura se describe a continuación. Los valores entre corchetes [ ], son los que el usuario introduce.

- La ruta en la que el Servidor Web se instala: */usr/bin/Perl/swp*. Esta es la ruta en la que se instalan los todos los archivos y directorios que el Servidor utiliza.
- El puerto de escucha del Servidor: [*80*]. El valor numérico del puerto en el que el Servidor trabaja, por default es el puerto 80.
- El nombre de dominio del Servidor: [*Hostname*]. El dominio bajo el que se registra el Servidor, por default es el nombre de la computadora en la que se esta instalando el software.
- La ruta del directorio raíz: *ruta_servidor /html*. Es la ruta en la cuál se buscan los documentos requeridos o solicitados.
- El nombre de la página de inicio: *index.html*. Se encuentra dentro del directorio html, y es la página de inicio o la página por default del Servidor.
- La ruta del archivo de registros de accesos y logeos: *access.log*. Ruta del Archivo en el cual se registran la toda información sobre los accesos al Servidor Web tal como la fecha, el tipo de petición y la ruta del documento solicitado.

- La ruta del archivo registro de errores: *error.log*. Ruta del Archivo que registra los errores en peticiones hechas al servidor.
- Especificación de los Módulos adicionales. Se especifican los nombres de los módulos que el servidor incorpora y que debe cargar cuando se inicie.
- Asociación de tipos a los módulos adicionales. Se especifica la asociación de los tipos extensión de los módulos adicionales a una extensión o tipo de archivo ya considerado dentro de los tipos MIME.

Una vez introducida esta información el programa procede a crear los directorios y copiar los archivos necesarios en el sistema, de acuerdo a las rutas especificadas en el paso anterior. Los directorios que se crean son los siguientes.

- Directorio Raíz del Servidor Web, *swp*: es donde se instala el servidor y donde residen todos los archivos que este utiliza para su funcionamiento.
- Directorio de Librerías, *lib*: es donde residen las librería desarrolladas para el funcionamiento del Servidor Web, este se ubica dentro del directorio raíz *swp*.
- Directorio de Módulos Adicionales, *modulos*: en este se ubican todos los módulos adicionales que se van agregando al Servidor Web para incorporar nuevas funciones, esta ubicado dentro de el directorio de librerías *lib*.
- Directorio de logeos y accesos, *logs*: este se ubica en dentro del directorio raíz, y almacena los archivos de tipo log que registran la información de logeo y acceso.
- Directorio raíz de documentos, *html*: es el directorio donde se encuentran los documentos o páginas que el Servidor Web almacena, se ubica dentro del directorio raíz.
- Directorio raíz de imágenes, *imagenes*: en el se almacenan las imágenes de los documentos o páginas que el Servidor Web almacena, se ubica dentro del directorio raíz de documentos.

Los archivos que se copian durante el proceso son los siguientes:

- *setup.pl*: Archivo de código en Perl que instala el software del Servidor Web en el sistema. Se ubica dentro del directorio raíz del servidor.
- *sep.pl*: Archivo de código en Perl que se encuentra en el directorio raíz del servidor y controla el funcionamiento del servidor. Es el programa principal a través del cual se ejecutan todos los programas del servidor.
- *Config.pm*: Es una librería de Perl desarrollada para leer o accesar a la configuración actual del Servidor Web. Se encuentra ubicado en el directorio *lib*.
- *Server.pm*: Librería de Perl que implementa las funciones que crean el servidor, lo controla y ejecuta los procesos para servir las páginas. Se ubica dentro del directorio *lib*.
- *Server2.pm*: Consiste en una librería de Perl que es accesada por la librería de *server.pm* para procesar la información de la URL con el objetivo de identificar el tipo de archivo y ubicarlo dentro del servidor. Se encuentra en el directorio *lib*.
- *Logs.pm*: es una librería de Perl que procesa y registra la información de los accesos y logeos al servidor, así como la información sobre los errores que ocurren durante los procesos que el servidor ejecuta. Se encuentra dentro del directorio *lib*.
- *mime.types*: Es un archivo de texto plano que almacena información sobre los tipos de datos de Internet que se pueden enviar por el servidor. Se ubica en el directorio raíz del servidor
- *server.cfg*: Consiste en un archivo de texto plano que contiene la configuración actual del Servidor Web. Se encuentra ubicado en el directorio raíz del servidor.

- *Readme.txt*: Es un archivo de texto que se ubica en el directorio raíz del servidor y que almacena información sobre el software, contiene indicaciones de cómo instalar y ejecutar el Servidor Web.
- *index.html*: Es la Página Web de inicio del servidor. Se ubica dentro del directorio raíz de documentos.
- *logo.jpg*: Es un archivo de imagen que representa el logotipo del Servidor Web. Este se muestra en la página de inicio y se ubica en el directorio raíz de *imagenes*.
- *usuarios.txt*: Consiste en un archivo en texto plano en el cual se registra la información sobre los usuarios autorizados para accesar a la información del Servidor Web. Se ubica dentro del directorio.
- *passw.html*: Es la página de autenticación del Servidor Web, en la que el usuario tiene que introducir su nombre de usuario y su password en el caso de que el modo protección este activado. Se ubica en el directorio *html*.
- *mod_cgi*: Módulo que implementa el manejo de CGI. Se ubica en el directorio *Modulos*.

Existen directorios y archivos que no se copian durante el proceso de instalación, debido a que se crean automáticamente cuando se les invoca por primera vez. Estos archivos son los siguientes:

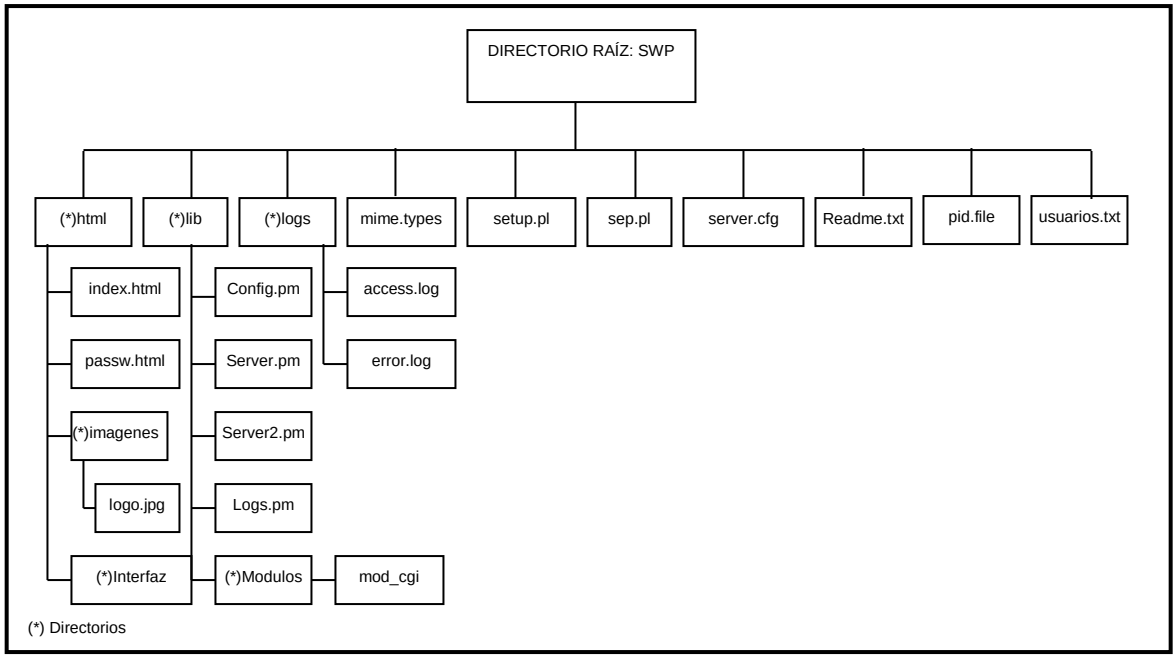
- *access.log:* Consiste en un archivo en texto plano en el cual se registra la información sobre los accesos al Servidor Web. Se ubica dentro del directorio *Logs*.
- *error.log:* Consiste en un archivo en texto plano en el cual se registra la información sobre los errores que ocurren durante el Servidor Web se ejecuta. Se ubica dentro del directorio *Logs*.

- *pid.file*: es un archivo de texto plano, que almacena el pid o el id del proceso en el sistema operativo, que corresponde a el proceso que esta ejecutando el Servidor Web.

Por otra parte, el Directorio Interfaz, que es en el que residen los archivos que conforman la Interfaz Gráfica para la administración del software, debe ser copiado manualmente a la ruta donde se instala el directorio raíz.

La estructura de los directorios y archivos que forman parte del software del Servidor Web se ilustra en la Figura N°5.

FIGURA N° 5. Estructura de Directorios y Archivos del Servidor Web.



Después de la creación de los directorios y copia de los archivos, el programa de instalación procede a configurar el Servidor Web en base a los parámetros obtenidos de esta parte.

2. CONFIGURACION

Una vez instalados los directorios y los archivos necesarios, el programa de instalación *setup.pl* inicia el proceso de configuración del Servidor Web que se registra en el archivo *server.cfg*. dicho archivo posee una estructura definida sobre los valores o variables de configuración que se deben establecer.

El programa de instalación contiene una sección al final del mismo, en la que se asocian los parámetros ingresados por el usuarios y los parámetros ya establecidos por el programa, a las variables de configuración del archivo *server.cfg*. Dichas variables se han colocado en el programa de instalación precedidos de la etiqueta <DATA>, por medio de la cual se reconoce que la parte de asociación inicia.

Para realizar tal asociación, se ha creado dentro del programa, una rutina que abre el archivo *server.cfg*, busca la línea del programa con la etiqueta <DATA> y crea un lazo que lee cada línea y la escribe en el archivo cfg hasta que encuentre el fin de archivo. Al finalizar esta rutina el servidor esta configurado y listo para iniciar.

En caso de error o interrupción de esta operación, el archivo *server.cfg* se puede copiar directamente al directorio raíz del servidor y editar con la información correspondiente.

3. LECTURA DE CONFIGURACION

La lectura de la configuración actual del Servidor Web, es un proceso interno que se ejecuta a través de métodos desarrollados en la librería *Config.pm*, la cual es invocada por el programa principal, antes de iniciar el Servidor Web, o por la librería de control para ejecutar sus procesos. Consiste básicamente, en abrir los archivos que guardan la configuración e interpretar las líneas presentes en dicho

archivo almacenando esta información en una variable, que se retorna, para que los métodos de otras librería puedan ejecutar los procesos correspondientes.

En el programa se especifican los nombres de los archivos que contienen la información de configuración, *server.cfg* y *mime.types*. El proceso de esta librería consiste abrir el archivo de configuración para lectura utilizando la función *open()* de Perl, e iniciar un lazo que lo recorre línea a línea, identificando si es de comentario o no para ignorarla o procesarla. El archivo de configuración contiene tags o etiquetas que indican el contenido de valor almacenado. Por ejemplo la línea donde se localiza el número de puerto de escucha especifica: *ServerPort 80*, donde *ServerPort* es la etiqueta y *80* el valor.

De esta manera, el programa puede distinguir el valor del parámetro y saber a que corresponde. Los parámetros leídos o identificados se almacenan en un arreglo asociativo que consta de una parte que es la etiqueta o nombre del parámetro, y otra que es el valor de dicho parámetro, de esta forma cuando el programa principal o la librería de control necesita accesar a ellos, simplemente accesa a dicho arreglo.

Finalmente cuando se encuentra el fin de archivo, este se cierra y retorna la variable con los valores de los parámetros obtenidos al programa o la librería que lo invoco. La misma operación se ejecuta para cargar los tipos MIME, la diferencia es que el procedimiento se ejecuta en una función distinta y que este archivo no posee tag o etiquetas, sino que esta formado por dos columnas, lo que facilita el acceso al mismo. Una vez el programa principal ha obtenido y establecido los parámetros de configuración puede proceder a iniciar el Servidor Web.

4. ARRANQUE

El proceso de arranque o inicio del Servidor Web se lleva a cabo en el programa principal *sep.pl*. Dicho proceso se ejecuta desde una consola o ventana de comando. Al realizar esta acción, se ejecuta dicho programa y se pasa el argumento *start*, que le indica que el proceso que debe ejecutar, es el de inicio.

Para iniciar el servidor, el programa crea un objeto para el acceso a la librería de configuración. Con dicho objeto el programa principal puede accesar a los métodos de la Librería de configuración y cargar los parámetros, en el momento en que sea necesario. Esta operación retorna un objeto de tipo *config*, en el que se almacenan los resultados obtenidos, que identifican las rutas de los directorios y el nombre de la computadora sobre la que se ejecuta el servidor, al igual que el número de puerto de escucha.

A continuación, se verifica el argumento que recibe el programa al ser ejecutado, si es *start*, significa que el servidor debe ser iniciado; lanzando una llamada al sistema para crear un nuevo proceso a través de la función *fork* de las librerías estándar de Perl, que realiza llamadas para crear procesos a partir de un proceso padre existente. Posteriormente, se crea un objeto para accesar a la librería de control del servidor *Server.pm*. Por medio de este objeto se invoca al método que en dicha librería crea el puerto e inicia el lazo de espera de conexiones.

Una vez invocado el método, se imprime un mensaje en una ventana en modo consola, indicando que el servidor esta iniciado y listo para la espera de peticiones. Dicha ventana permanece abierta mientras el servidor esta levantado.

5. ATENCION DE PETICIONES

Después del proceso de inicio, el Servidor Web queda bajo el control de la librería *Server.pm*, que implementa la función principal que crea el servidor y atiende las peticiones. Esta librería crea constructores u objetos para accesar a la librería de configuración y a los métodos desarrollados dentro de la misma clase. Dentro de esta librería de control se ha desarrollado el método *server()* que constituye la función principal del software del Servidor Web.

En dicho método se crean dos objetos de las librerías estándar de Perl, que se utilizan para la creación del servidor y el envío de las respuestas al cliente. Uno es de la clase *HTTP::Daemon*, la cual se utiliza para crear un servidor HTTP/1.1 que escucha en un puerto por conexiones entrantes. El segundo objeto es de la clase *HTTP::Response*, que encapsula respuestas o mensajes HTTP. El objeto de la clase *HTTP::Daemon* se crea con el proceso que se presenta a continuación.

```
require HTTP::Daemon;
$server = new HTTP::Daemon
 (LocalPort => 80,
 LocalAddr => 'localhost',
 Listen => 5 );
```

En el segmento de código anterior, se construye el servidor que debe escuchar en un puerto. La instrucción requiere *HTTP:::Daemon*, incluye la clase en la librería de control *Server.pm*. Con la sentencia new *HTTP::Daemon* se crea el objeto servidor y se asigna a la variable *$server*, con la cual se hace referencia al mismo en todo el programa y se accede a los métodos de la clase *HTTP::Daemon*. Los valores entre paréntesis son los parámetros o argumentos que se pasan para la creación del objeto. Su significado se interpreta de la siguiente manera:

- LocalPort: El Puerto de escucha del Servidor Web , su valor depende de la configuración actual del mismo. Para asignar dicho valor se accesa a librería de lectura de configuración a través del objeto creado al inicio de la clase.

- LocalAddr: La dirección de enlace o nombre de la computadora donde se ejecuta el Servidor Web. Al igual que el parámetro anterior, para asignar el valor se accesa a librería de lectura de configuración.
- Listen: Es el tamaño de cola para las conexiones.

Posterior a la creación del servidor, se inicia el lazo para la espera de conexiones entrantes y su aceptación, a través del método *accept()* de la clase *HTTP::Daemon*, el cual retorna un valor cuando una conexión con un cliente esta disponible. El objeto que retorna *accept()* se asigna a una variable que hace referencia a la conexión establecida con el cliente.

A continuación se establece el *timeout* o el tiempo de espera para las conexiones; se leen los datos del cliente quien retorna un objeto *HTTP::Request*, que es una clase estándar de Perl que encapsula peticiones HTTP. El objeto que retorna indica si la conexión con el cliente fue satisfactoria o ha fallado y hace referencia a el objeto solicitado, para el caso una Página Web.

Si la conexión es satisfactoria, se prepara la respuesta a la petición y se encuentra la página o documento adecuado. La respuesta esta formada por un código de respuesta, una cabecera o head, la URL y el contenido. La cabecera contiene información sobre puerto de escucha, el nombre del host, es decir información de configuración. La URL es el path que se recibe en la petición. El contenido es la Página Web.

Para encontrar la página adecuada se invoca a la función *load_page()*, que se ha implementado dentro de la librería de control y que recibe como parámetros, la información de la respuesta enviada al cliente para establecer la petición, ya que en ella se encuentra los datos de la URL, que básicamente es utilizada para encontrar el archivo. En dicha función como primer paso, se identifica la

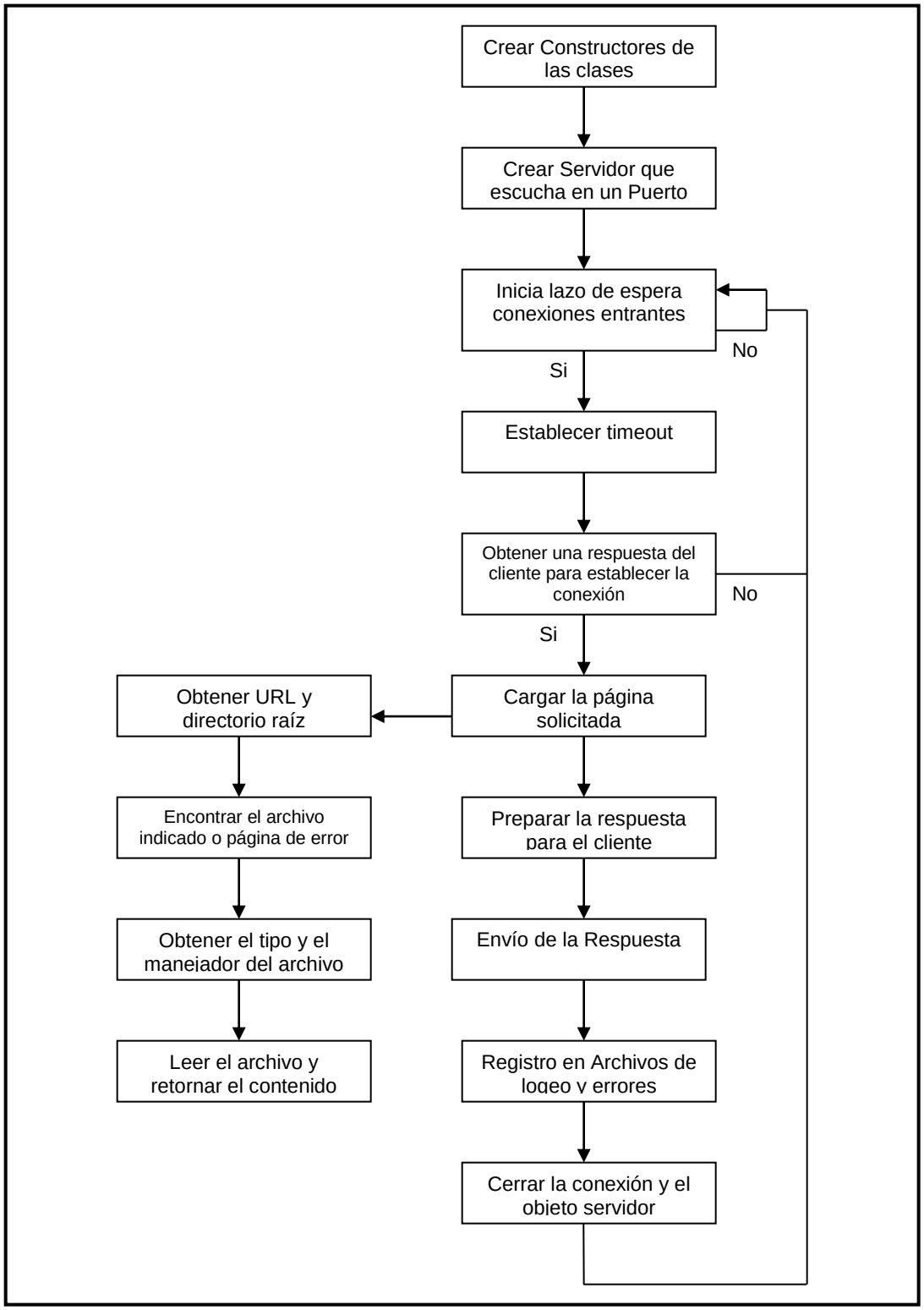
información sobre la URL y se obtiene del archivo de configuración, el directorio raíz de documentos, en el que se debe realizar la búsqueda.

Dicha búsqueda se realiza a través de la función *find_file()*, implementada en la librería de tratamiento de archivos *Server2.pm*, a la que se le pasan la URL y el directorio raíz como parámetros. Dicha búsqueda se realiza obteniendo de la URL, la parte que corresponde al nombre del archivo que se compara con los nombres de los archivos del directorio raíz de documentos. Al encontrarlo, la función retorna el nombre del archivo.

Una vez encontrado el archivo, se obtiene la extensión del mismo, para saber su tipo y buscar el manejador correspondiente. A continuación se abre el archivo, se lee su contenido y se almacena en una variable que se devuelve a la función de preparación de respuesta. En caso de no encontrar el archivo se despliega una página indicando que la página solicitada no ha sido encontrada. Localizado el archivo y preparada la respuesta, se envía a través del método *send_response()* de la clase *HTTP::Daemon*, que envía un objeto como una respuesta a una conexión establecida con un cliente.

Posteriormente, se escriben los archivos de logeo, con la información correspondiente, la de acceso si se logro enviar la respuesta o de error si el proceso fue fallido. A continuación se cierra la conexión y se cierra el objeto *$server*. El proceso de Atención de Peticiones se ilustra en la Figura N° 6.

FIGURA N° 6. Proceso de Atención de Peticiones del Servidor Web.



6.FINALIZACIÓN

La finalización del Servidor Web se realiza en el programa principal *sep.pl*, desde una ventana de comando o consola. Cuando se ejecuta la orden, se invoca el programa principal enviándole el argumento *stop*, que indica que el servidor debe ser detenido.

Al recibir este argumento, el programa elimina todos los procesos actuales a través de la función *kill()* de las librerías estándar de Perl, y el programa finaliza. Con estas acciones el Servidor Web deja de realizar sus funciones y queda detenido.

3.4 DISEÑO DE LA INTERFAZ DEL SERVIDOR WEB

3.4.1 REQUERIMIENTOS

El diseño de la Interfaz para la administración y configuración del Software del Servidor Web consta de los siguientes requerimientos:

1. Presentación
2. Instalación del Software
3. Configuración del servidor
4. Inicio del servidor
5. Reinicio del servidor
6. Finalización del servidor
7. Administración de Usuarios: adición, consulta y eliminación
8. Definir la protección contra usuarios no deseados o prohibidos

Estas son las opciones básicas que la interfaz con la que el usuario interactúa, debe incluir para administrar y operar el Servidor Web. Las opciones de instalación, configuración, inicio, reinicio y finalización del Servidor Web, se

realizan en una interfaz de modo texto, en una terminal o ventana de consola. La administración de usuarios y la definición del modo de protección se realizan en una interfaz grafica, a través de un Sitio Web.

3.4.2 DESCRIPCION DE LA INTERFAZ EN MODO TEXTO

1. Instalación y Configuración del Software

La instalación y configuración del Servidor Web 1.0 se realiza a trabes del srcipt llamado *setup.pl*, este se ejecuta desde una ventana de consola ejecutando la orden siguiente:

```
perl /root/swp/swp/setup.pl
```

La ejecución de la orden y los resultados se muestran en la Figura N° 7.

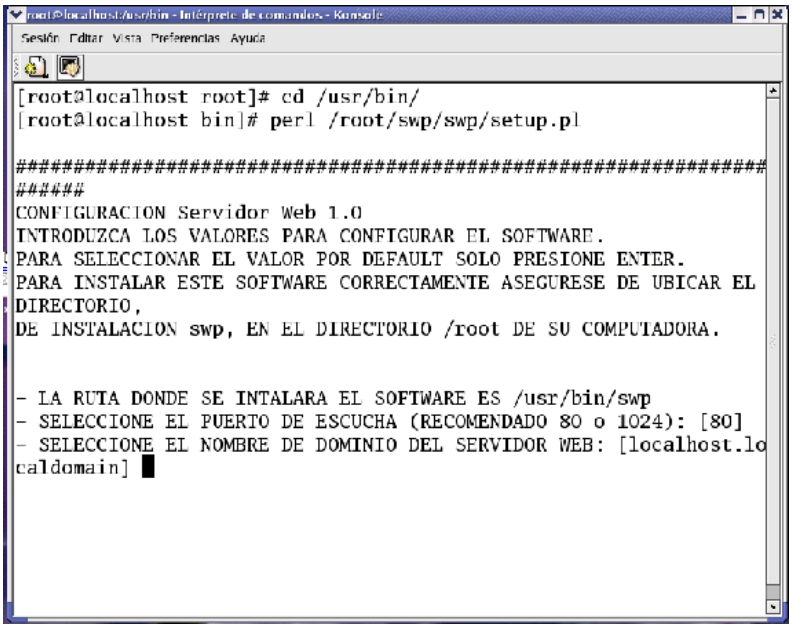


Figura N° 7. Proceso de Instalación, solicitud de parámetros.

Como se puede observar, en la ventana se indica al usuario que debe introducir tres parámetros para proceder. Los parámetros solicitados son los siguientes:

- La ruta donde se instala el software del Servidor Web.
- El puerto de escucha del servidor
- El nombre del dominio del servidor o nombre de la computadora.

Posterior a la introducción de los parámetros solicitados el programa de instalación procede a crear los directorio y a copiar los archivos necesarios para el funcionamiento del Servidor Web. A continuación se procede a la configuración del Servidor Web. En esta parte es donde se asocian los parámetros introducidos por el usuario con los parámetros establecidos en el archivo de configuración server.cfg. Si por algún motivo no el programa no pudiera actualizar el archivo, aparecería un mensaje de error indicando que ha ocurrido un error y que debe editar dicho archivo manualmente. En la ventana de consola aparece lo siguiente.

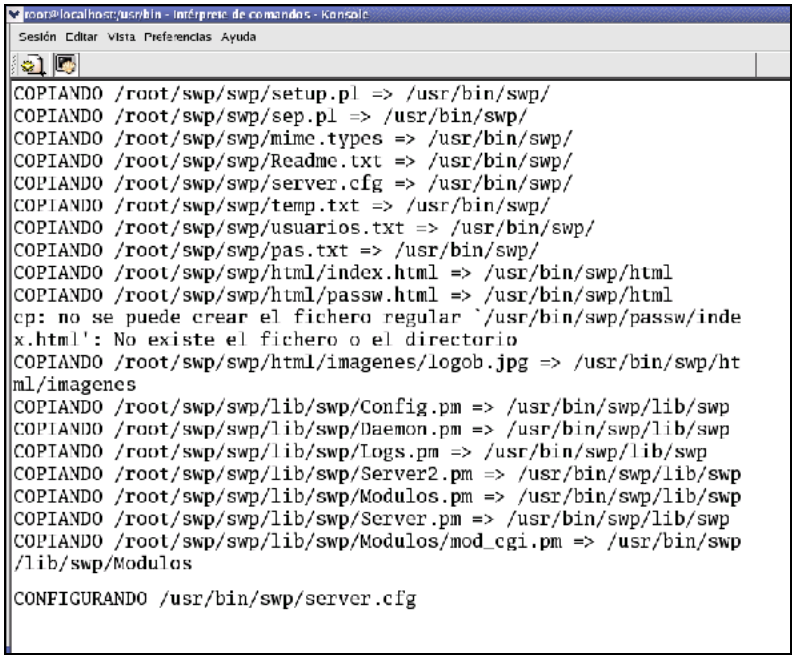
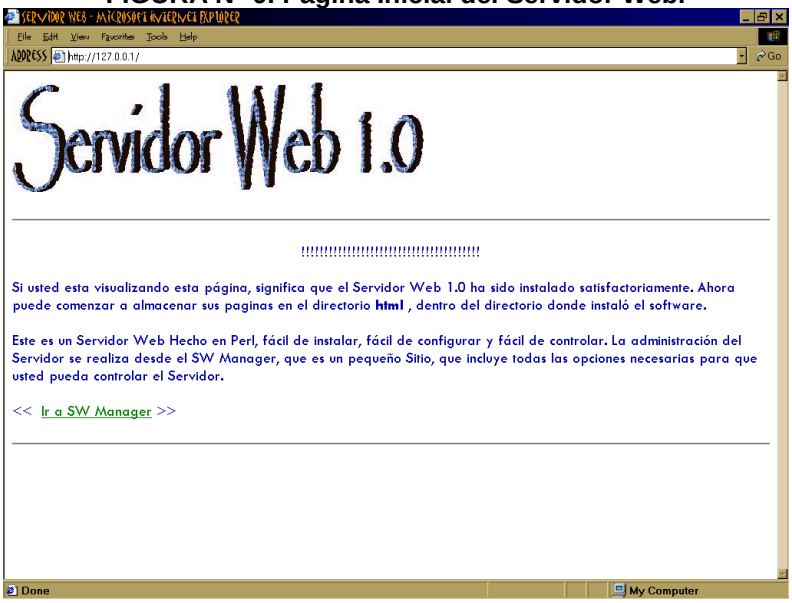


Figura N° 8. Proceso de Instalación, copia de archivos y configuración.

Si la instalación y configuración del Servidor Web se llevó a cabo de manera satisfactoria, al abrir el navegador y escribir en la barra de direcciones

http://127.0.0.1, se visualizará en él, la página de inicio del Servidor Web. Esta se muestra en la Figura N° 9.

FIGURA N° 9. Página Inicial del Servidor Web.



La página de inicio contiene un mensaje indicando que la instalación fue satisfactoria, además proporciona instrucciones sobre el directorio donde deben residir las Páginas Web, para que el cliente pueda visualizarlas cuando las solicite. La página también contiene información sobre el Sitio Web para administrar y controlar el Servidor e incluye un link hacia la página de presentación del mismo.

2. Inicio del Servidor

El proceso de inicio del Servidor Web se lleva a cabo desde una ventana de consola ejecutando el siguiente comando:

```
perl /usr/bin/swp/sep.pl start
```

Con esta orden se invoca al programa principal *sep.pl*, el cual recibe un argumento *start* que le indica que el servidor debe iniciarse. Una vez iniciado el Servidor Web, se despliega en la ventana de consola, un mensaje indicando que el servidor esta

iniciado. Dicha ventana se puede minimizar y debe permanecer abierta, ya que si se cierra los procesos del Servidor Web finalizan. En la Figura N° 10 se muestra la ventana de consola.

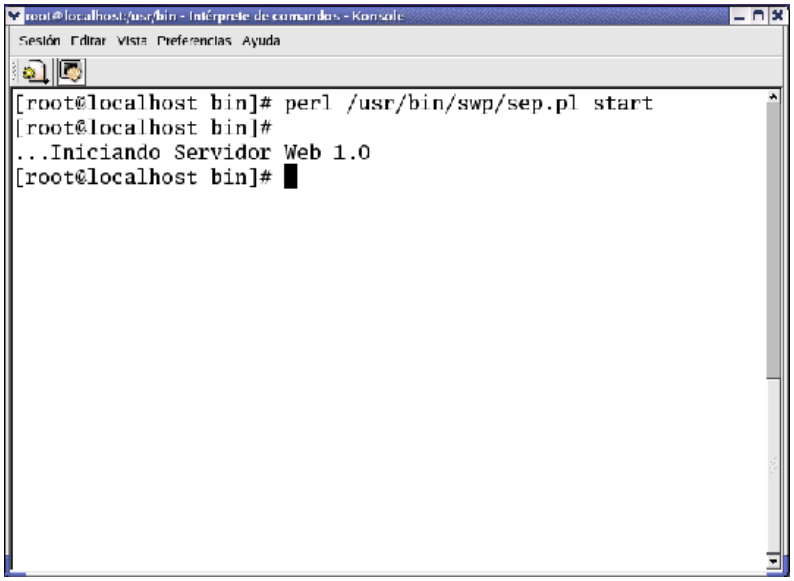


Figura N° 10. Pantalla de Inicio del Servidor Web 1.0.

3. Finalización del Servidor

Al ejecutar este comando en una ventana de consola, se Finaliza el Servidor Web, terminando todos los procesos del Servidor Web y deteniendo la ejecución del programa principal. El comando es el siguiente:

```
[root@localhost bin]# perl /usr/bin/swp/sep.pl stop
```

En la Figura N° 11 se visualiza la ventana de consola. Una vez eliminado el proceso que ejecuta el Servidor Web, aparece un mensaje indicando que el proceso fue eliminado.

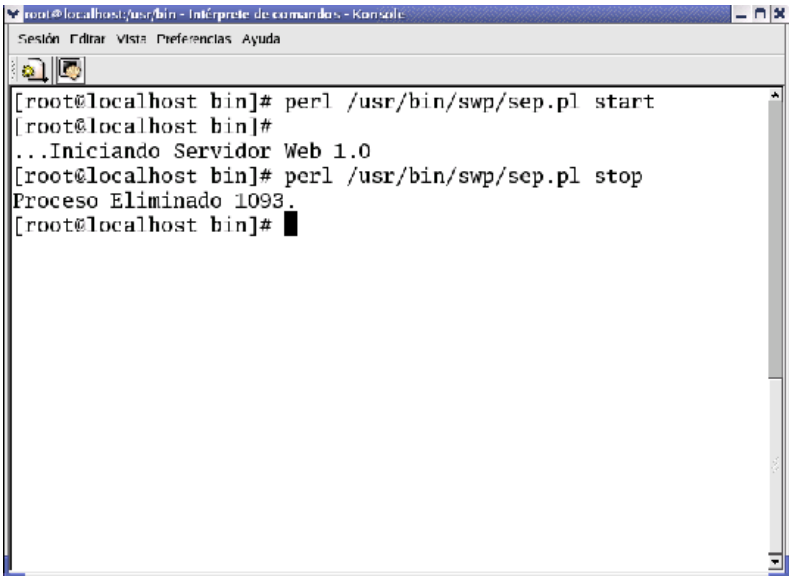


Figura N° 11. Pantalla de Finalización del Servidor Web 1.0.

4. Reinicio del Servidor

El proceso de inicio del Servidor Web se lleva a cabo desde una ventana de consola ejecutando el siguiente comando:

perl /usr/bin/swp/sep.pl start

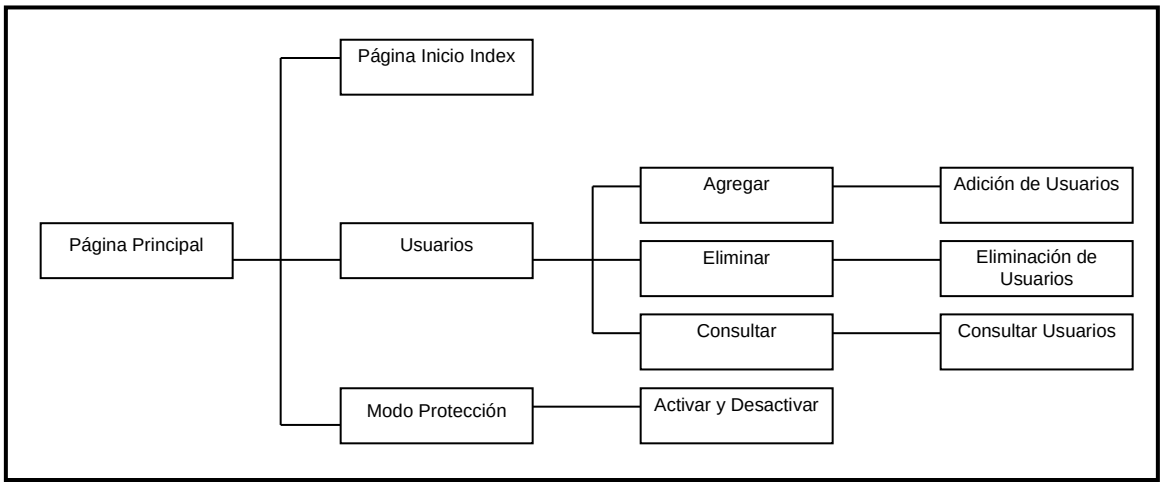
Una vez ejecutada la orden, se invoca a la función que termina los procesos actuales y automáticamente se ejecuta el proceso de inicio del servidor.

3.4.3 DESCRIPCION DE LA INTERFAZ EN MODO GRAFICO

1. Descripción del Sitio Web para la Administración del Software

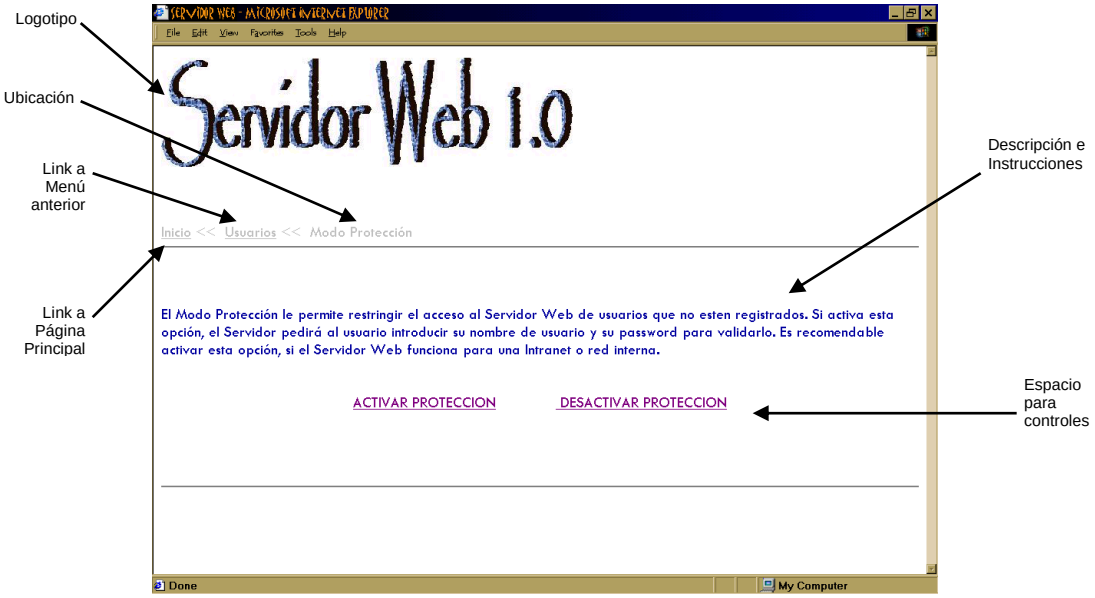
Las opciones de administración de usuarios y control del modo de protección, planteadas en los requerimientos de la interfaz del Servidor Web, están incluidas y se ejecutan desde un Sitio. Dicho sitio consta de una página principal que contiene todas las opciones del Sitio y desde donde se cargan las páginas de ejecución de las operaciones. En la Figura N° 12 se presente el árbol de opciones del Sitio Web.

FIGURA N° 12. Árbol de Opciones del Sitio Web Administrador.



En cada página se describe la función que se realiza y contiene los controles necesarios para realizar la operación. Todas las páginas incluyen el logotipo del Servidor Web y poseen un link hacia la página de presentación. En caso de ser un sub menú, un link hacia la opción principal de la que se derivan. De igual manera se indica la ubicación dentro del sitio. En la Figura N° 13 se describe el formato general de las páginas del Sitio Web.

FIGURA N° 13. Formato General de las Páginas Web.



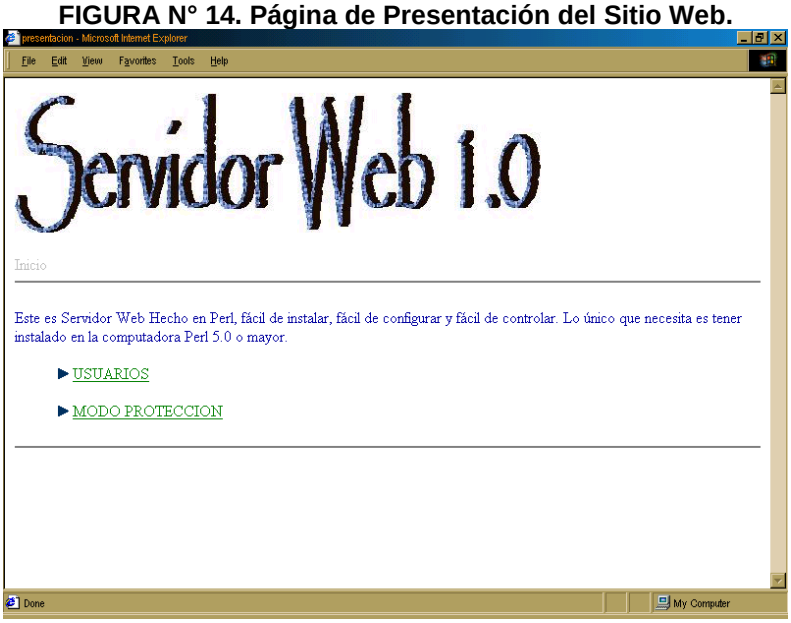
El Sitio Web esta formado por archivos de extensión html, y no necesita más que un navegador para visualizarse. Para ingresar, el usuario debe ejecutar la página de presentación, desde la cual tendrá acceso a todas las opciones disponibles. El sitio se ubica en el directorio Interfaz dentro del directorio raíz de documentos del Servidor Web, *html*, ya que dicho sitio ejecuta sus procesos a través del servidor.

2. Componentes Del Sitio Web

1. Presentación

Esta es la página principal del Sitio Web que incorpora una breve descripción del software y los requerimientos del mismo. La página contiene las opciones disponibles para la administración y operación del Servidor Web. En la Figura N°14 se muestra la página de presentación del Sitio Web administrador.

FIGURA N° 14. Página de Presentación del Sitio Web.



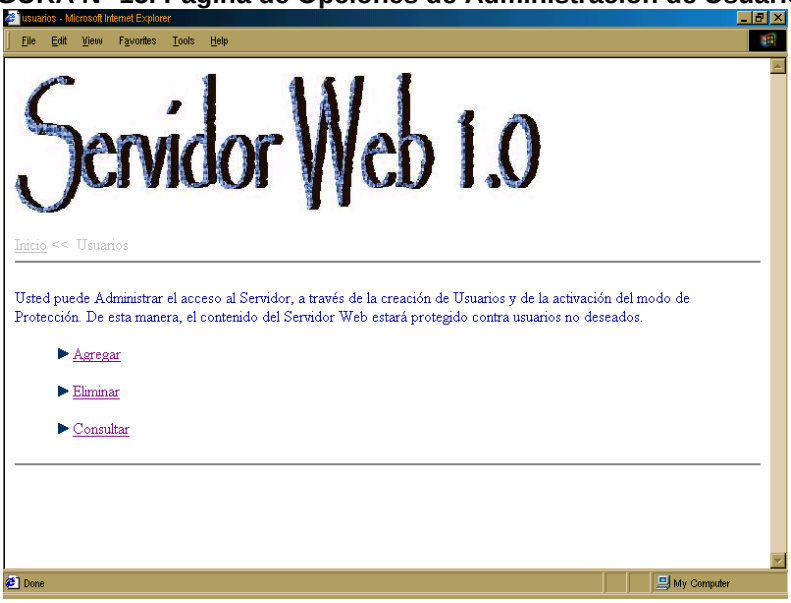
Las opciones que la página de presentación contiene son la siguientes:

- Usuarios: Administra los Usuarios del Servidor Web y activa el Modo de Protección.
- Modo Protección: Activa y desactiva el Modo de Protección del Servidor Web.

2. Usuarios

Esta opción se incluye para gestionar la administración y el control de los usuarios del Servidor Web. A través de ella se pueden adicionar, eliminar y consultar los usuarios registrados para accesar al contenido del Servidor Web, cuando el modo de protección esta activado. En la Figura N° 15 se observa la página de Usuarios.

FIGURA N° 15. Página de Opciones de Administración de Usuarios



Esta página incluye un sub menú con las opciones disponibles para administrar los usuarios. A continuación se describe cada una de ellas.

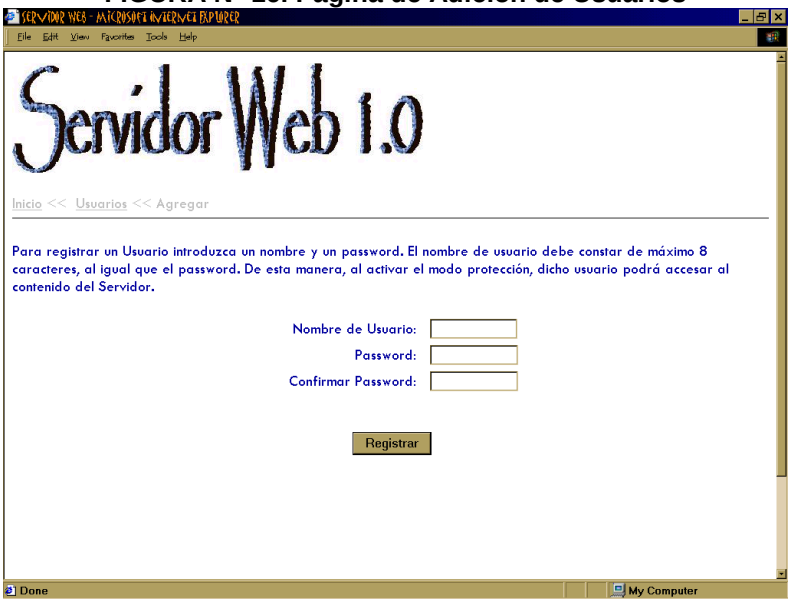
2.1 Adición

Con esta opción se pueden registrar usuarios del Servidor Web. La información que se solicita es la siguiente:

- Nombre de Usuario: Cadena de 8 caracteres máximo que identifica al usuario.
- Password: Identificador de 8 caracteres que validará al usuario cuando accede al servidor.
- Confirmar Password: Se utiliza para asegurarse de que la cadena que se introdujo en password sea la correcta.

En la Figura N° 16 se observa la página de adición de usuarios. Al presionar el botón Registrar, la información introducida en los campos se envía al programa que se encarga de realizar este proceso.

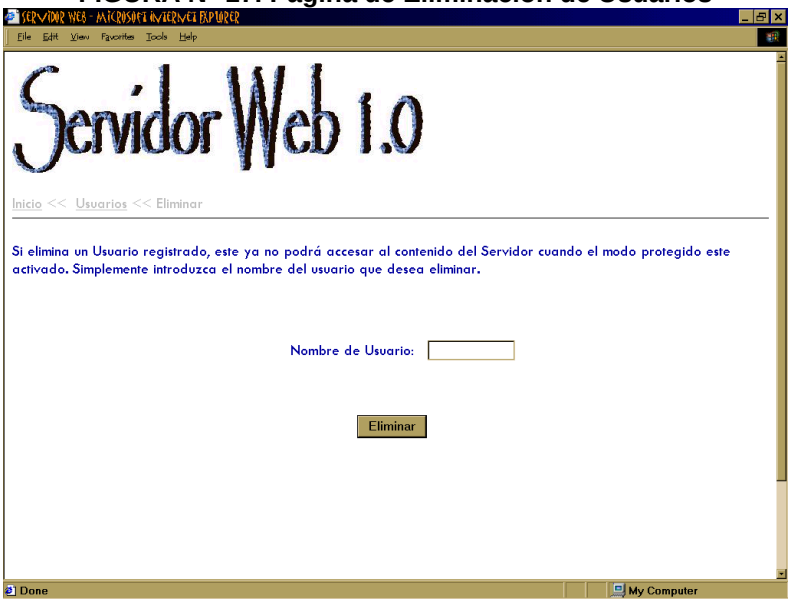
FIGURA N° 16. Página de Adición de Usuarios



2.2 Eliminación

La eliminación de usuarios registrados se realiza simplemente introduciendo en la caja de texto, el nombre del usuario que se desea eliminar. Al presionar el botón Eliminar, se invoca el programa que ejecuta este proceso. En la Figura N° 17 se muestra la página de eliminación.

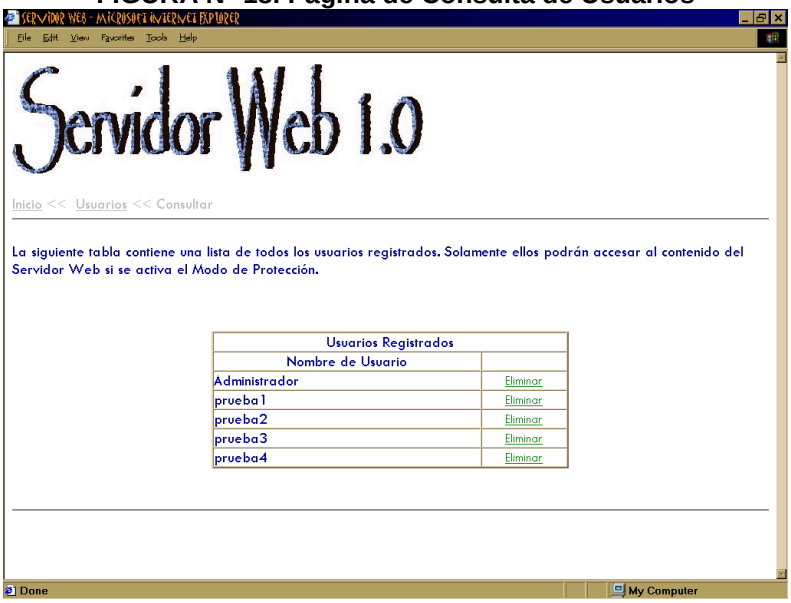
FIGURA N° 17. Página de Eliminación de Usuarios



2.3 Consultar

A través de esta opción se pueden visualizar en una tabla, todos los usuarios registrados que pueden accesar al contenido del Servidor Web cuando el modo de protección esta activado. En la Figura N° 18 se observa la página de consulta.

FIGURA N° 18. Página de Consulta de Usuarios

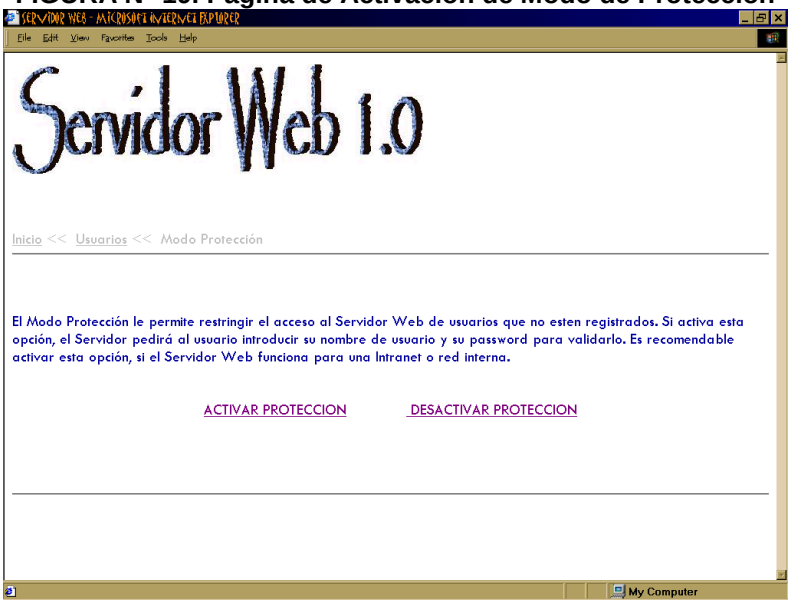


Como muestra la figura, la tabla posee dos columnas; una donde se muestra el nombre del usuario y la otra que incluye un link hacia la página de eliminación de usuarios.

3. Modo Protección

En esta página se define la protección contra usuarios no deseados o prohibidos, es decir, que se restringe el acceso al contenido del Servidor Web únicamente para los usuarios que estén registrados en el archivo correspondiente. La información contenida en dicha página se muestra en la Figura N° 19.

FIGURA N° 19. Página de Activación de Modo de Protección



Cuando se elige el link ACTIVAR PROTECCION, se ejecuta el programa que activa el Modo de Protección del Servidor Web. De esta manera sólo los usuarios que posean un nombre de usuario y un password registrado podrán accesar al contenido del mismo. Por otra parte, cuando se escoge el link DESACTIVAR PROTECCION, el Modo Protección es desactivado y cualquier usuario puede accesar al contenido del Servidor Web.

3.5 DISEÑO DEL MECANISMO DE AUTORECUPERACION EN CASO DE FALLAS DEBIDAS A FACTORES EXTERNOS

3.5.1 MECANISMO ADOPTADO

Un porcentaje importante de incidentes que acarrean perdida de Datos o reinstalar el Sistema Operativo, esta comprendido por los cortes de energía eléctrica o fallos en los equipos que están conectados 7 días, 24 horas, en el preciso momento en que varios usuarios tienen sus bases de datos abiertas o ejecutan procesos en el Servidor. Estos sucesos son imposibles de predecir por lo cual se convierte en una necesidad, el contar con un mecanismo que permita aumentar la tolerancia a los fallos. Entre los mecanismos más comúnmente implementados se encuentran los que disminuyen los riesgos de las fallas, en caso de que un dispositivo de almacenamiento masivo, como un disco duro y los que aumentan la protección del equipo en general en caso de bajón o corte de energía eléctrica, como los UPS automatizados o inteligentes.

Actualmente, Linux es conocido como un sistema operativo estable; la problemática se genera cuando el hardware, no es tan fiable como se desearía. En la mayoría de los casos, cuando un sistema falla normalmente es debido a un fallo de hardware o a un fallo humano. En los casos en que un fallo de hardware provoca graves consecuencias, debido a la naturaleza del servicio, se implementan sistemas tolerantes a fallos (fault tolerant ó FT); en los cuales, el servicio esta siempre activo. El problema de estos sistemas, es que son extremadamente caros y normalmente no hay presupuesto. Además suelen ser soluciones cerradas, totalmente dependientes de la empresa contratada. Se suele poner un servidor tolerante a fallos, varias interfaces de red, con tomas de alimentación redundantes y climatización especial.

Existen en Linux, herramientas que ayudan a implementar sistemas tolerantes a fallos, tanto en hardware como en software, en el caso del diseño para el Servidor

Web desarrollado, este será a nivel de software, es decir el control de las fallas del sistema se realizara a través de las configuraciones que se realicen a nivel de sistema operativo, utilizando las herramientas que este posee. Sin embargo, esto siempre requiere una inversión en hardware, que es mucho menor, que si el control de las fallas se hiciera a nivel, únicamente de hardware, ya que se tendría que adquirir equipo altamente especializado y mucho más costoso.

Las herramientas a utilizar consisten en un arreglo redundante de discos independientes o RAID, para el control de fallas a nivel de dispositivos de almacenamiento masivo y disponibilidad 7 días, 24 horas del equipo; para el control de fallas a nivel de factores externos, específicamente, los bajones o cortes de energía eléctrica, se propone las herramientas de Linux para configurar el monitoreo de UPS o SAI, esto permitirá al equipo detectar una falla de energía, cerrar el sistema correctamente y restaurarlo cuando se restablezca la energía. Ambas herramientas en conjunto proporcionaran un equipo tolerante a fallos, que podrá recuperarse en caso de una falla en el suministro de energía eléctrica, así como en el caso en que un disco duro falle. En los siguientes apartados se describen ambas herramientas y se presenta el diseño de la configuración en el equipo para cada una.

3.5.2 TOLERANCIA DE FALLAS

La tolerancia de fallas es la capacidad de un subsistema o un equipo de experimentar una falla sin comprometer el procesamiento y la integridad de los datos. El controlador RAID proporciona este soporte a través de matrices redundantes en los niveles de RAID 1, 5, 10 y 50. El sistema o equipo todavía puede funcionar adecuadamente incluso habiéndose producido una falla en el disco de una matriz, aunque en cierto modo, pueda disminuir el rendimiento.

A menudo, la tolerancia de fallas está asociada a la disponibilidad del equipo porque eso permite que esté disponible durante las fallas. Sin embargo, esto significa que también es importante que también esté disponible durante la solución del problema. La matriz del disco RAID continua manejando las peticiones mientras se produce la reconstrucción.

3.5.3 RAID

1. Concepto de RAID

RAID son las siglas en inglés de Redundant Array of Independent Disks, en español, Arreglo Redundante de Discos Independientes. RAID es una matriz de múltiples discos duros independientes que proporcionan un alto rendimiento y una tolerancia de fallas. Un subsistema de discos RAID mejora el rendimiento de entrada/salida (E/S) respecto a un equipo que utilice una sola unidad. El ordenador host trata la matriz RAID como si se tratara de una sola unidad de almacenamiento o bien de varias unidades lógicas. Al permitir el acceso a varios discos simultáneamente, aumenta la velocidad de las operaciones de E/S. Los sistemas RAID mejoran la fiabilidad del almacenamiento de datos y la tolerancia de fallas. Los datos perdidos a causa de un fallo en la unidad de disco pueden recuperarse mediante la reconstrucción de los datos que faltan a partir de las unidades de paridad y datos restantes (para todos los niveles de RAID excepto RAID 0).

Los sistemas RAID también mejoran la fiabilidad del almacenamiento de datos y la tolerancia de fallas. Los datos perdidos a causa de un fallo en la unidad de disco pueden recuperarse mediante la reconstrucción de los datos que faltan a partir de las unidades de paridad y datos restantes. RAID se basa en la combinación de múltiples unidades de disco pequeñas y baratas que se agrupan en un conjunto de discos para llevar a cabo acciones que no se pueden realizar con unidades grandes y costosas. La computadora las considerará como si fueran una única

unidad de disco lógica. RAID es el método que se usa para expandir información en diversos discos utilizando técnicas como el *vaciado del disco* (RAID Nivel 0), la *creación de réplicas del disco* (RAID nivel 1) y el *vaciado del disco con paridad* (RAID Nivel 5) para obtener redundancia, menos latencia y/o aumentar el ancho de banda para leer o escribir en discos y maximizar así la posibilidad de recuperar información cuando el disco duro no funciona.

RAID está basado en el concepto de que los datos tienen que distribuirse en cada conjunto de discos de manera consistente. Para ello, los datos se rompen en *chunks* o grupos de datos con un tamaño que varía normalmente entre 32K y 64K aunque se pueden usar otros tamaños. Cada grupo de datos se escribe en el disco duro según el nivel de RAID. Cuando se leen los datos, se invierte el proceso de manera que parece que existan muchas unidades de disco en una sola. Cualquier entidad o persona que necesite tener a mano grandes cantidades de datos, obtendrá grandes beneficios de la tecnología RAID. Entre otros beneficios, se incluyen los siguientes:

- Mayor velocidad
- Mayor capacidad de almacenamiento usando un solo disco virtual.
- Disminución del impacto del fallo de un disco.

2. Hardware y Software RAID

Existen dos posibilidades de usar RAID: hardware RAID o software RAID.

Hardware RAID

El sistema basado en el hardware gestiona el subsistema independientemente de la máquina y presenta a la máquina un único disco por conjunto de discos RAID. Un ejemplo del hardware RAID sería el que se conecta a un controlador SCSI y presenta el conjunto de discos RAID en una sola unidad de disco. Un sistema externo RAID se encarga de mover la inteligencia RAID a un controlador que se

encuentra en un subsistema de discos externo. Todo el subsistema está conectado a la máquina con un controlador SCSI normal y para la máquina es como si se tratara de una sola unidad de disco.

Los controladores RAID también tienen la forma de tarjetas que actúan como un controlador SCSI del sistema operativo pero se encargan de todas las comunicaciones del disco actual. En estos casos, tiene que conectar las unidades de disco al controlador RAID como si se tratara de un controlador SCSI pero tiene que añadirlas a la configuración del controlador RAID; de todas maneras el sistema operativo nunca nota la diferencia.

Software RAID

El software RAID implementa los diversos niveles de RAID en el código del kernel (dispositivo de bloque). Ofrece la solución más barata ya que las tarjetas de controladores de disco o los chassis hot-swap[21] son bastante caros. El software RAID usa discos IDE más baratos así como Discos SCSI. Con las computadoras rápidas actuales, el rendimiento del software RAID aumenta con respecto al del hardware RAID.

El controlador MD en el kernel de Linux es un ejemplo de la solución RAID que es completamente independiente del hardware. El rendimiento del conjunto de discos del software RAID depende del rendimiento y de la carga del equipo o computadora servidor. Entre las principales funciones del software RAID, se encuentran:

[21] Un chasis de hot-swap permite quitar un disco duro sin tener que apagar la computadora.

- Proceso de reconstrucción de subprocesos.
- Configuración basada en el kernel.
- Portabilidad de los conjuntos de discos entre máquinas Linux sin reconstrucción.
- Reconstrucción de los conjuntos de discos con el uso de los recursos que no se usan del sistema.
- Soporte para las unidades de disco en las que se pueden hacer cambios en caliente (hot-swappable), es decir sin apagar el equipo.
- Detección automática de CPU con el objetivo de obtener beneficios de las mejoras de CPU.

3. Niveles RAID y soporte lineal

RAID soporta varias configuraciones, entre las que se incluyen los niveles 0, 1, 4, 5 y lineal. Estos tipos RAID se definen a continuación.

- *Nivel 0*. Nivel RAID 0, también llamado striping o de distribución por bandas, es una técnica de vaciado de datos. Esto significa que los datos que se escriben en la unidad de disco se rompen en grupos y se escriben en los discos que forman parte del conjunto, lo que permite un rendimiento alto de E/S a un costo inherente pero no proporciona redundancia. La capacidad de almacenamiento del nivel 0 es igual a la capacidad de los discos pertenecientes al hardware RAID o igual a la capacidad total de las particiones miembro del software RAID. En este nivel no hay redundancia, por lo que si un disco falla, se perderá toda la información.

- *Nivel 1.* Nivel RAID 1, o réplicas ha sido la técnica más usada de RAID. El nivel 1 proporciona redundancia al escribir datos idénticos en cada uno de los discos miembros dejando una copia en cada disco. Esta técnica es muy conocida debido a su simplicidad y al alto nivel de transferencia de datos cuando se leen

éstos pero normalmente actúan independientemente y dan altos niveles de transferencia de datos I/O. El nivel 1 ofrece una gran fiabilidad de los datos y mejora el rendimiento de las aplicaciones de lectura intensa solo que a un precio bastante alto. La capacidad de almacenamiento del nivel 1 es igual a la capacidad de las réplicas de los discos duros en el hardware RAID o en una de las réplicas de las particiones del software RAID.

- *Nivel 4.* El nivel 4 usa paridad concentrada en una sola unidad de disco para proteger los datos. Es mejor la transferencia de E/S que la de un fichero grande. Debido a que el disco con la paridad representa un cuello de botella inherente, el nivel 4 se usa raramente sin tecnologías como el caché de retroceso en la escritura o write-back caching. Aunque el nivel 4 es la opción en algunos esquemas de particionamiento RAID, no se permite en las instalaciones RAID del sistema operativo Red Hat Linux. La capacidad de almacenamiento del nivel 4 del hardware RAID es igual a la capacidad de los disco miembro menos la capacidad de cada disco. La capacidad de almacenamiento del software RAID en el nivel 4 es igual a la capacidad de las particiones miembro menos las dimensiones de una de las particiones si tienen el mismo tamaño.

- *Nivel 5.* Este es el tipo de RAID más común. Al distribuir la paridad entre los discos miembro, el nivel 5 elimina el cuello de botella de la escritura del nivel 4. El único cuello de botella sería el proceso para calcular la paridad, con los software RAID y las computadoras modernas no hay problemas. Como con el nivel 4, el resultado es un rendimiento asimétrico haciendo que el de la lectura sea menor del de la escritura. El nivel 5 normalmente se usa para el caché de la escritura en retroceso para reducir la asimetría. La capacidad de almacenamiento del nivel 5 del hardware RAID es igual a la capacidad de los discos miembro menos la capacidad de cada disco miembro. La capacidad del

nivel 5 del software RAID es igual a la capacidad de las particiones miembro menos el tamaño de cada una de las particiones si tienen el mismo tamaño. Este es quizás el modo RAID más útil cuando uno desea combinar un mayor número de discos físicos y todavía conservar alguna redundancia. Si uno de los discos falla, todos los datos permanecerán intactos, gracias a la información de paridad. Si existen discos de reserva disponibles, la reconstrucción comenzará inmediatamente después del fallo del dispositivo. Si dos discos fallan simultáneamente, todos los datos se perderán. RAID-5 puede sobrevivir a un fallo de disco, pero no a dos o más.

- *Lineal RAID.* El nivel lineal de RAID consiste en un simple reagrupamiento de las unidades de disco para crear una unidad de disco virtual más grande. Los grupos de datos o chunks están situados en los discos miembro siguiendo una secuencia de manera que pasan al siguiente cuando el anterior se ha llenado. Esto no da ningún rendimiento ya que las operaciones de E/S no se rompen entre cada uno de los discos miembro. El nivel lineal de RAID no da redundancia y de hecho reduce la fiabilidad, si uno de los discos falla, no se puede usar el conjunto de discos y se perderá toda la información. La capacidad es la capacidad total de todos los discos miembro.

4. Discos de Reserva

Los discos de reserva son discos que no forman parte del grupo RAID hasta que uno de los discos activos falla. Cuando se detecta un fallo de disco, el dispositivo se marca como defectuoso y la reconstrucción se inicia inmediatamente sobre el primer disco de reserva disponible. De esta manera, los discos de reserva proporcionan una buena seguridad extra, especialmente a sistemas RAID-5 que tal vez, sean difíciles de lograr (físicamente). Se puede permitir que el sistema funcione durante algún tiempo con un dispositivo defectuoso, ya que se conserva toda la redundancia mediante los discos de reserva.

5. Requisitos Para configurar RAID software

Para cualquiera de los niveles RAID se necesita lo siguiente:

- Un núcleo. Instalar un kernel que permita usar las herramientas RAID. Cualquier kernel 2.4.x es válido.
- Los parches RAID. Normalmente existe un parche disponible para los núcleos recientes.
- El paquete de herramientas RAID (raidtools).

3.5.4. CONFIGURACIÓN DE RAID SOFTWARE

Únicamente se describirán las configuraciones RAID-1,4,5 ya que un array de estos niveles son capaces de sobrevivir a un fallo de disco, que es el aspecto de interés para el diseño que se esta realizando. Un modo lineal o un RAID-0 fallarán totalmente cuando se pierda un dispositivo.

1. Configuración Modo Gráfico

En Linux la composición de RAID se hace a nivel de partición, por lo que se puede montar por ejemplo un RAID 5 con 3 particiones en 3 discos diferentes (realmente pueden ser el mismo disco, pero no tendría sentido y se perderían las capacidades de redundancia y rendimiento de los dispositivos RAID).

RAID puede configurarse durante la instalación gráfica de Red Hat Linux o durante el inicio rápido del sistema. Se puede utilizar el programa fdisk o Disk Druid para crear una propia configuración RAID, pero estas instrucciones se centrarán principalmente en cómo utilizar Disk Druid para llevar a cabo esta tarea. Antes de poder crear un dispositivo RAID, lo primero es crear las particiones RAID, usando las siguientes instrucciones paso a paso.

- Crear una partición. En Disk Druid, se elige el botón Add para crear una nueva partición. (Ver Figura N° 20).

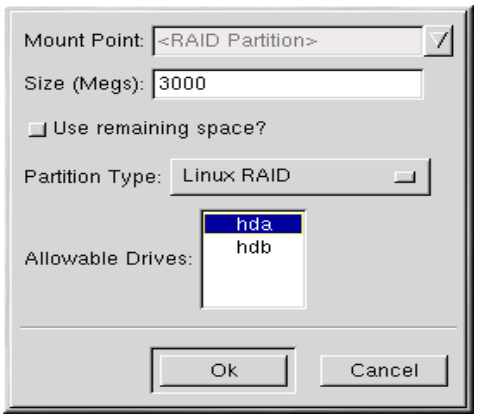


Figura N° 20**. Crear una nueva partición RAID.**

- Ahora se podrá introducir un punto de montaje (esto debe poder hacerse una vez se haya creado el dispositivo RAID).
- Introducir el tamaño que se desea para la partición.
- Seleccionar la opción *Use remaining space,* si se quiere que la partición aumente de tamaño automáticamente para ocupar todo el espacio libre disponible en el disco. En este caso, el tamaño de la partición aumentará o disminuirá en función de los tamaños que tomen el resto de las particiones del disco. Si se quiere tener más de una partición que pueda crecer, las particiones repartirán el espacio libre en proporciones iguales.
- Elegir la opción *Linux RAID* del menú *Partition Type*.
- Finalmente, en *Allowable Drives*, seleccionar el disco donde se quiere crear el RAID. Si se tiene varios discos, todos los discos podrán ser seleccionados desde aquí y deberá deseleccionar los discos que no tengan un array RAID.

Seguir estos pasos para crear tantas particiones como se necesitan para la configuración RAID. Es necesario tener en cuenta que no todas las particiones tienen porqué ser RAID. Por ejemplo, en la Figura N° 21, tan sólo la partición /home es un dispositivo RAID por software.

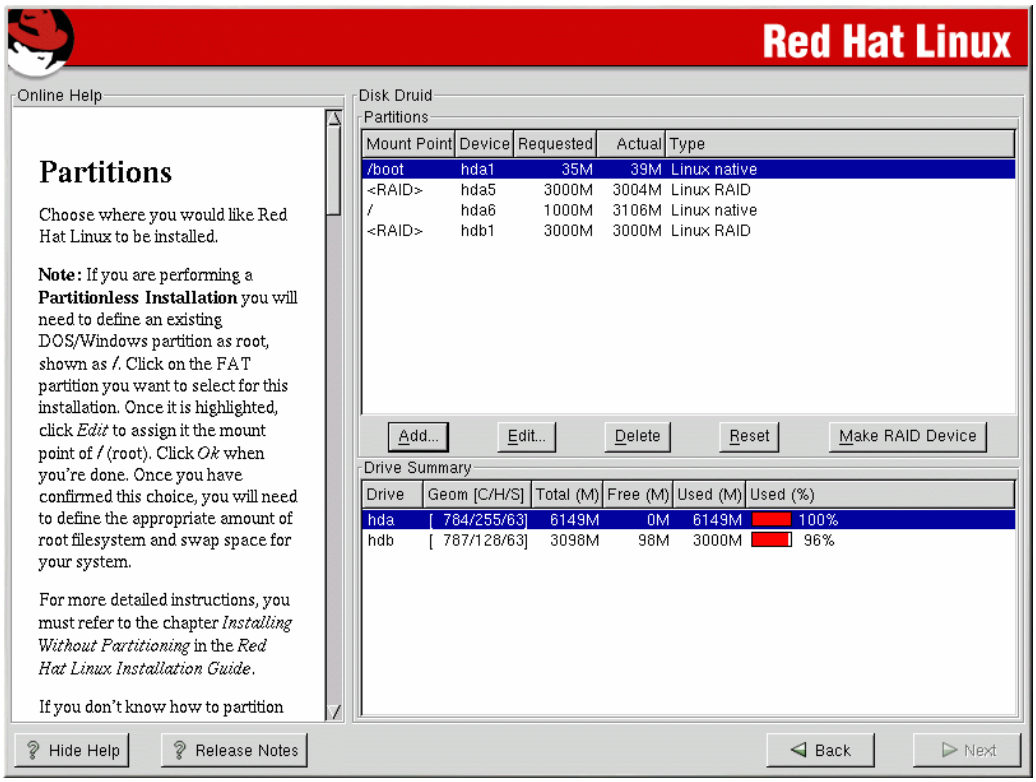


Figura N° 21**. Particiones RAID.**

Una vez que se hayan creado todas las particiones que se necesitan como RAID, seleccionar el botón *Make RAID Device* en la pantalla principal de Disk Druid (Figura N° 21). A continuación, aparecerá la Figura N° 22, donde se podrá crear un dispositivo RAID.

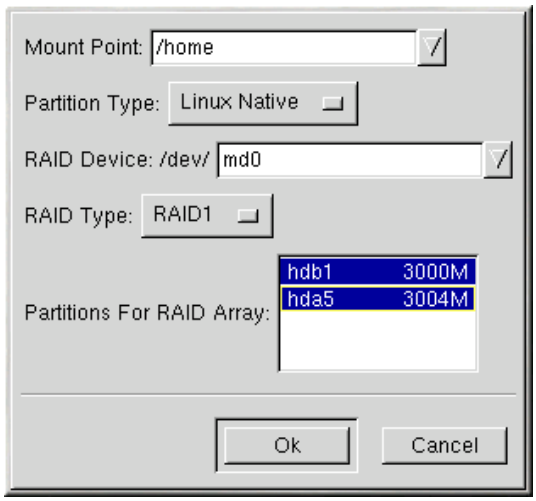


Figura N° 22**. Crear un dispositivo RAID.**

- Primero, introducir un punto de montaje.
- A continuación, elegir el tipo de partición para la partición.
- Elegir el dispositivo RAID. Se deberá elegir md0 para el primer dispositivo, md1 para el segundo dispositivo, y así para el resto, a no ser que se tengan razones muy concretas para hacerlo de otra forma. Los dispositivos RAID de md0 a m7 y sus combinaciones tan sólo deberían usarse una vez.
- Elegir el tipo de RAID entre RAID 0, RAID 1, y RAID 5.
- Para finalizar, seleccionar qué particiones irán en el array RAID (como en Figura N° 23) y hacer click en el botón Next.

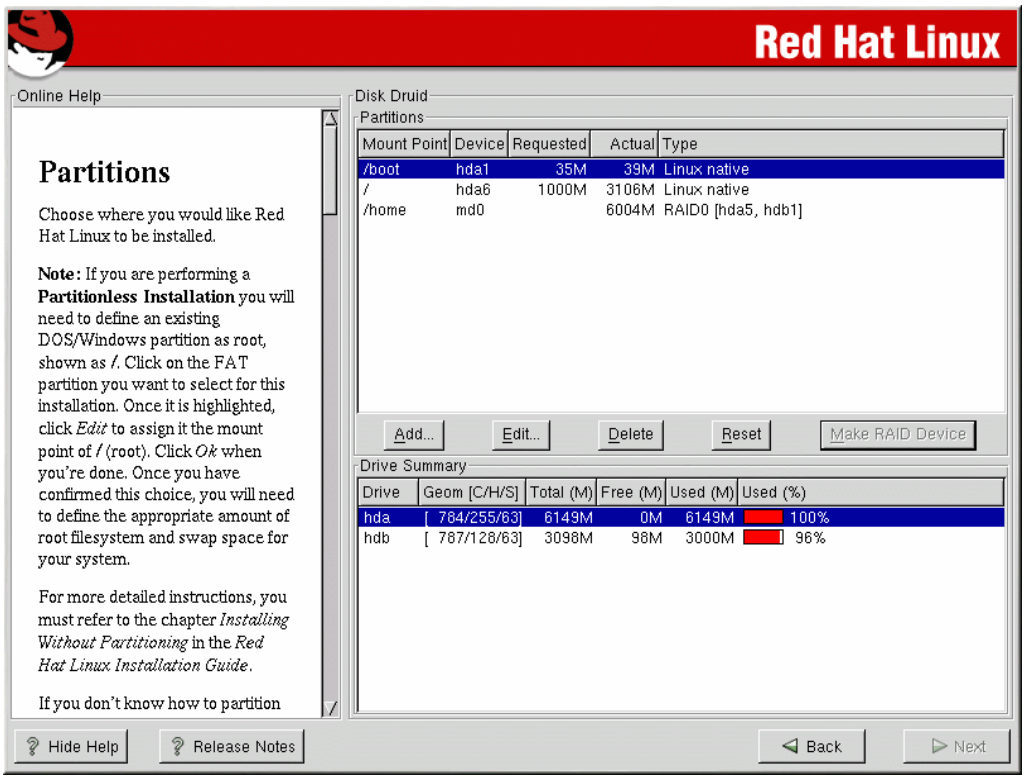


Figura N° 23**. Crear un Array RAID.**

En este momento, se podrá continuar con el proceso de instalación.

2. Configuración Modo Texto

En primer lugar se debe aplicar el parche al núcleo, configurarlo para incluir el soporte del nivel RAID que se quiera usar. Compilarlo e instalarlo. A continuación desempaquetar, configurar, compilar e instalar las herramientas RAID. Al reiniciar debe existir el fichero */proc/mdstat*. Este debe indicar que se tiene registrada la personalidad RAID (es decir, el modo RAID) correcta y que actualmente no hay dispositivos RAID activos. Se deben crear las particiones que se quieren incluir en el grupo RAID.

RAID-1

Se tienen dos dispositivos de aproximadamente el mismo tamaño y se quiere que cada uno de los dos sea un duplicado del otro. Finalmente, se tienen más dispositivos que se quieren guardar como discos de reserva preparados, que automáticamente formarán parte del duplicado si uno de los dispositivos activos se rompe.

Configurar el fichero /etc/raidtab:

```
raiddev /dev/md0
        raid-level          1
        nr-raid-disks               2
        nr-spare-disks  0
        chunk-size      4
        persistent-superblock 1
        device          /dev/sdb6
        raid-disk       0
        device          /dev/sdc5
        raid-disk       1
```

Si se tienen discos de reserva, se pueden añadir al final de la especificación de dispositivos de la siguiente forma:

```
        device          /dev/sdd5
        spare-disk      0
```

En este momento se puede comenzar la inicialización del RAID. Se debe construir el duplicado, es decir, los contenidos de los dos dispositivos se deben sincronizar. Para realizar esto de debe ejecutar la siguiente orden, para comenzar la inicialización del duplicado.

```
mkraid /dev/md0
```

Comprobar el fichero */proc/mdstat*. Debe indicar que se ha puesto en funcionamiento el dispositivo /dev/md0, que está siendo reconstruido el duplicado y una cuenta del tiempo estimado para la terminación de la reconstrucción. La

reconstrucción se realiza usando el ancho de banda libre de E/S. De esta manera, el sistema debe ser capaz todavía de responder en gran medida. El proceso de reconstrucción es transparente, por lo que realmente se puede usar el dispositivo aunque la duplicación esté actualmente en curso.

RAID-4

Se Tienen tres o más dispositivos de aproximadamente el mismo tamaño, un dispositivo es significativamente más rápido que los otros dispositivos y se quieren combinar todos en un único dispositivo más grande, conservando todavía alguna información de redundancia. Finalmente, se tienen varios dispositivos que se desean usar como discos de reserva.

Configurar el fichero */etc/raidtab* de la siguiente forma:

```
raiddev /dev/md0
        raid-level          4
        nr-raid-disks       4
        nr-spare-disks      0
        persistent-superblock 1
        chunk-size          32
        device              /dev/sdb1
        raid-disk           0
        device              /dev/sdc1
        raid-disk           1
        device              /dev/sdd1
        raid-disk           2
        device              /dev/sde1
        raid-disk           3
```

Si se tuviera discos de reserva, se insertarían de forma similar, siguiendo las especificaciones de discos RAID:

```
        device          /dev/sdf1
        spare-disk      0
```

El array se puede inicializar con la orden siguiente:

```
mkraid /dev/md0
```

RAID-5

Se tienen tres o más dispositivos de aproximadamente el mismo tamaño, se quieren combinar en un dispositivo mayor, pero conservando todavía cierto grado de redundancia para la seguridad de datos. Finalmente, se tienen varios dispositivos para usar como discos de reserva, que no tomarán parte en el array antes de que otro dispositivo falle. Si se utilizan N dispositivos donde el tamaño del más pequeño es S, el tamaño de todo el array será (N-1)*S. El espacio que falta se usa para información de paridad (redundancia). De esta manera, si cualquier disco falla, todos los datos permanecerán intactos. Pero si dos discos fallan, todos los datos se perderán.

Configurar el fichero /etc/raidtab de la siguiente manera:

```
raiddev /dev/md0
        raid-level          5
        nr-raid-disks       7
        nr-spare-disks      0
        persistent-superblock 1
        parity-algorithm        left-symmetric
        chunk-size          32
        device              /dev/sda3
        raid-disk           0
        device              /dev/sdb1
        raid-disk           1
        device              /dev/sdc1
        raid-disk           2
      device              /dev/sdd1
      raid-disk           3
      device              /dev/sde1
      raid-disk           4
      device              /dev/sdf1
      raid-disk           5
      device              /dev/sdg1
      raid-disk           6
```

Si se tuviera discos de reserva, se insertarían de forma parecida, siguiendo las especificaciones de discos RAID (para cada uno de los dispositivos):

```
device        /dev/sdh1
spare-disk    0
```

Un tamaño de segmento de 32KB es un buen valor por defecto para muchos sistemas de ficheros de propósito general de estas proporciones. Si el array sobre el que se utiliza el raidtab anterior es un dispositivo de 7 por 6 GB = 36 GB (recuerde que (N-1)*S = (7-1)*6 = 36). Contiene un sistema de ficheros ext2 con un tamaño de bloque de 4KB. Se podría incrementar tanto el tamaño del segmento unitario del array como el tamaño de bloque del sistema de ficheros si el sistema de ficheros fuera o bien mucho mayor o bien si simplemente contuviera ficheros muy grandes.

Configurar el fichero */etc/raidtab* y ejecutar la orden, para comprobar el funcionamiento:

```
mkraid /dev/md0
```

En este momento los discos deben comenzar a trabajar ya que empiezan la reconstrucción del array. Si se observa el fichero */proc/mdstat* se podrá observar lo que esta sucediendo. Si el dispositivo se ha creado correctamente, el proceso de reconstrucción comenzará en ese momento. El array no será consistente hasta que esta fase de reconstrucción haya terminado. No obstante, el array es totalmente funcional.

Una vez se tenga el dispositivo RAID funcionando, se puede parar o rearrancarlo usando las siguientes ordenes, respectivamente:

```
raidstop /dev/md0
raidstart /dev/md0,
```

3. Comprobación del funcionamiento del esquema RAID

Si se piensa usar un RAID para obtener tolerancia a fallos, también puede que se desee comprobar su configuración para ver si realmente funciona. A continuación se describen dos maneras de realizar dicha comprobación.

Simulación de un fallo de disco

Si se quiere simular un fallo de disco se debe desconectar la unidad, con el sistema apagado. Si de interés comprobar si los datos pueden sobrevivir con un disco menos de los habituales, se debe apagar el sistema, desconectar el disco y encenderlo de nuevo. Se puede observar en el registro del sistema (generado por syslogd) y en */proc/mdstat* para ver qué es lo que está haciendo el RAID.

Cuando se haya reconectado el disco de nuevo se podrá añadir el nuevo dispositivo al RAID otra vez, con la orden raidhotadd.

Simulación de corrupción de datos

Un RAID (ya sea hardware o software) asume que si una escritura en un disco no devuelve un error, entonces la escritura ha tenido éxito. Por tanto si un disco corrompe datos sin devolver un error, los datos se corromperán. Naturalmente, esto es muy improbable que ocurra, pero es posible, y produciría un sistema de ficheros corrupto. Un RAID no puede y no está pensado para proteger contra la corrupción de datos del medio de almacenamiento físico. Por tanto, tampoco tiene ningún sentido corromper a propósito los datos de un disco para ver cómo manejará el sistema RAID esa situación. Es más probable que la capa RAID no descubra nunca la corrupción, sino que el sistema de ficheros en el dispositivo RAID se corrompa. Un RAID no es una garantía absoluta para la integridad de datos, simplemente le permite conservar sus datos si un disco muere (con RAIDs de niveles iguales o superiores a 1).

3.5.5 SAI (UPS)

Un Sistema de Alimentación Ininterrumpida (SAI), en ingles Uninterruptible Power Supply (UPS), es un aspecto importante a considerar para los equipos que pueden o no, formar parte de un Red, en especial si se posee un Servidor Web, DNS o de Correos, que deben estar activos las 24 horas, 7 días.

Tener un UPS no implica, en muchos casos, estar completamente protegido contra los problemas de alimentación. En ambientes donde el equipo puede pasar largas horas sin vigilancia alguna, la existencia del UPS no es una garantía contra el mal cerrado o apagado de su sistema o equipo, debido a que, usualmente el respaldo no dura tanto. Lo ideal en estos casos es que la computadora detecte la falla de alimentación y antes que el UPS pierda toda la carga, cierre o apague su equipo correctamente y desconecte el mismo.

Un UPS es diseñado para asegurarse de que la PC consiga la energía siempre a un mismo nivel constante, eso se consigue con las baterías de la UPS. El UPS contiene una batería que constantemente está cargándose para estar a la capacidad máxima. Cuando hay un bajón de tensión o una sobrecarga de tensión, el UPS inmediatamente (en 1 o 2 milisegundos) reemplaza la energía que estaba usando por la de las baterías. Las batería de la UPS no duran para siempre, le alertan al usuario con una alarma que ocurrió algo, para que este pueda guardar los cambios en sus archivos, y apagar la computadora sin complicaciones. Un UPS puede conectarse a la computadora, vía puerto serial o bien USB. Para esto se debe instalar un software especial que habilita y monitoria al UPS, para que apague todo automáticamente en caso de su ausencia. Esto es muy conveniente si deja la PC, todo el tiempo encendida.

Un SAI puede ser una buena opción para un sistema Red Hat Linux ya que le proporcionará el tiempo necesario para apagar el sistema correctamente en el

caso de la interrupción del servicio eléctrico. El fichero */etc/sysconfig/ups* se utiliza para especificar información sobre cualquier Sistema de Alimentación Ininterrumpida (SAI o UPS) conectado a su sistema. Se pueden utilizar los siguientes valores para configurar dicho fichero:

- **SERVER=<*value*>**, donde **<*value*>** puede tomar los siguientes valores:
  - **yes** — Si se ha instalado un SAI en el sistema.
  - **no** — Si no se ha instalado ningún SAI en el sistema.
- **MODEL=<*value*>**, donde **<*value*>** debe estar seleccionado a uno de los siguientes valores o bien a **NONE** si no hay ningún SAI instalado en el sistema:
  - **apcsmart** — Para un dispositivo APC SmartUPS™ o similar.
  - **fentonups** — Para un dispositivo Fenton UPS™.
  - **optiups** — Para un dispositivo OPTI-UPS™.
  - **bestups** — Para un SAI Best Power™.
  - **genericups** — Para un SAI genérico.
  - **ups-trust425+625** — Para un SAI Trust™.
- **DEVICE=<*value*>**, donde **<*value*>** especifica dónde está conectado el SAI, como pueda ser **/dev/ttyS0**.
- **OPTIONS=<*value*>**, donde **<*value*>** es un comando especial que hay que pasarle al SAI.

1. Requisitos

- Una unidad SAI con puerto serie.
- Software para monitorizar el SAI. Se recomiendan las herramientas *NUT (Network UPS Tools)* o *Power Guard.*

Antes de instalar es importante verificar que el SAI tenga la batería cargada al máximo y que el equipo a conectar este tomando energía del mismo, además el cable serie debe estar correctamente conectado. Para efectos de ejemplificaciones, en este caso se tomara como software de monitoreo la herramienta NUT.

2. Funcionamiento General

A continuación se explica brevemente como funciona el control del SAI.

1. Todo está funcionando perfectamente.
2. Se da un corte de energía eléctrica, y el SAI entra en modo batería.
3. La batería llega a su carga mínima.
4. El sistema maestro notifica a los esclavos que dentro de poco se deben apagar, o si es una red, el equipo detecta que debe apagarse.
5. Cuando los esclavos o el equipo reciben la orden, si es una red:
    1. Generan un evento NOTIFY_SHUTDOWN.
    2. Esperan el tiempo definido en FINALDELAY.
    3. Ejecutan el comando definido en SHUTDOWNCMD.
    4. Se apagan correctamente.
6. En el caso de una red, el sistema maestro espera que todos los clientes se desconecten.
7. El maestro, o el equipo en caso que no sea una red, empieza la secuencia de apagado:
    1. Genera un evento NOTIFY_SHUTDOWN
    2. Espera el tiempo definido en FINALDELAY
    3. Crea el fichero definido en POWERDOWNFLAG
    4. Ejecuta el comando definido en SHUTDOWNCMD
8. El proceso de apagado se lleva a cabo normalmente, y el sistema va parando los servicios y desmontando unidades.
9. El sistema encuentra en fichero definido en POWERDOWNFLAG, y ejecuta el apagado del SAI.
10. Cuando se restaura la energía eléctrica, todos los sistemas se activan y todo vuelve a su estado normal.

El hecho de crear un fichero sirve para que los scripts de apagado del sistema comprueben si existe, y si es así, envíen al SAI el comando de apagado. Esto es porque no es conveniente tener el SAI encendido si ningún sistema está funcionando. El último script que se ejecuta durante el apagado es */etc/init.d/halt*. Este script llama a otro script */etc/init.d/ups-monitor* con el parámetro *poweroff*. Si existe el fichero definido en POWERDOWNFLAG, el SAI se apagará y quedarán todas las computadoras sin electricidad.

3.5.6. CONFIGURACION DEL SAI

En esta sección se describirá como configurar el sistema que controla al SAI por el puerto serie.

1. Especificación del SAI

En primer lugar se debe configurar el fichero */etc/nut/ups.conf*. Este fichero contiene los parámetros necesarios para poderse conectar al SAI.

```
[elsai]
        driver = mge-utalk
        port = /dev/ttyS0
```

Lo que va entre corchetes *([elsai])* es un nombre descriptivo, puede ser cualquier nombre (sin espacios ni signos de puntuación). La línea *driver* especifica qué tipo de comunicación se usará para *hablar* con el SAI. En este caso se ha utilizado el *mge-utalk*, que sirve para modelos de SAI MGE. La línea *port* especifica cual es el puerto serie por el cual se llevara la comunicación con el SAI.

2. Listas de control de acceso

Una de las características de NUT es que permite configurar niveles de acceso a las distintas máquinas de la red. Los controles de acceso se configuran en el fichero */etc/nut/upsd.conf*. Este es un ejemplo:

```
ACL localhost 127.0.0.1/32
ACL maquina1 172.16.2.3/32
ACL maquina2 172.16.2.6/32
ACL maquina3 172.16.2.1/32
ACL maquina4 172.16.2.18/32
ACL maquina5 172.16.2.14/32
ACL all 0.0.0.0/0

ACCESS grant monitor localhost
ACCESS grant monitor maquina1
ACCESS grant monitor maquina2
ACCESS grant login maquina3
ACCESS grant monitor maquina4
ACCESS grant monitor maquina5
ACCESS deny all all
```

En el primer bloque se definen una serie de nombres asociados a unas IPs. En el segundo bloque se define qué clase de acceso tienen los distintos nombre antes mencionados. La directiva grant significa *acceso permitido*, y la directiva *deny* significa acceso denegado. El orden de las ACL es importante, así que primero se debe permitir el acceso a las máquinas que se desea permitir, y luego se deniega a las restantes.

3. Creación de los usuarios de acceso

Para que las máquinas (tanto el servidor como los esclavos) tengan acceso al sistema de control, será necesario crear una serie de usuarios con sus correspondientes contraseñas. Estos usuarios se configuran en el fichero */etc/nut/upsd.users*:

```
[admin]
        password = mipassword
        allowfrom = localhost
        actions = SET
        instcmds = ALL

[control]
        password = otropassword
        allowfrom = localhost
        upsmon master

[clientes]
        password = yotropassword
        allowfrom = maquina1 maquina2 maquina4 maquina5
        upsmon slave
```

Lo que va entre corchetes es el nombre del usuario que se quiere crear. La línea *password* es la contraseña que se le asigna a ese usuario. La línea *allowfrom* especifica la máquina desde la cual se puede conectar ese usuario. La línea *actions* define qué acciones puede realizar ese usuario. La línea *instcmds* define los comandos que pueden ser usados por ese usuario. Por último, la línea *upsmon master* define que ese usuario será el controlador del servidor, y *upsmon slave* será el controlador remoto.

4. Configuración del monitor de SAI

A continuación se debe configurar el fichero */etc/nut/upsmon.conf*. Este fichero contiene la configuración del demonio *upsmon*:

```
MONITOR elsai@localhost 1 control password master
MINSUPPLIES 1
SHUTDOWNCMD "/sbin/shutdown -h +0"
NOTIFYCMD /usr/local/bin/mensaje-ups
POLLFREQ 60
POLLFREQALERT 10
HOSTSYNC 15
DEADTIME 15
POWERDOWNFLAG /etc/killpower
NOTIFYMSG ONLINE "UPS %s en estado normal"
NOTIFYMSG ONBATT "Alguien ha quitado el cable del SAI %s"
NOTIFYMSG LOWBATT "Batería demasiado baja"
NOTIFYMSG FSD "Ha llegado el momento de apagar el SAI"
NOTIFYMSG COMMOK "Comunicación con el SAI restablecida"
```

```
NOTIFYMSG COMMBAD "Comunicación con el SAI no disponible"
NOTIFYMSG SHUTDOWN "Apagando"
NOTIFYMSG REPLBATT "Batería cambiando"
NOTIFYFLAG ONLINE SYSLOG+EXEC
NOTIFYFLAG ONBATT SYSLOG+WALL+EXEC
NOTIFYFLAG LOWBATT SYSLOG+WALL+EXEC
NOTIFYFLAG FSD SYSLOG+WALL+EXEC
NOTIFYFLAG COMMOK SYSLOG+EXEC
NOTIFYFLAG COMMBAD SYSLOG+EXEC
NOTIFYFLAG SHUTDOWN SYSLOG+EXEC
NOTIFYFLAG REPLBATT SYSLOG+EXEC
RBWARNTIME 43200
NOCOMMWARNTIME 300
FINALDELAY 5
```

La línea *MONITOR* especifica que se quiere monitorizar el SAI llamado *elsai*, que está conectado a la máquina *localhost* (localmente), que tiene *1* batería, el login es *control*, el password es *password* y se encuentra en modo *master*.

La línea *NOTIFYCMD*, como parámetro tiene el comando */usr/local/bin/mensaje-ups*. Este es el contenido de ese script:

```
#!/bin/sh

echo "
ATENCION
----------------------

MENSAJE DEL SAI: $NOTIFYTYPE" | mail -s "MENSAJE DEL SAI" admin@dominio.com
```

Cada vez que ocurra un evento, se enviará un mail a la dirección *admin@dominio.com*. El evento se almacena en la variable *$NOTIFYTYPE*

5. Iniciando el servidor

Una vez configurado el SAI y el Servidor, se procede a iniciar el servidor con el comando siguiente:

```
/etc/init.d/nut restart
```

6. Configuración del puesto de control

El puesto de control será la computadora o el equipo desde el cual se monitoreara el SAI. En la configuración ejemplo del servidor corresponde a *maquina5*. Esta computadora será capaz de saber en todo momento el estado del SAI de manera gráfica, a través de gráficos presentados en el navegador web.

En definitiva, las herramientas de control deben ser instaladas en una máquina que tenga un Servidor Web y posibilidad de ejecutar CGIs.

Instalación y configuración

El primer paso es instalar el paquete *nut-cgi* con el comando *apt-get install nut-cgi.* Luego en */etc/nut/hosts.conf* añadir la línea siguiente:

```
MONITOR elsai@172.16.2.2 "SAI Principal"
```

La línea *MONITOR* tan solo contiene el SAI remoto y una descripción del mismo. Como medida de seguridad, los autores de NUT han incluido un fichero llamado *upsset.conf*, en el que se debe descomentar la línea siguiente, cuando se haya configurado de manera correcta el Servidor Web.

```
###
I_HAVE_SECURED_MY_CGI_DIRECTORY
###
```

Luego se procede a abrir el navegador y se escribe la siguiente URL *http://localhost/cgi-bin/upsstats.cgi,* y se visualizara una página con los detalles del SAI, como la carga de la batería, voltaje de entrada, voltaje de salida.

3.6 COSTOS

Para la instalación del Software de Servidor Web, lo que se necesita es tener una computadora conectada a Internet, dicha conexión tiene que ser un enlace dedicado, ya que es necesario que el servidor este disponible las 24 horas. La instalación de la red interna o la intranet de la empresa, es opcional, ya que una empresa que no desea hacer una inversión en una red, ya sea porque no la necesitan o porque no esta contemplado dentro de su presupuesto, bastaría con que tuviera solamente el servidor con salida a Internet. Por esa razón no se contempla los gastos en que se incurriría con la implementación de la red, y se deja a criterio de la empresa.

El software del Servidor Web esta desarrollado en el lenguaje de programación Perl, para trabajar bajo la plataforma o Sistema Operativo, Linux que al igual que Perl, es de libre distribución, es decir no tiene ningún costo de licencia ni costo de adquisición. Es necesario adquirir un equipo o computadora servidor para almacenar el software. Es importante mencionar, que estos costos son recomendaciones, ya que pueden invertir en un potente servidor con grandes capacidades y de marca reconocida, o invertir en otro tipo de computadora servidor, que no necesariamente tiene que ser un equipo especializado o de marca reconocida, pero que si tenga buenas capacidades en lo que a hardware se refiere. Por otra parte, para los requerimientos mínimos de la computadora servidor, se tomo como base, los requerimientos mininos para instalar el sistema operativo (Linux, RedHat 8.0), debido a que es la plataforma bajo la que el software funcionaria.

De la misma manera, es necesario contar con una línea dedicada para el enlace de la computadora servidor a Internet. Las propuestas varían en precio y servicios, algunas proporcionan enlaces, otras ofrecen enlaces IP y tienen la ventaja que ofrece los servicios de Firewall Básicos para seguridad de la red. La inversión en

equipos de red depende si la empresa ya tiene su red instalada o no desea instalar la red, simplemente instalar el servidor directamente a la conexión. Se deja a criterio de la empresa elegir el hardware y el enlace dedicado que mejor se adapte a sus necesidades y presupuestos.

En cuanto a los UPS Inteligentes, sus precios varían en gran manera en el mercado nacional e internacional, con respecto a las capacidades de cada uno de ellos. Para efectos del proyecto realizado es recomendable utilizar un UPS inteligente que posea un puerto serial, un cable para conectarlo a la computadora y el software para monitorear el estado del mismo. Es importante mencionar que esta adquisición es opcional, ya que la empresa o entidad que adquiera el software puede o no implementar el diseño de autorrecuperación por fallas en el suministro de energía planteado. Se deja a criterio de las mismas la adquisición de un ups con estas características. En la Tabla Nº 18, se detallan los costos de inversión para la instalación y funcionamiento del software del Servidor Web.

Requerimiento	Costo
Linux, Distribución Red Hat 8.0	$0.00 (Libre Distribución)
Software de Servidor Web	$0.00 (Comercialización accesible con UDB)[22]
Perl 5.8.0	$0.00 (Libre Distribución)
1 computadora servidor	$987.62
Enlace Dedicado (Instalación)	$151.42 (Costo Promedio)
UPS inteligente	$202.00
Total	**$1,341.04**

Tabla Nº 18. Total Costo de Inversión

A continuación se detallan las especificaciones mínimas sugeridas para el hardware requerido.

[22] Los costos de Comercialización del Software están sujetos al criterio de La Universidad Don Bosco, ya que ellos son dueños de las Patentes de proyectos realizados dentro la misma.

Servidor

Sistema Base: Procesador de 1.70 GHz
Memoria: 512 MB DDR
Disco Duro : 40 GB
Monitor: DELL E551, 15 in
CD- ROM: Unidad CD-ROM 48X
Floppy: Unidad Floppy Disk 1.4 MB
Modem: MDM, V.92, BCOM, 56K
Tarjeta de Red: INTEL PRO-100S W/IPSEC, NIC
Mouse: Logitec System Mouse
Teclado: KYBD 104, US, SLTEK,LC,MG

1 fuente de poder

Gestión inteligente de batería
Puerto interfaz DB-9 RS-232
Incluye software de gestión para monitoreo

Mantenimiento y Mensualidad

El costo por mantenimiento del software de servidor es mínimo, ya que se incluye un manual de usuario, en el que se detalla toda la información necesaria para facilitar la gestión y operación del software, por esta razón, no se requerirá personal altamente capacitado para administrarlo. El mantenimiento del software en cuanto a desarrollo es innecesario, debido a que el prototipo desarrollado, es estable y trabaja de manera eficiente de acuerdo a los requerimientos planteados. En cuanto a la incorporación de nuevas funcionalidades al software, se deja a criterio de la empresa, ya que serán ellos mismos los que realicen el desarrollo de las mismas.

Por otra parte, la adquisición del enlace dedicado incluye un gasto mensual por consumo, el cual tiene un costo promedio de $143.51.

CAPITULO IV. DESARROLLO DEL PROTOTIPO

4.1 DESARROLLO DE LA INSTALACIÓN Y CONFIGURACIÓN

Este es el proceso mediante el cuál el software es instalado en la computadora. Se realiza a través del programa *setup.pl*. El programa define valores o parámetros y solicita otros al usuario para configurar el software. Estos parámetros son los siguientes:

1. La ruta de instalación: esta definida como /usr/bin/swp y no se puede modificar.
2. El puerto de escucha: el valor por default es 80, y se sugiere utilizar este o el puerto 1024.
3. El hostname o nombre de la computadora: es identificado automáticamente.

A continuación el programa procede verificar que el usuario tenga privilegios de root para realizar la instalación, de lo contrario muestra un mensaje de error indicando que debe tener privilegios de root para realizar la instalación. Si el usuario tiene privilegios de root, se crean los directorios y se copian los archivos necesarios para el correcto funcionamiento del servidor. Si en alguno de estos dos procesos ocurriera un error, el programa despliega un mensaje indicando el nombre del directorio o del archivo que no se pudo crear o copiar y la razón, continuando con el siguiente paso en el programa.

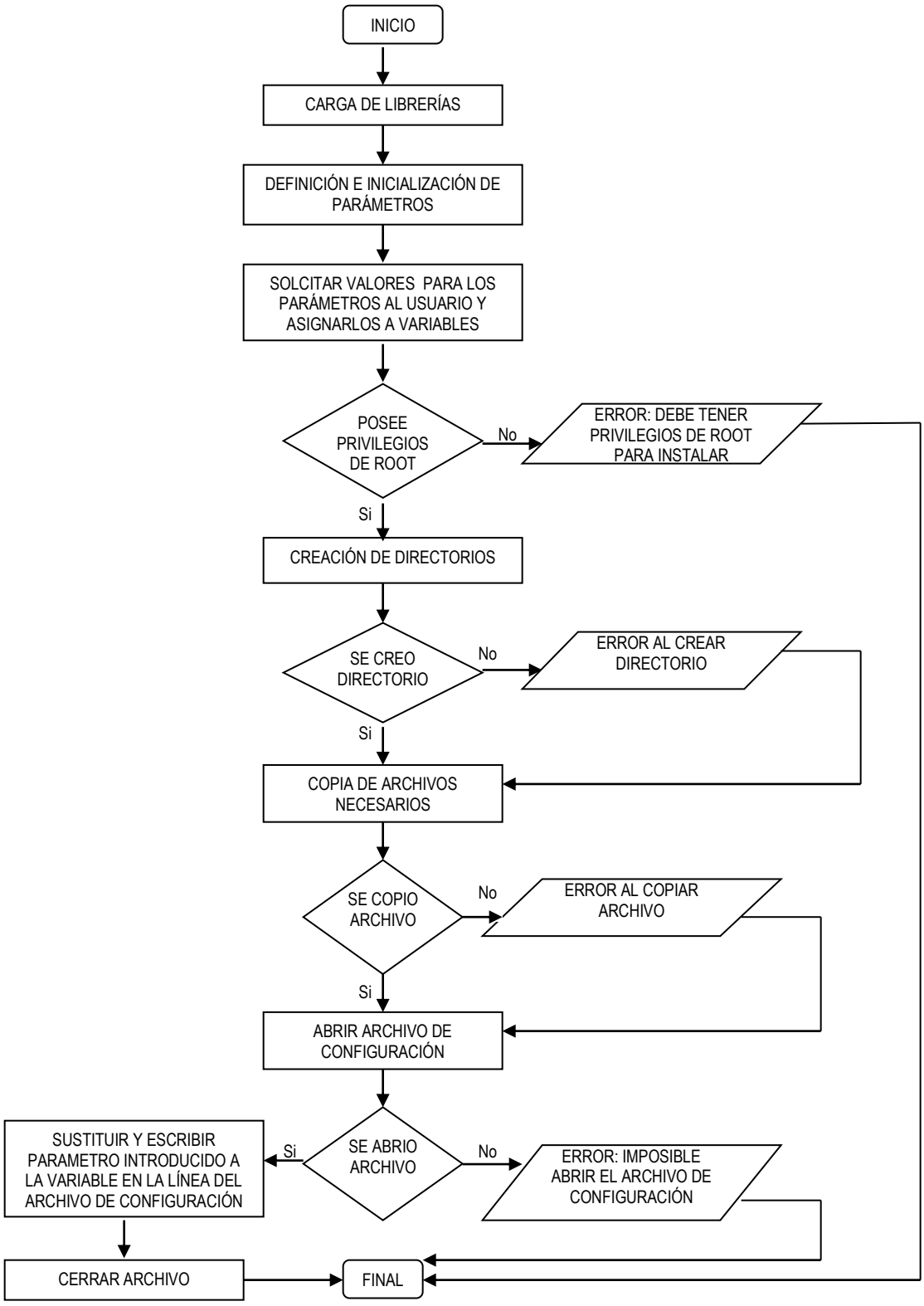
Finalmente, el programa realiza la actualización del archivo de configuración, en el que se almacenan los parámetros que posteriormente serán utilizados por otros programas, para que el software realice sus funciones. Para esto el programa abre el archivo *server.cfg*, y escribe en cada línea los parámetros obtenidos de la instalación. Se ha incluido al final del programa, una parte precedida por la etiqueta <DATA>, que indica al programa que todo lo que esta después de ella, ya

no debe ser ejecutado, pero si puede ser leído. Los pasos que se realizan en la actualización se describen a continuación.

1. Se abre el archivo de configuración.
2. Se lee la línea correspondiente después de la etiqueta <DATA>.
3. Se sustituye en la línea el valor del parámetro de acuerdo a los valores de instalación.
4. Se escribe esta línea en el archivo *server.cfg.*
5. Al no existir más líneas después de la etiqueta, se cierra el archivo.
6. Si el archivo de configuración no pudo ser abierto, se despliega un mensaje indicando que es imposible abrir el archivo de configuración, por lo que se le indica al usuario que debe editarlo manualmente, terminando la ejecución del programa.

A continuación se presenta el flujograma de instalación y configuración del software del Servidor Web.

FIGURA N° 24. Flujograma de Instalación y configuración.



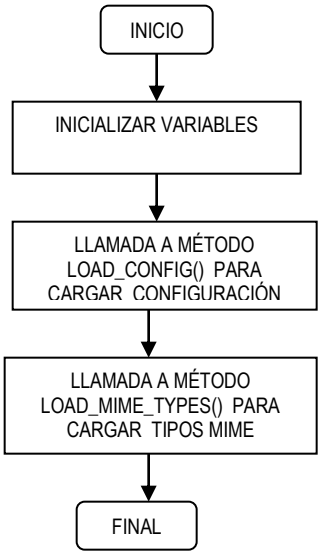
4.2 DESARROLLO DE LA LECTURA DE CONFIGURACIÓN

La lectura de los parámetros de configuración del Servidor Web se realiza a través de la librería *Config.pm*. Los métodos que se implementan en esta librería son llamados o accesados por el programa principal o por otras librerías cuando necesitan identificar parámetros establecidos en el archivo de configuración. La librería implementa dos métodos principales:

- *load_config*: carga la configuración del Servidor Web, establecida en el archivo *server.cfg*.
- *load_mime_types*: accesa a la información sobre los tipos mime soportados por el Servidor Web y establecidos en el archivo *mime.types*.

A continuación se ilustra el proceso que se ejecuta cuando se realiza una llamada a la librería *Config.pm*.

FIGURA N° 25. Flujograma de Lectura de configuración.



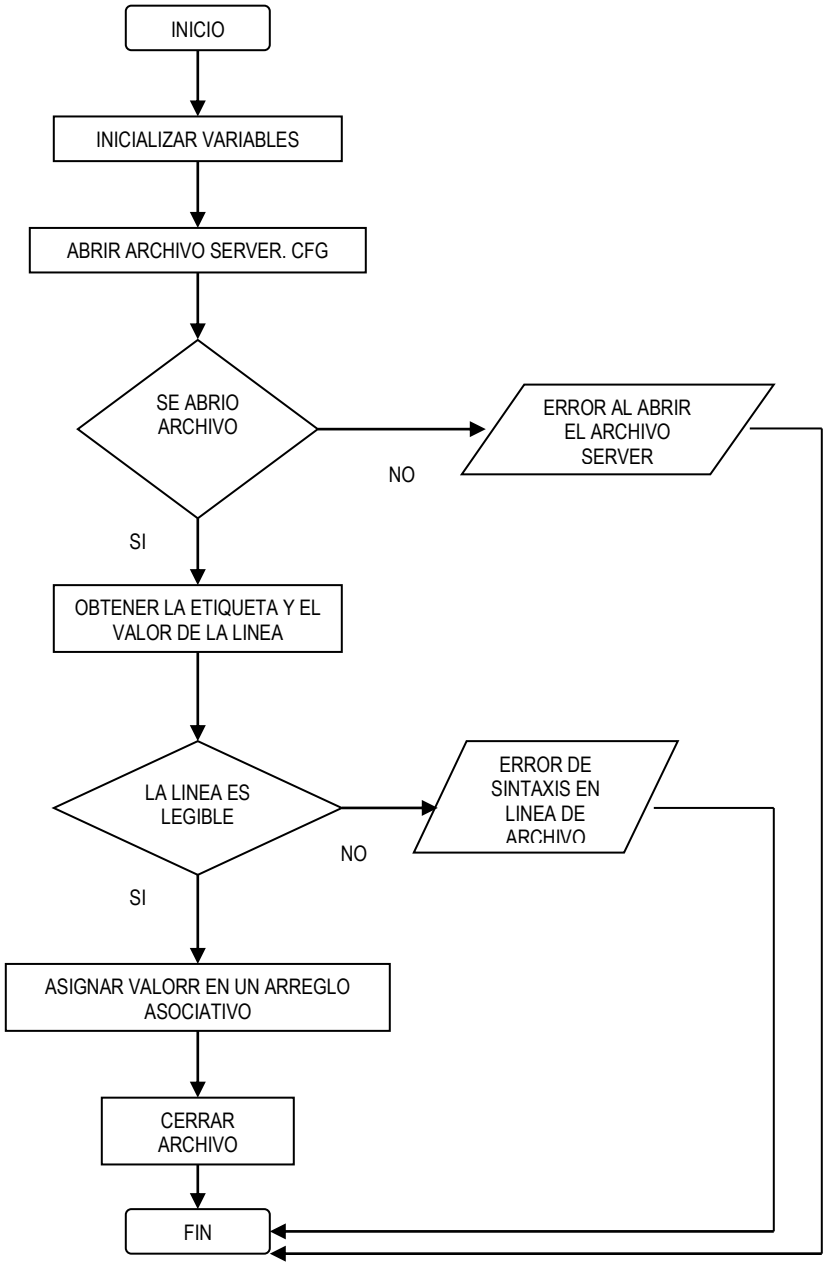
1. Método load_config()

Este método es implementado dentro de la librería C*onfig.pm* y es el que se encarga de accesar al archivo de configuración *server.cfg* para obtener los parámetros de configuración del Servidor Web. A continuación se describe el funcionamiento del método.

1. Se inicializan la variable *$arc_config*, con el nombre del archivo de configuración, así como las variables que se utilizan a lo largo del programa.
2. Se abre el archivo *server.cfg* para lectura.
3. Si el archivo se abrió con éxito, se inicia un lazo que lee cada una de las líneas del archivo para obtener y guardar en un arreglo asociativo, el parámetro y su valor correspondiente. De lo contrario se despliega un error indicando que fue imposible abrir el archivo de configuración, terminando el programa.
4. Si se produce un error al procesar la línea se despliega un error indicando que existe un error de sintaxis en el archivo.
5. Finalmente, se cierra el archivo.

El diagrama de flujo del método *load_config(),* para accesar a la configuración del Servidor Web se muestra en la Figura N° 26.

FIGURA N° 26. Flujograma para cargar información de configuración.



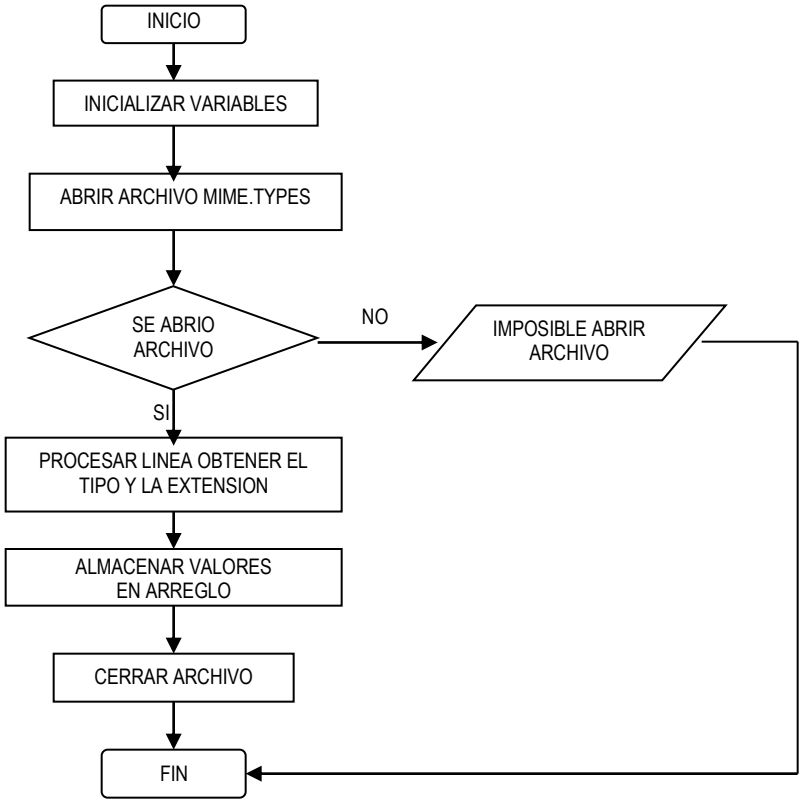
2. Método load_mime_types()

Este es el método implementado en la librería C*onfig.pm*, que se encarga de obtener información sobre los tipos mime en el archivo *mime.types*. A continuación se describe el funcionamiento del mismo.

1. Se inicializan la variable *$arc_mime* con el nombre del archivo de los tipos mime, así como las demás variables que se utilizan a lo largo del programa.
2. Se abre el archivo *mime.types* para lectura.
3. Si el archivo se abrió con éxito, se inicia un lazo que lee cada una de las líneas del archivo para obtener y guardar en un arreglo asociativo, el tipo de dato y la extensión correspondiente. De lo contrario se despliega un error indicando que fue imposible abrir el archivo, terminando el programa.
4. Si se produce un error al procesar la línea se despliega un error indicando que existe un error de sintaxis en el archivo.
5. Finalmente, se cierra el archivo.

El diagrama de flujo del método *load_mime_types(),* para obtener información sobre los tipos mime se muestra a continuación.

FIGURA N° 27. Flujograma para cargar información sobre tipos mime.



4.3 DESARROLLO DEL PROGRAMA PRINCIPAL

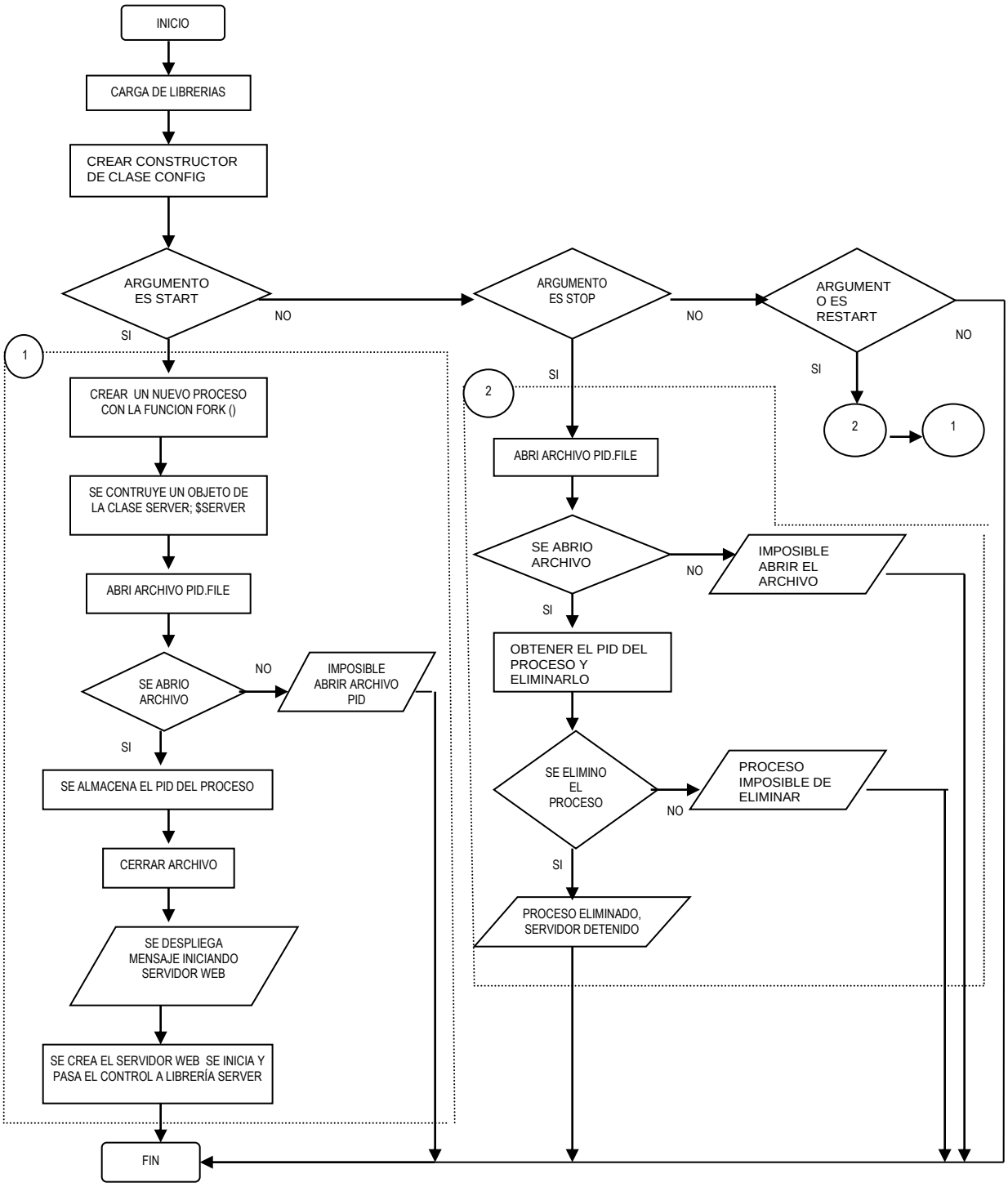
El programa principal llamado *sep.pl*, constituye la parte en la que el Servidor Web es iniciado, detenido o reiniciado. La estructura del programa se describe en el siguiente listado.

1. Se identifica la ubicación del directorio actual para proceder a la carga de las librerías que se utilizaran*, Config.pm, Server.pm, Server2.pm*.
2. Se crea el constructor de la clase o librería *Config* para poder accesar a los parámetros de configuración en cualquier punto del programa.
3. A continuación se procede a identificar el argumento que se recibe de la ventana de comando para conocer la acción a ejecutar.
4. Si el argumento recibido es *start*, se crea un nuevo proceso a través de función de las librerías estándar del perl *fork().*
5. Se construye un objeto de la clase Server, para posteriormente iniciar el servidor y pasar el control a dicha librería.
6. Se identifica mediante el constructor de la librería *Config*, la ruta del archivo *pid.file*, el cuál es abierto para escritura.
7. Si el archivo se abrió satisfactoriamente, se escribe el pid (process id) del proceso que la función *fork()* devuelve. Esto es de utilidad para detener el servidor.
8. A continuación se cierra el archivo.
9. Si el archivo *pid.file* no pudo ser abierto, se despliega un error indicando que es imposible abrir el archivo terminando el programa.
10. Luego de haber almacenado el pid del proceso, se despliega un mensaje en la ventana de consola indicando que el Servidor Web esta siendo iniciado.

11. Con el objeto de la clase Server se accesa al método *Server(),* que es el que crea el Servidor Web, e inicia el lazo para la espera de conexiones entrantes, pasando el control a esta librería.
12. Si el argumento que se recibe de la línea de comando es *stop*, se procede a abrir el archivo *pid.file*.
13. Si dicho archivo es abierto exitosamente, se procede a leer el pid del proceso almacenado, es decir, el pid del proceso que ejecuta el Servidor Web. Luego de leer el pid, dicho proceso es eliminado mediante el uso de la función *kill* de las librerías estándar de perl.
14. Si el proceso fue eliminado con éxito, se despliega un mensaje indicándolo, terminando de esta manera el programa por lo que el Servidor Web es detenido.
15. Si el proceso no pudo ser eliminado, se despliega un mensaje indicando que fue imposible realizar dicha acción, por lo que se debe hacer manualmente, ya que el programa termina, pero el proceso aun sigue activo.
16. Si el argumento recibido es *restart*, se ejecutan los pasos del 12 al 15, y luego los pasos del 4 al 11, que corresponden a detener el Servidor Web e iniciarlo nuevamente.

En la Figura N° 28 se muestra el diagrama de flujo que ejecuta el programa principal *sep.pl*.

FIGURA N° 28. Flujograma del programa principal.



4.4 DESARROLLO DE LA ATENCION DE PETICIONES

Una vez iniciado el Servidor Web, el programa principal *sep.pl* pasa el control a la librería *Server.pm*, en la cuál se han desarrollado los métodos y desde donde se accesan a las otras librerías implementadas para llevar a cabo la atención de peticiones. Los métodos desarrollados dentro de la librería son los siguientes.

- *Server():* La función o el método principal de la librería, desde donde se crea y se ejecuta el Servidor Web y la que controla todo el proceso de atención de peticiones, así como registra la información en los archivos de logeo (access.log y error.log).

- *new_response()*: Este método crea un objeto de la librería estándar de perl HTTP::Response, que se encarga de construir el mensaje de respuesta para el cliente, agregando toda la información de cabecera y de contenido que los mensajes de respuesta necesitan.

- *load_page():* Se encarga de encontrar y cargar la página solicitada. Para encontrar la página este método llama al método *find_file()* de la librería Server2.pm. Por otra parte, en este método es donde se realiza la gestión de autenticación en caso de que el modo protección este activado.

- *envio():* Este es el método que prepara la respuesta a la petición y carga la página solicitada. Esto lo hace a través de las llamadas a los métodos *new_response* y *load_page*, respectivamente.

1. Método Server()

Este constituye la función principal del software, en el que se crea el Servidor Web, se inicia, y se ejecuta el lazo de espera de conexiones entrantes. Por otra parte, dentro de este método es donde se registra la información de logeo, ya sea que se realizó el acceso en el archivo *acess.log* , o si ocurrió algún error durante el proceso, se registra en el archivo *error.log*.

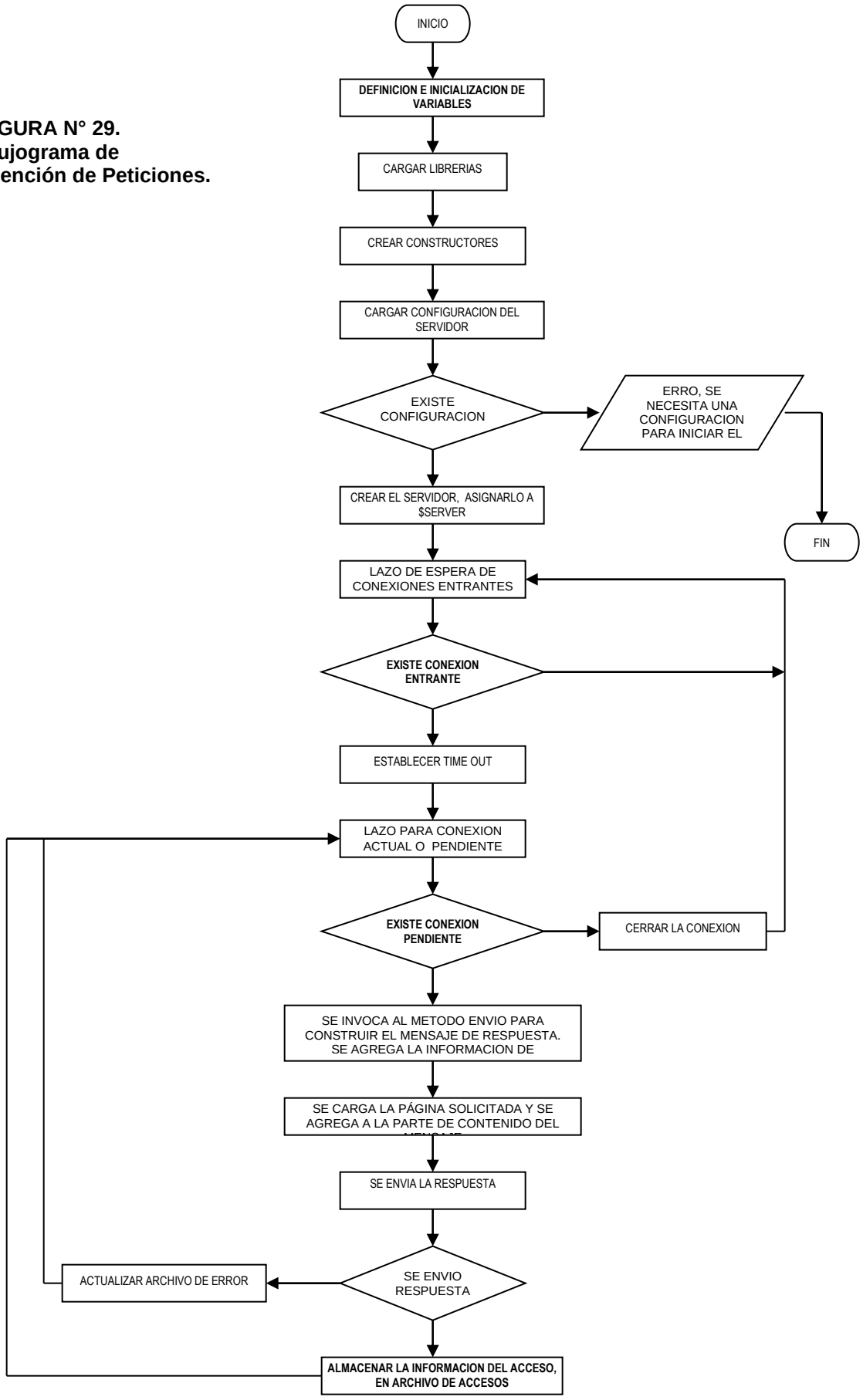
El proceso que se sigue en este método se describe a continuación.

1. Definición e inicialización de las variables.
2. Se realiza la carga de las librerías a utilizar: *Config, Server2 y Logs*.
3. Se crean los constructores de las clases o librerías a utilizar.
4. Se carga la configuración del Servidor Web a través del constructor de clase *Config*.
5. Si existe una configuración se procede a crear el objeto servidor, de lo contrario se despliega un mensaje indicando que se necesita tener una configuración para iniciar el Servidor Web, terminando el programa.
6. Se crea el objeto servidor a través del uso de la clase o librería estándar de perl *HTTP:: Daemon*, asignándolo a la variable $server, a través de la cual se hará referencia al Servidor Web durante la ejecución del programa.
7. Se inicia el lazo de espera de conexiones entrantes, a través del método *accept()*. Con la asignación *$new_sock=$server->accept()*, se indica al Servidor Web que existe una conexión entrante con un cliente, con *$new_s*ock se hace referencia a dicha conexión a lo largo del programa.
8. Se establece el timeout en segundos.

9. Se inicia el lazo que verifica que existe una conexión pendiente con un cliente y que aún espera la respuesta o parte de una respuesta, a una solicitud. Si no existe una conexión pendiente o ya se terminaron de enviar los datos de la respuesta, el programa regresa a la espera de una nueva conexión entrante.
10. Si existe una conexión pendiente, se inicia la construcción del mensaje de respuesta, invocando al método *envio()*, que agrega la información de configuración que se necesita en la parte de la cabecera del mensaje, a través del método *new_response().*
11. Se termina el mensaje de respuesta, en el método *envio()*, agregando la información correspondiente al contenido del mensaje, con la página encontrada. Para encontrar la página se llama al método *load_page()*.
12. Como parte final del método *envio()*, se envía la respuesta a través del método *send_response()* de las librerías estándar de perl, con el objeto *$new_sock*.
13. Si la respuesta se envió satisfactoriamente, se llama al método *LogAccess()* de la librería *Logs*, para registrar la información correspondiente al acceso al Servidor Web. De lo contrario se llama al método *LogError()*, también de la librería *Logs*, ya que un error ocurrió mientras se ejecutaba el proceso. En ambos casos el programa regresa a la espera de nuevas conexiones.
14. Si se termino de enviar la información, se cierra la conexión, es decir se cierra el objeto *$new_sock*, y se regresa al lazo de espera de nuevas conexiones.
15. El programa termina de ejecutarse, cuando se realiza una llamada al programa principal enviando el argumento *stop*.

El flujograma del método *Server()* se muestra en la Figura N° 29.

FIGURA N° 29.
Flujograma de Atención de Peticiones.

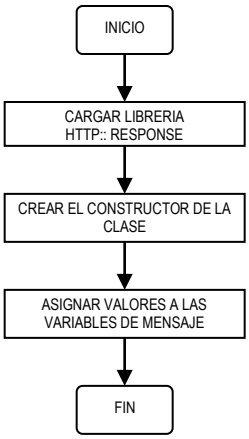


2. Método new_response()

Este método crea un objeto de clase estándar de perl *HTTP::Response*, la cual encapsula respuestas o mensajes HTTP. De esta forma este método es el que agrega toda la información que la cabecera del mensaje de respuesta necesita. Dicho método es ejecutado desde el método *envio().* Su funcionamiento se describe a continuación.

1. Se inicializan las variables con los parámetros recibidos: la url solicitada y los parámetros de configuración.
2. Se carga la librería *HTTP::Response*.
3. Se crea el constructor de la clase para accesar a sus métodos.
4. Se asignan a las variables del mensaje los valores correspondientes. Estas variables del mensaje son: *code*, código de respuesta; *message*, código de mensaje; *protocol,* protocolo utilizado; *content_type,* el tipo de dato que se envía; *date,* la fecha de la petición; *server,* el servidor en que corre el Servidor Web; *header,* estado de la conexión.
5. Finalmente se retorna el resultado.

FIGURA N° 30. Flujograma de Método new_response().



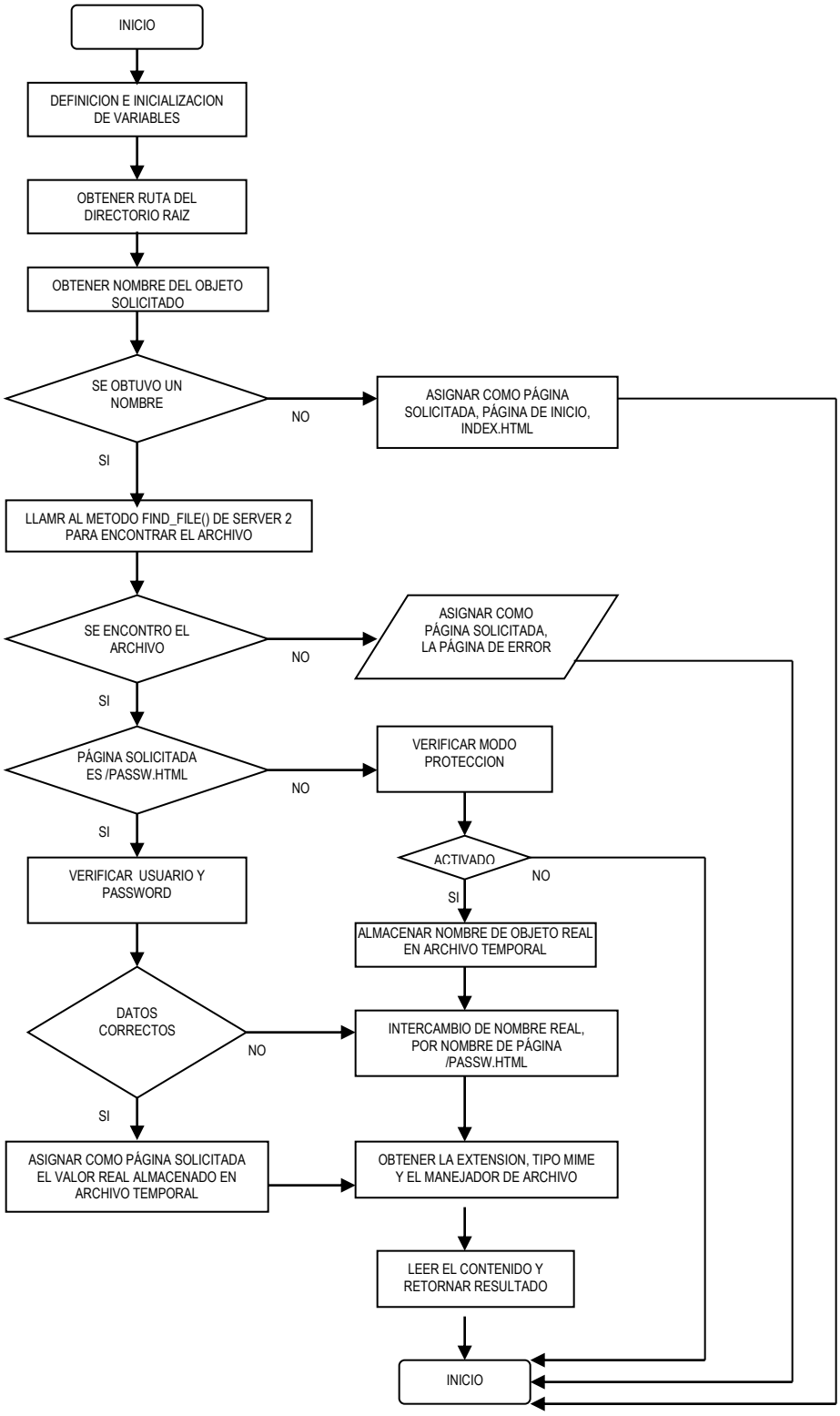
3. Método load_page()

Este es el método en el que se encuentra y se carga el contenido de la página solicitada. Es accesada desde el método *envio()* y recibe dos parámetros; el objeto solicitado y la información de la url. En este método llama a la función o método *find_file()* de la librería *Server2.pm* para encontrar el archivo solicitado. Por otra parte es dentro de este método donde se realiza la verificación del estado del modo protección y se verifica el nombre del usuario y su password en caso de que el modo este activado. El proceso se describe a continuación.

1. Se definen y se inicializan las variables con los parámetros recibidos.
2. Se obtiene de la información de configuración, la ruta del directorio raíz de documentos.
3. Se obtiene de la información del objeto solicitado, el nombre del archivo o página.
4. Si se obtuvo un nombre se procede a encontrar la página solicitada a través de una llamada al método *find_file()* de la librería *Server2*, que se encarga de realizar dicha acción. El contenido retornado de dicha función se asigna a la variable *$tmp_page*, que representa el objeto requerido.
5. Si no se obtuvo un nombre de objeto solicitado, se asigna a la variable *$tmp_page*, el nombre de la página de inicio, *index.html*.
6. Si la función *find_file()* retorna la página encontrada, se identifica si dicha página tiene por nombre */passw.html* que es la página de autenticación, de esta manera se procede a verificar que los datos obtenidos de dicha página, es decir el nombre de usuario y el password estén registrados en el archivo *usuarios.txt*. Si la función *find_file()* no encuentra la página solicitada, envía la página de error.

7. Si el usuario y el password son correctos, se procede a asignar a la variable *$tmp_page,* el valor correspondiente de la página real solicitada, almacenada en el archivo temporal, y se continua con el paso 15. Si el usuario o el password son incorrectos, el nombre del objeto solicitado continua siendo la página de autenticación y esta es la información que se envía en *$tmp_page*.
8. Si la página solicitada no es */passw.html*, quiere decir, que se esta solicitando la página real, o que no se sabe si el modo protección esta activado. De esta manera se procede a verificar si el modo protección esta activado. Si es así, se almacena en un archivo temporal, el nombre del objeto o página real que se solicita.
9. Se intercambia el nombre del objeto real, por el nombre de la página de autenticación */passw.html*, en la que el usuario debe ingresar su nombre de usuario y su password.
10. Luego del paso anterior o si el modo protección no esta activado, se procede a obtener la extensión, el tipo mime del archivo u objeto solicitado, así como el manejador de archivo, en caso de que el tipo identificado lo necesite (si es un script CGI). La identificación de la extensión del archivo se hace invocando al método *get_extention()* de la librería *Server2*. Una vez identificada, se accesa al archivo de configuración para conocer el tipo de dato. Teniendo el tipo de dato se identifica en el archivo de configuración si esta definido un manejador para este tipo de archivo, si es así se obtiene.
11. Finalmente, se lee y se retorna el contenido del objeto o archivo solicitado para ser agregado en la parte de contenido del mensaje de respuesta.

FIGURA N° 31. Flujograma de Método load_page().



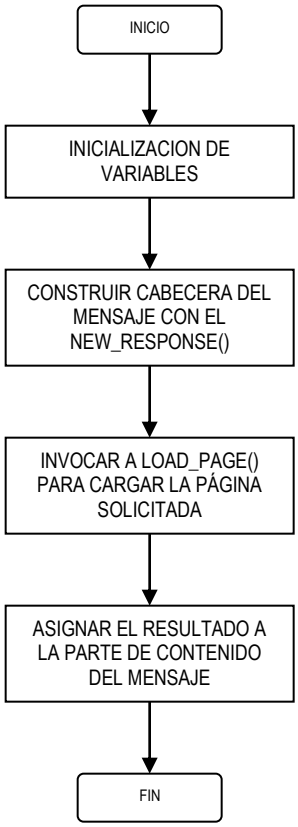
4. Método envio()

El mensaje de respuesta consta de dos partes, la cabecera del mensaje y el contenido del mensaje. En este método es donde se prepara el mensaje de respuesta a la petición, invocando al método *new_response()*; y donde se carga la página solicitada invocando al método *load_page()*. Es llamado desde la función principal *server(),* retornando el mensaje de respuesta construido, para finalmente enviar la respuesta de la solicitud, al cliente. Recibe como parámetros, la información sobre la requisición del cliente y el objeto que hace referencia a la conexión establecida con el cliente. A continuación se describe su funcionamiento.

1. Se inicializan las variables con los parámetros recibidos.
2. Se crea y se asigna el contenido de cabecera del mensaje de respuesta, a la variable *$resp_head*, a través de invocar a el método *new_response().*
3. Se carga la página solicitada, asignando a la variable *$text*, el objeto retornado por el método *load_page().*
4. Se asigna al mensaje de respuesta, la parte correspondiente al contenido del mismo, que esta representado por la variable *$text*.
5. Finalmente se retorna el resultado al método *server().*

En la Figura N° 32 se muestra el diagrama de flujo del proceso que ejecuta el método *envio()*.

FIGURA N° 32. Flujograma de Método envio().



4.5 DESARROLLO DE LIBRERÍA SERVER2

Esta librería implementa dos métodos, uno para encontrar la página o archivo solicitado y el otro para obtener la extensión del archivo.

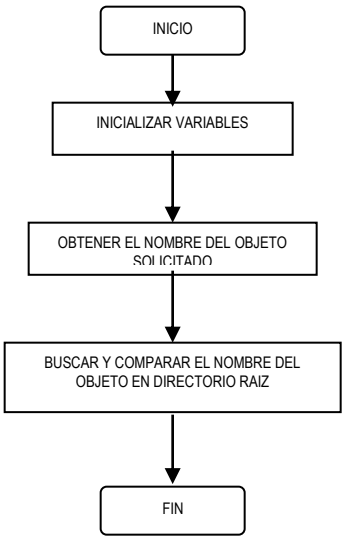
1. Método find_file()

Es invocado por el método *load_page()* de la librería *Server*, para encontrar el archivo o página solicitada en el directorio raíz de documentos. Recibe un parámetro, la ruta del objeto solicitado. Los pasos del proceso se enumeran a continuación.

1. Inicializar variables con la información del parámetro recibido.
2. Separar de la ruta recibida, el nombre del objeto solicitado.

3. Se realiza la búsqueda comparando el nombre del objeto solicitado con los objetos que se encuentran en el directorio raíz de documentos.
4. Se retorna el resultado, el nombre del objeto si se encontró o un objeto null, si el archivo no fue encontrado.

FIGURA N° 33. Flujograma de Método find_file().

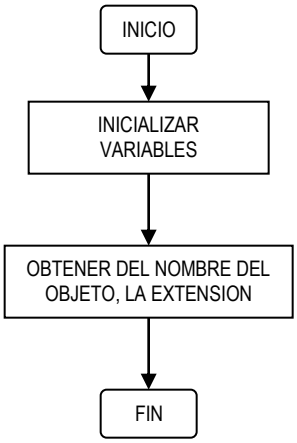


2. Método get_extention()

Al igual que el anterior, es invocado por el método *load_page()* de la librería *Server*, para identificar la extensión del objeto solicitado. Los pasos del proceso se describen a continuación.

1. Inicializar variables con la información del parámetro recibido.
2. Se obtiene a través de la función *substr()* de las librerías estándar de perl, todos los caracteres que se encuentren después de un punto. Estos representan la extensión del objeto solicitado.
3. Se retornan los resultados obtenidos.

FIGURA N° 34. Flujograma de Método get_extention().

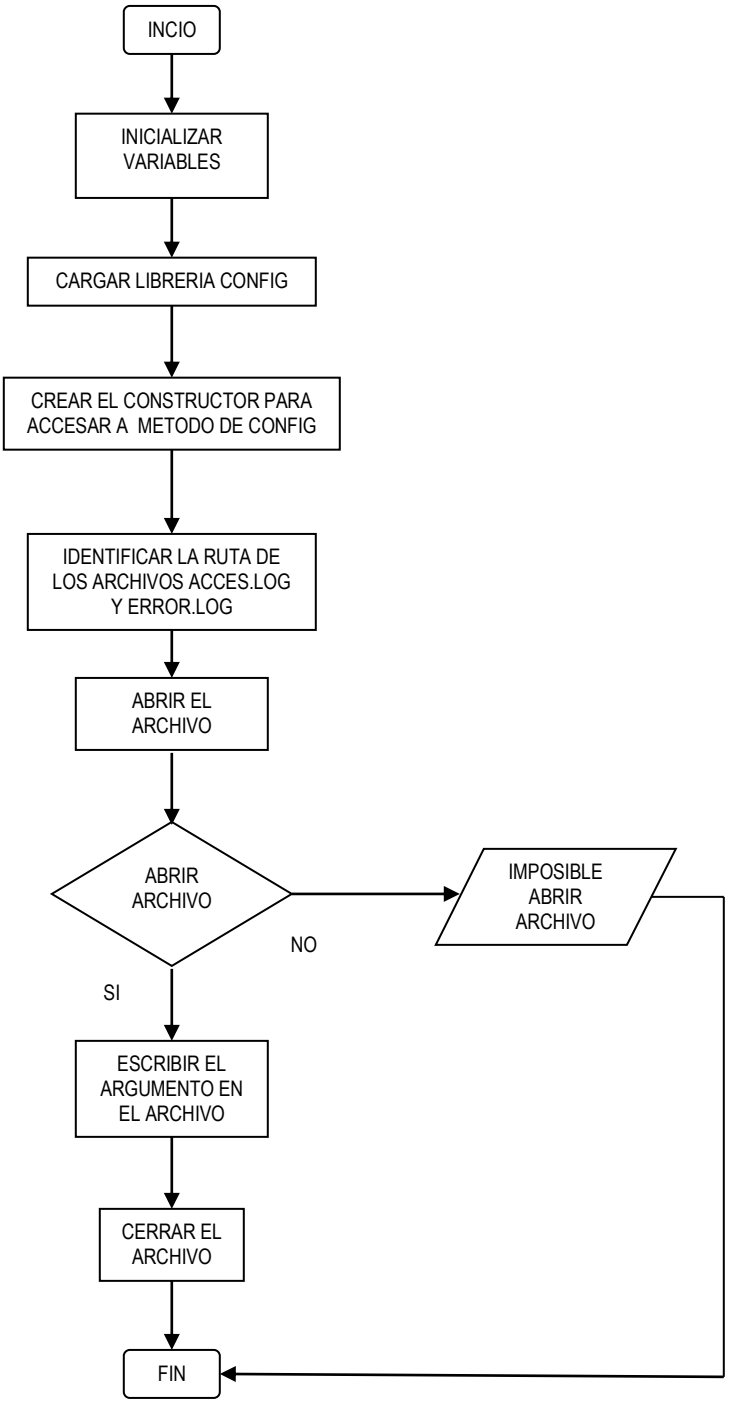


4.6 DESARROLLO DE LIBRERÍA LOGS

Esta librería, es la que define, carga y registra la información en los archivos de logeo del Servidor Web. Dichos archivos son dos, *access.log*, en el que se registran toda la información sobre los accesos al Servidor Web; y *error.log* en el que se registran la información sobre los errores que ocurren en la ejecución del Servidor Web. Se recibe un parámetro, que corresponde a la información correspondiente al acceso o al error. El proceso que se sigue en esta librería se define a continuación.

1. Se inicializan las variables que representan a cada uno de los archivos.
2. Se carga la librería de lectura de configuración, *Config*.
3. Se crea el constructor de la librería *Config*, para conocer la ruta donde se ubican los archivos.
4. Se identifica la ruta del archivo correspondiente.
5. Se abre el archivo para escritura.
6. Si el archivo se abrió con éxito, se escribe la información que contiene el parámetro recibido, de lo contrario se finaliza el método.
7. Se cierra el archivo.

FIGURA N° 35. Flujograma de Librería Logs.



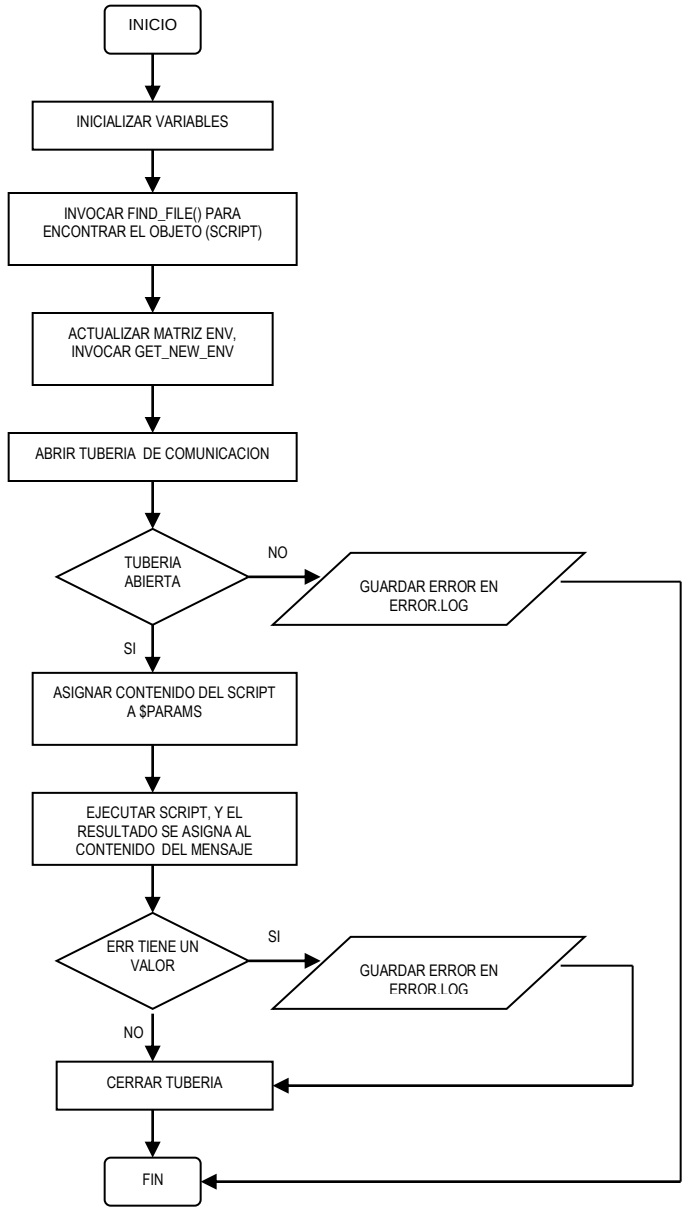
4.7 DESARROLLO DEL MODULO_CGI

Este módulo es un archivo de extensión .pm (perl module) que implementa el manejo de CGI (Common Gateway Interface), que es un estándar para comunicar aplicaciones externas con un Servidor Web. Se ejecuta desde el método *exec_cgi()*. Para la implementación de este módulo se ha utilizado una tubería como canal de comunicación, a través de la cual se ejecuta el script. A continuación se describe el proceso.

1. Se inicializan las variables.
2. Se recibe como parámetro la información de cabecera correspondiente a la url del objeto solicitado. Con esta información se encuentra el archivo, invocando al método *find_file().*
3. Se actualiza el arreglo *ENV*, que contiene toda la información del entorno, para el script cgi, invocando al método *get_new_env().*
4. Se abre la tubería para la ejecución del script y se definen 3 conexiones, una para entrada (se hace referencia por el manejador IN), salida (se hace referencia por el manejador OUT) y para la captura de errores (se hace referencia por el manejador ERR). A partir de ese momento la tubería ejecutará los comandos que reciba del script en perl, ya que este le proporciona la entrada estándar.
5. Si se logro abrir la tubería, se asigna el contenido del archivo a la variable *$params*. De lo contrario se almacena en el archivo correspondiente, el error.
6. Se ejecuta el script a través de la rutina *print OUT $params*. La palabra reservada *print* es la que ejecuta, *OUT*, indica que es salida de datos y *$params* representa los comandos que se ejecutan. El resultado del script es asignado como contenido del mensaje de respuesta.
7. Si *ERR* contiene un valor, quiere decir que un error ha ocurrido al ejecutar el script, por lo que se almacena en el archivo correspondiente, el error

8. correspondiente. En este caso el srcipt no se ejecuta, por lo que el programa asume como contenido, el texto del script.
9. Se cierra la tubería y las conexiones en la misma.

FIGURA N° 36. Flujograma Módulo CGI.



1. Método get_new_env()

Este es invocado por el método *exec_cgi()*, y actualiza la matriz asociativa %ENV, que es la que almacena los valores de entorno de un script cgi. La ejecución del método consiste en la asignación a cada una de las variables, el valor correspondiente del script cgi. Las variables de entorno que se actualizan son las siguientes.

GATEWAY_INTERFACE: Versión de la aplicación CGI.
DOCUMENT_ROOT: Directorio Raíz de documentos.
REQUEST_METHOD: Método solicitado GET oPOST.
CONTENT_LENGTH: Tamaño del contenido.
CONTENT_TYPE: Tipo o formato del Contenido solicitado.
HTTP_ACCEPT: Los tipos de datos que se han aceptado.
HTTP_USER_AGENT: El nombre y la Versión del Navegador utilizado.
HTTP_AUTH_TYPE: Si esta definido, el tipo de autorización.
QUERY_STRING: Los datos solicitados.
SCRIPT_NAME: Nombre del script que se ejecuta.
SERVER_NAME: El nombre del servidor en que se ejecuta el script.
SERVER_SOFTWARE: El nombre del software del Servidor Web.
SERVER_PORT: El puerto en el que corre el Servidor Web.
SERVER_PROTOCOL: El protocolo utilizado.
REMOTE_HOST: El nombre del host remoto.
REMOTE_ADDR: La Dirección IP del host remoto.

CONCLUSIONES

1. Internet es un medio de comunicación que ofrece valiosas oportunidades para la introducción de las Empresas Salvadoreñas al mercado global, proporcionándoles un incremento en sus ingresos y ayudándoles a potenciar la satisfacción de sus clientes, con el acceso a mejores comunicaciones, bienes y servicios, con una disponibilidad de 24 horas, 7 días a la semana, impulsando de esta manera, tanto su crecimiento económico y tecnológico, como el del país.

2. En el mundo de la informática, el software de computadora conocido como Servidores Web, constituyen la herramienta de accesibilidad para que una persona o empresa publiquen sus Páginas Web, ofreciendo información de importancia al mundo, a través de la construcción de Sitios Web.

3. La mayor limitante que existe para las Medianas Empresas Salvadoreñas de El Salvador, en cuanto al aprovechamiento que trae consigo la utilización de Internet como herramienta de trabajo, lo constituyen la falta de alternativas orientadas a las capacidades económicas, tecnológicas y de mantenimiento de las mismas, así como el poco conocimiento sobre los beneficios, las ventajas, y la implementación de las arquitecturas tecnológicas basadas en código libre o código abierto.

4. El prototipo desarrollado en esta investigación contempló la construcción de un modelo básico que incluyen algunas de las características que debe poseer todo Servidor Web para llegar a ser un software totalmente funcional. El mayor aporte consiste en el establecimiento de un precedente para el estudio y desarrollo de este tipo de tecnologías, y en servir de incentivo para la realización de este tipo de proyectos destinados al beneficio del sector empresarial salvadoreño.

5. El prototipo realiza la atención de peticiones por parte de los clientes, manejando de manera eficiente la presentación de Páginas Web con texto e imágenes, Páginas Web con formularios para el envío de información y archivos que representan scripts cgi escritos en Perl. Por otra parte, se incluye la gestión de acceso al Servidor Web, a través de la definición de un modo de protección que impide que usuarios no registrados, tengan acceso a los recursos de información que el Servidor Web almacena.

6. El empleo de herramientas de libre distribución para la construcción de la arquitectura sobre la que el software del Servidor Web opera; sistema operativo y herramienta de desarrollo; proporciona ventajas en la reducción de costos de inversión y de adquisición. Por otra parte, la implementación del diseño del esquema de Firewall planteado, contribuirá a reducir dichos costos, ya que se ha realizado utilizando las herramientas de firewall que el sistema operativo incorpora, las cuales son muy eficientes y proporcionan un nivel de seguridad bastante aceptable. De esta manera, no se tendrá que en invertir en software especializado y de alto precio para la seguridad de la red.

7. Se presenta una propuesta de desarrollo de un mecanismo de autorecuperación del Servidor Web en caso de fallas debidas a factores externos, tales como fallas en los dispositivos de almacenamiento masivo y fallas en el suministro de energía eléctrica (ver sección 3.5). La propuesta definida para el desarrollo de dicho mecanismo en esta investigación, pretende garantizar la mayor disponibilidad posible del Servidor Web en Internet. Por otra parte, la implementación del mismo contribuirá a la reducción de costos de mantenimiento ya que no será necesario contar con una vigilancia constante por parte del personal de informática.

RECOMENDACIONES

1. Es importante para el desarrollo económico y tecnológico del sector empresarial salvadoreño, incluir dentro de los planes de estudio superior, la promoción de investigaciones y proyectos de desarrollo dirigidos al mejoramiento o construcción de la infraestructura tecnológica del sector. De la misma manera, resulta importante promover el seguimiento de estas investigaciones y proyectos, por medio de estudios que estén enfocados en la búsqueda de elementos y métodos que mejoren la funcionalidad del proyecto original.

2. La promoción de proyectos enfocados a difundir las ventajas y beneficios, al sector empresarial salvadoreño; de los productos de libre distribución y los procedimientos a seguir, para la implementar una arquitectura basada en ellos; es de mucha importancia, ya que se contribuirá al aumento de las alternativas que actualmente se conocen dentro del mercado actual, de entre las cuales, las empresas puedan elegir la que más se adecue a sus capacidades económicas y tecnológicas.

3. En futuras mejoras al prototipo desarrollado, se recomienda desarrollar los módulos que implementen el manejo de scripts hechos en lenguajes de programación diferentes a Perl, además de incluir módulos para el soporte de bases de datos.

4. Para garantizar la seguridad e integridad del contenido del Servidor Web, es fundamental implementar el diseño del firewall planteado en el manual del usuario, y el mecanismo de autorecuperación de fallas debidas a factores externos; así como la búsqueda de aspectos y elementos que contribuyan a mejorarlos. Ambos diseños aumentaran el nivel de seguridad y de disponibilidad a tiempo completo, del Servidor Web. Por otra parte, es

recomendable construir un esquema para encriptar las contraseñas de los usuarios registrados en el archivo correspondiente, para accesar al Servidor Web cuando el modo protección esta activado.

5. En cuanto a la seguridad del trafico que circula a través de la red, debe realizarse una investigación para realizar un esquema basado en los protocolos de seguridad como el SSH, SSL, Kerberos, TSL. El sistema operativo Linux, posee herramientas para implantar dicho esquema, por lo que debe hacerse un estudio de las mismas con el fin de determinar el mejor esquema a implementar.

6. Se recomienda utilizar los valores sugeridos en la documentación y en los scripts mismos, para la instalación del software, ya que estos se apegan a los estándares establecidos por otros Servidores Web funcionales.

BIBLIOGRAFÍA

Libros

Stout, Rick. OPTIMIZACION DE SERVIDORES WEB. ANÁLISIS Y ESTADÍSTICAS. Editorial McGraw-Hill, Segunda Edición, 1997.

Bonilla, Gildaberto. ESTADISTICA I. ELEMENTOS DE ESTADISTICA DESCRIPTIVA Y PROBABILIDAD. UCA Editores, Primera Edición, 1993.

Witthen, Jefrrey. ANÁLISIS Y DISEÑO DE SISTEMAS DE INFORMACIÓN. Editorial McGraw-Hill, Tercera Edición, Agosto 1997,Colombia.

E.B. Pineda, E.L. de Alvarado, F.H. de Canales. METODOLOGÍA DE LA INVESTIGACIÓN, Segunda Edición OPS, 1994.

Documentación de Empresas

Documentación proporcionada por CONAMYPE ("Comisión Nacional de la Micro y Pequeña Empresa"), sobre Empresas registradas en El Salvador, año 2,000.

Personas Entrevistadas

Ing. Carlos Amaya

Sr. Alejandro Mayorga

Direcciones de Internet

http://www.conamype.org

http://www.cdec.unican.es

http://www.turismo.uma.es

http://www.search.cpan.org

http://www.iespana.es/perl-es/

http://www.mrbit.es/hsa/vai/muestreo/
http://www.exitoexportador.com
http://home.fundes.org
http://www.redhat.es
http://www.linux.org
http://standards.ieee.org/guides/style/
http://www.cisco.com
http://www.apache.org
http://www.etsimo.uniovi.es/perl/tutor/
http://www.lfcia.org/openprojects/camllets/doc/html/node1.html
http://www.activeperl.com
http://es.tldp.org
http://docs.kde.org
http://europe.redhat.com

GLOSARIO

A

ANTIVIRUS, Programa de software para la protección/Remoción de programas indeseados (Virus Informático).

ADAPTADORES DE RED, Dispositivo que conecta un equipo (por ejemplo un PC) a la red y controla el protocolo eléctrico para la comunicación con esa red; también se denomina tarjeta adaptadora de red, o NIC.

ANCHO DE BANDA, La máxima cantidad de datos que un cable de red puede transportar, medido en bits por segundo (bps).

B

BPS, (Bits por segundo) es la unidad en que se mide la velocidad de transferencia efectiva de un módem o de una conexión serie.

BROWSER (NAVEGADOR), Aplicación de software para navegar por Internet.

BYTE, Unidad de información que equivale a ocho bits.

C

CABLE UTP (PAR TRENZADO SIN PROTECCIÓN), es un conjunto de cables de pares trenzados dentro de una cubierta plástica. El uso mas común de este tipo de cables es para cableado telefónico.

CACHE, 1.(memoria de visitas) Es una copia que el navegador mantiene en la computadora, de las páginas visitadas recientemente, de esta manera si el usuario requiere volver a entrar a estos sitios ya lo hará a través de su disco duro y no desde Internet. La ventaja de este tipo de memoria es que disminuye el tiempo de carga de las páginas. 2. Igual que los procesadores, los discos duros tienen un pequeña caché. La caché es un tipo de memoria especial situada entre el procesador y la memoria principal de la computadora. El procesador trabaja efectuando operaciones cada microsegundos; la memoria funciona en orden de nanosegundos. Para mantener al procesador ocupado se le deben de suministrar

datos e instrucciones de forma continua a gran velocidad. Esta es la función de la caché.

CGI (Common Gateway Interface), Es un interfaz que sirve para que los programas externos puedan correr o ejecutarse, bajo un servidor de información. Actualmente, los servidores de información soportados son http.

CLIENTE/SERVIDOR, Este término define la relación entre dos programas de computación en el cual uno, el cliente, solicita un servicio al otro, el servidor, que satisface el pedido.

COMANDO, Instrucción que un usuario da al sistema operativo de la computadora para realizar determinada tarea.

COMERCIO ELECTRÓNICO, Se llama así al conjunto de transacciones comerciales que se realizan por medio de Internet. Generalmente los usuarios compran con su tarjeta de crédito

CONACYT, Comisión Nacional de Ciencia y Tecnología.

CONSOLA (VENTANA DE), Ver SHELL.

CPU (CENTRAL PROCESSING UNIT), Unidad central de proceso. En el caso del PC, es la computadora propiamente dicha, al que acompaña el monitor, el teclado y el ratón. Es el procesador que contiene los circuitos lógicos que realizan las instrucciones de la computadora.

D

DIAL UP, Acceso a Internet por marcado telefónico

DIRECCIÓN IP (INTERNET PROTOCOL), Es una dirección única a nivel mundial que se asigna a los dominio y subdominios.

DISCO DURO, Es un dispositivo de almacenamiento, que nació como evolución del diskette. Tiene una capacidad mucho mayor y es mucho más rápido, pero no está diseñado para ser trasladado de lugar, sino para permanecer dentro de la computadora.

DOMINIO, Conjunto de caracteres que identifica la dirección de un Sitio Web. En Internet, nombre con letras que equivale a la expresión numérica de una dirección IP.

DNS (DOMAIN NAME SYSTEM), Método de identificación de una dirección de Internet.

E

ENCRIPTAR, Proteger archivos expresando su contenido en un lenguaje cifrado. Los lenguajes cifrados simples consisten, por ejemplo, en la sustitución de letras por números.

ENLACE, Imagen o texto destacado, mediante subrayado o color, que lleva a otro sector del documento o a otra Página Web.

ESTACIÓN DE TRABAJO, Es un dispositivo que hace uso de los servicios de red ofrecidos por el servidor de archivos.

F

FIREWALL, Aplicación de seguridad que impide accesos no autorizados a determinadas partes de una red.

FTP (FILE TRANSFER PROTOCOL), Protocolo – o servidor de procesos – de trasferencia de ficheros entre dos servidores. Sirve para enviar y recibir archivos de Internet.

FUNDAPYME, Fundación para la Pequeña y Mediana Empresa.

G

GB, Abreviatura de GigaByte.

GIGABYTE (GB), Unidad de medida de una memoria. 1 gigabyte equivalen a 1024 megabytes que, a su vez equivalen a 1.073.741.824 bytes.

H

HARDWARE, Artículos, accesorios y dispositivos tangibles de los equipos de computación.

HIPERVÍNCULO, Documento con Hipertexto que con un clic en alguna de sus partes traslada al usuario a otro documento o a otra parte del mismo documento.

HOSTING, Servicio de alquiler de servidores informáticos a terceros.

HTML (HYPER TEXT MARKUP LANGUAGE), Lenguaje para escribir páginas en Internet.

HTTP (HYPERTEXT TRANSFER PROTOCOL), Protocolo de transferencia de hipertexto a través del cual se trasmiten datos por la red.

HUB (CONCENTRADOR), Los hubs o concentradores conectan entre si a los componentes de una red y proporcionan conexión compartidas a los nodos.

I

INTERFACE (INTERFAZ), Entorno de interacción entre el usuario y la computadora

INTERNET, Red de computadoras a nivel mundial, con el fin de compartir, divulgar información, comercialización de productos ya sea con fines lucrativos, educativos o recreativos, conocido también como WEB.

IP (INTERNET PROTOCOL), Protocolo de comunicación entre servidores y redes.

ISO, Organización Internacional de Estándares

ISP (INTERNET SERVICE PROVIDER), Proveedor de servicios de Internet

K

KILOBYTE (KB), Unidad de medida de una memoria. 1 kilobyte equivalen a 1024 bytes.

KERNEL, Ver NÚCLEO.

L

LENGUAJE DE PROGRAMACIÓN, Sistema de escritura para la descripción precisa de algoritmos o programas informáticos.

LINK, Ver ENLACE.

LINUX, Sistema Operativo cuyos códigos de desarrollo son distribuidos gratuitamente.

LOGIN, Identificación y entrada de un usuario en una red.

M

MACOS, Sistema operativo de las computadoras Apple Macintosh.

MEGABYTE, Múltiplo del byte. Unidad de información que equivale a 1.000 bytes.

MEMORIA RAM (RANDOM ACCESS MEMORY), Memoria donde la computadora almacena datos que permiten al procesador acceder rápidamente al sistema operativo las aplicaciones y los datos en uso.

MODELO OSI, El modelo OSI (Sistema Abierto de Interconexión) es el modelo más usado para interconexión de redes. La organización internacional para la estandarización (ISO por sus siglas en ingles) desarrollaron el modelo OSI, el cual sirve como referencia para tareas de comunicación .

MODEM, Consiste en un dispositivo que permite que la computadora se comunique a través de líneas telefónicas, convirtiendo las señales digitales en análogas y viceversa (modulador –demoulador).

MULTIPLATAFORMA, Compatible con entornos informáticos de distintos fabricantes.

MULTITAREA, Es cuando una computadora es capaz de realizar más de una tarea a la vez. Puede ser en paralelo (si tiene más de un procesador) o concurrente (si sólo tiene uno).

N

NÚCLEO, Parte principal de un sistema operativo, encargado del manejo de los dispositivos, la gestión de la memoria, del acceso a disco y en general de casi todas las operaciones del sistema que permanecen invisibles para nosotros.

O

OS/2, Sistema operativo multitarea de IBM creado para computadoras PC, hoy en día en desuso.

P

PÁGINA WEB, Una de las páginas que componen un sitio de la World Wide Web Un Sitio Web agrupa un conjunto de páginas afines. A la página de inicio se la llama home page.

PID, Process IDentification. Numero que identifica un proceso en el sistema, este numero es único para cada proceso.

PROTOCOLO, Conjunto de normas y reglas del lenguaje para la conexión a una red.

PROTOCOLO TCP, (Trasmisión Control Protocol). Protocolo de control de la Transmisión. Es un protocolo orientado a conexión y proporciona mecanismos que ofrecen seguridad en cuanto a la entrada del paquete o paquetes en el destino. Además de su capacidad de ordenamiento y no duplicación .

PUERTO, En una computadora, es el lugar específico de conexión con otro dispositivo, generalmente mediante un enchufe. Puede tratarse de un puerto serial o de un puerto paralelo.

R

RAID, Siglas en inglés de Redundant Array of Independent Disks, en español, Arreglo Redundante de Discos Independientes. RAID es una matriz de múltiples

discos duros independientes que proporcionan un alto rendimiento y una tolerancia de fallas.

RED, Una red es un sistema de transmisión de datos que enlaza varias computadoras, dispositivos periféricos y líneas de transmisión creando un ambiente de conjunto para el procesamiento de datos. Consta de tarjeta de interfaz, cables y software de transmisión y sistemas operativo de redes.

ROOT, Persona o personas encargadas de la administración del sistema Tiene todo el privilegio para hacer y deshacer, por lo que su uso para tareas que no sean absolutamente necesarias es muy peligroso.

S

SAI, Un Sistema de Alimentación Ininterrumpida (SAI), en ingles Uninterruptible Power Supply (UPS),

SERVIDOR, Equipo de computación que funciona como gestor y dador de recursos. Sistema dedicado a dar servicios de software y comunicaciones a los diferentes computadoras conectadas en una red

SHELL, Traducido del inglés concha o caparazón. El shell es el intérprete de comandos que se establece entre el usuario y el kernel.

SISTEMA OPERATIVO, Programa que administra los demás programas en una computadora.

SOCKET, Ver Puerto.

SOFTWARE, Conjunto organizado de instrucciones para computadoras, con el fin de desarrollar una actividad determinada en el computador.

SV-NET, Red Nacional promovida por el CONACYT, para organizar los subdominios en El Salvador.

T

TUBERÍA, Las tuberías son conexiones entre procesos. La salida de un proceso se encadena con la entrada de otro, con lo se puede procesar unos datos en una sola línea de comando.

U

UPS, Ver SAI.

W

WINDOWS, Sistema operativo para computadores personales, creado y diseñado por Microsoft Corporation.

ANEXOS

ANEXO I

MODELO DE CUESTIONARIO PARA ENTREVISTAS A EXPERTOS

1. ¿Conoce el funcionamiento de los Servidores Web (el proceso que realizan)?
2. ¿Conoce sobre la Arquitectura interna de los Servidores Web a nivel de software ?, Describirla.
3. ¿Cuáles son los componentes mínimos o básicos de esa arquitectura?
4. ¿Considera que es necesario agregar algún componente a la arquitectura general de los servidores web? ¿Cuáles? Razones.
5. ¿Cuál considera que es el lenguaje de programación adecuado para el desarrollo de este tipo de software?
6. ¿Qué opina del lenguaje de programación, Perl (si no fue el sugerido)?
7. ¿Si usted tuviera que realizar este software, cuáles serian los pasos que seguiría en el desarrollo?
8. Sugerencias.

ANEXO II

MODELOS DE ENCUESTAS

ENCUESTA USUARIO TÉCNICO

La presente encuesta tiene como objetivo recolectar información confiable que permita analizar el impacto de Internet y los beneficios que ofrece para la Mediana Empresa en El Salvador, así como determinar el tipo de Tecnología que utilizan para mantenimiento de su Sitio y sus capacidades de Inversión. La información que proporcione será de mucha importancia para realizar un proyecto relacionado con el tema, por lo que le agradeceríamos responder en forma objetiva las preguntas que se le formulen. Por su colaboración, Gracias.

1. ¿En su Empresa tienen conexión a Internet ? Si su respuesta es sí continué, si es No, ir a la pregunta 4.
   a. ❑ Sí b. ❑ No

2. ¿Cuál es el fin principal de acceso a Internet en su empresa? (marque todas las que apliquen)
   a. ❑ Correo Electrónico b. ❑ Labores Educativas c. ❑ Compras d. ❑ Entretenimiento
   e. ❑ Trabajo f. ❑ Noticias g. ❑ Búsqueda de información h. ❑ Pasatiempo
   i. ❑ Chat j. ❑ Otro_____

3. ¿ Que tipo de información busca regularmente en Internet su empresa ? (marque todas las que apliquen)
   a. ❑ Noticias y Actualidad b.❑ Compras c.❑ Deportes d.❑ Cultura y Arte
   e. ❑ Cine f. ❑ Búsqueda de Empleo g.❑ Música h.❑ Juegos
   i. ❑ Política j. ❑ Eventos k.❑ Educación l. ❑ Ciencia y Tecnología
   m.❑ Economía y Finanzas n.❑ Computación o.❑ Clasificados p.❑ Comercio Electrónico
   q. ❑Otras r. ❑ Ninguna

4. ¿ Su empresa Utiliza la Publicidad en Internet, a través de banners u otros métodos, en algún sitio de Internet?
   a. ❑ Sí b. ❑ No

5. ¿ Posee su empresa Pagina Web o Sitio Web? (si su respuesta es no, pase a la pregunta 15)
   a. ❑ Sí b. ❑ No

6. ¿Por qué razón su empresa tiene presencia en Internet?
   a. ❑ Imagen b.❑ Información sobre productos/servicios c. ❑ Publicidad
   d. ❑ Servicio al Cliente e.❑ Ventas En Línea f. ❑ Comunicación Interna
   g. ❑ Otro _____

7. ¿ Como mantiene su Sitio Web o su Página Web (Si posee) ?
   a. ❑ WebHosting Propio b. ❑ WebHosting Con otra Empresa c. ❑ WebHosting Gratis
   d. ❑ Otro
   _____

8. ¿ Que tipo de Servidor Web Utiliza (si posee WebHosting Propio)?
   a. ❑ Código Libre (Ej. Apache, etc) b. ❑ Bajo Licencia (Ej. IIS, etc) c. ❑ Otro_____

9. ¿El Servidor Web que utiliza cumple con sus expectativas de funcionalidad?
   a. ❑ Sí b. ❑ No

10. ¿ Que tipo de problemas son los más frecuentes?
    a. ❑ Funcionalidad b. ❑ Operabilidad c. ❑ Velocidad d. ❑ Costos e. ❑ Mantenimiento
    f. ❑ Otro _____

11. ¿ Su empresa estaría dispuesta a experimentar con otras alternativas tecnologías en cuanto a lo que a Servidores Web se Refiere? ¿Por qué Razón?
    a.❑ Sí b.❑ No

12. ¿ Que tipo de Servidor Web Elegiría?
    a. ❑ Código Libre b. ❑ Bajo Licencia c. ❑ Otro _____

13. ¿ Esta enterado de las tarifas que cobran las Empresas Salvadoreñas que brindan el servicio de WebHosting?
    a. ❑ Sí b. ❑ No

14. ¿ Como le parecen las tarifas? (Si Posee Sitio Web, es la última Pregunta)
    a. ❑ Altas b. ❑ Bajas c. ❑ Justas

15. ¿Estaría interesado en tener presencia en Internet (si no posee Sitio Web)? ¿Por qué razón?
    a. ❑ Sí b. ❑ No

    Razones
    a. ❑ Ventas En Línea b.❑ Publicidad c. ❑ Información sobre productos/servicios
    d.❑ Servicio al Cliente e. ❑ Imagen f. ❑ Otro_____

16. ¿ Como mantendría su Sitio Web o su Página Web ?
    a.❑ WebHosting Propio b. ❑ WebHosting Con otra Empresa c. ❑ WebHosting Gratis
    d. ❑ Otro _____

17. ¿ Que tipo de Servidor Web Utilizaría (si elige WebHosting Propio)?
    a. ❑ Código Libre (Ej. Apache, etc) b. ❑ Bajo Licencia (Ej. IIS, etc)
    c. ❑ Otro _____

ENCUESTA USUARIO ADMINISTRATIVO

La presente encuesta tiene como objetivo recolectar información confiable que permita analizar el impacto de Internet y los beneficios que ofrece para la Mediana Empresa en El Salvador, así como determinar el tipo de Tecnología que utilizan para mantenimiento de su Sitio y sus capacidades de inversión. La información que proporcione será de mucha importancia para realizar un proyecto relacionado con el tema, por lo que le agradeceríamos responder en forma objetiva las preguntas que se le formulen. Por su colaboración, Gracias.

1. ¿ En su Empresa tienen conexión a Internet ? Si su respuesta es sí continué, si es No, ir a la pregunta 4
   a. ❑ Sí b. ❑ No

2. ¿ Cuál es el fin principal de acceso a Internet en su empresa? (marque todas las que apliquen)
   a. ❑ Correo Electrónico b. ❑ Labores Educativas c. ❑ Compras d. ❑ Entretenimiento
   e. ❑ Trabajo f. ❑ Noticias g. ❑ Búsqueda de información h. ❑ Pasatiempo
   i. ❑ Chat j. ❑ Otro_____

3. ¿ Que tipo de información busca regularmente en Internet su empresa ? (marque todas las que apliquen)
   a. ❑ Noticias y Actualidad b.❑ Compras c.❑ Deportes d.❑ Cultura y Arte
   e. ❑ Cine f. ❑ Búsqueda de Empleo g.❑ Música h.❑ Juegos
   i. ❑ Política j. ❑ Eventos k.❑ Educación l. ❑ Ciencia y Tecnología
   m.❑ Economía y Finanzas n.❑ Computación o.❑ Clasificados p.❑ Comercio Electrónico
   q. ❑Otras r. ❑ Ninguna

4. ¿ Su empresa Utiliza la Publicidad en Internet, a través de banners u otros métodos, en algún sitio de Internet?
   a. ❑ Sí b. ❑ No

5. ¿ Posee su empresa Pagina Web o Sitio Web? (si su respuesta es no, pase a la pregunta 9)
   a. ❑ Sí b. ❑ No

6. ¿Por qué razón su empresa tiene presencia en Internet? (marque todas las que apliquen)
   a. ❑ Imagen b.❑ Información sobre productos/servicios c. ❑ Publicidad
   d. ❑ Servicio al Cliente e.❑ Ventas En Línea f. ❑ Comunicación Interna
   g. ❑ Otro _____

7. ¿ Como mantiene su Sitio Web o su Página Web (Si posee) ?
   _____
   _____

8. ¿La forma en que mantiene su sitio cumple con sus expectativas? ¿ Por qué razón?
   a. ❑ Sí b. ❑ No
   _____
   _____

9. ¿Estaría interesado en tener presencia en Internet (si no posee Sitio Web)? ¿Por qué razón?
   a. ❑ Sí b. ❑ No
   _____
   _____

10. ¿ Como mantendría su Sitio Web o su Página Web (Tecnología bajo licencia, Servicios con otra empresa, Tecnología Gratuita)?
    _____
    _____

ANEXO III

LISTADO DE EMPRESAS ENCUESTADAS

1. 3M DE EL SALVADOR
2. ACES
3. ACSA
4. AGAVE, S.A. DE C.V.
5. AGDOSA
6. AGELSA
7. ALCATEL DE EL SALVADOR, S.A. DE C.V.
8. AVIPRO
9. AVISAL
10. AYRE DE EL SALVADOR
11. AZBA S.A. DE C.V.
12. CARVAJAL, S.A.
13. CESSA
14. CODIFARMA
15. COLEGIO LA ASUNCION
16. COMPRAMERICA, S.A.
17. COPRESA
18. CORMAR
19. CORPORACION 2 + 1
20. CORSETUR
21. CRIAVES
22. COMPUTER DATA SYSTEM
23. DIBARSA
24. DISTRIBUIDORA EUROPEA
25. DIZAC
26. DNC
27. DROGUERIA BETAFARMA
28. DROGUERIA SUIZA
29. DYMEL
30. ELECTRO GLOBAL
31. FARMACEUTICA RODIN
32. GAMMA SISTEMA FRIO
33. GMT
34. GRAFICOS E IMPRESOS
35. GRUPO IMPRECEM
36. GRUPO SANTILLANA
37. HYSTIK DE EL SALVADOR
38. ICAT
39. IFASAL
40. IMAGEN
41. INDUSTRIAS ORION
42. INLASA
43. INMOBILIARIA MICASA
44. INQUISALVA

45. INTERCOM
46. JARDIN BONTANICO
47. KANUX, S.A. HIELO GLACIAL
48. KRAFT DE EL SALVADOR
49. LA CADENA, S.A.
50. LA CENTROAMERICANA, S.A.
51. LABORATORIOS CAROSA
52. LABORATORIOS MORAZAN
53. LISTONES FANTASYA
54. LOPEZ Y LOPEZ, S.A.
55. MONOLIT
56. MONTAJES DE EL SALVADOR
57. MUDANZAS INTERNACIONALES
58. NHA, INGENIEROS, S.A.
59. PERFECT STTAFING
60. PERGAMINOS
61. PHELPS DODGE
62. RT, S.A. DE C.V.
63. SABESA
64. SABRE, S.A.
65. SAMARITAN'S PURSE
66. SANCHEZ CENTROAMERICA
67. SEASSA
68. SSASE
69. SUPER MEDIA DE EL SALVADOR
70. SYLVANIA, S.A.
71. TECHNO SCREEN
72. TELEVIP
73. TERRASINA
74. TEXTILES Y DERIVADOS
75. UNDISA
76. UNIFERSA
77. UNIPHARM
78. UNSSA
79. WARNER LAMBERT
80. WESTERN UNION
81. WORD EXCHANGE

UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERIA

ESCUELA DE INGENIERIA EN COMPUTACION

MANUAL DEL USUARIO

“SERVIDOR WEB 1.0”

CIUDADELA DON BOSCO DICIEMBRE 2003

NDICE

1. DESCRIPCIÓN DEL SERVIDOR WEB

Un Servidor Web es un programa que funciona sobre una máquina en red, esperando conexiones de otras maquinas llamadas clientes, para servir documentos solicitados desde un navegador, los cuales son almacenados en sus dispositivos de almacenamiento masivo o discos duros. El Servidor Web gestiona y administra el acceso a dichos documentos por parte de otros servidores u otros clientes.

El Servidor Web 1.0 es un software para servir paginas en Internet, escrito en el lenguaje de programación Perl. Esta hecho bajo el esquema de la programación mediante el uso de módulos, lo que hace su expansión o adición de nuevas funcionalidades mucho más fácil y eficiente.

2. REQUERIMIENTOS DE HARDWARE Y SOFTWARE

2.1 Requerimientos Mínimos de Hardware

El software del Servidor Web trabaja sobre la plataforma Linux, en su versión Red Hat 8.0 Servidor, por lo que los requisitos mínimos de hardware, son los requerimientos mínimos para una instalación de Linux Red Hat 8.0, tipo Servidor.

- Una computadora con las siguientes características:

CPU:

Mínimo: Tipo Pentium.

Recomendado: 200 Megaherzios de tipo Pentium o superior.

Espacio en disco duro:

Mínimo: 650MB.

Recomendado: 2.5GB.

Instalación completa: 4.5GB.

(Se requiere espacio adicional para el almacenamiento de ficheros.)

Memoria RAM:

Mínimo para el modo texto: 32MB.

Mínimo para el modo gráfico: 128MB.

Recomendado para el modo gráfico: 192MB.

- Conexión a Internet por medio de una Línea dedicada.
- UPS o fuente de poder con puerto serial y soporte para conectar a una computadora (no es indispensable).

2.2 Requerimientos Mínimos de Software

- Sistema Operativo Linux, versión Red Hat 8.0, tipo Servidor.
- Perl 5.0 o mayor
- Software del Servidor Web 1.0
- Software NUT: Network UPS Tools (no es indispensable)

3. INSTALACIÓN Y CONFIGURACIÓN

3.1 Programa de Instalación y Configuración

La instalación del Servidor Web 1.0 se realiza a través del srcipt llamado *setup.pl*, que viene con el paquete de archivos del software. Los pasos para realizar la instalación son los siguientes.

1. Inserte el CD con los archivos fuente.
2. Copie el directorio swp y su contenido al directorio /root

3. Abra o inicie una sesión en una consola o ventana de comando. Para abrir una ventana de comando, de un click en el botón marcado con un recuadro en la Figura N° 1.

Figura N° 1. Barra de Herramientas de Red Hat 8.0.

La ventana de comando es como se muestra en la Figura N° 2.

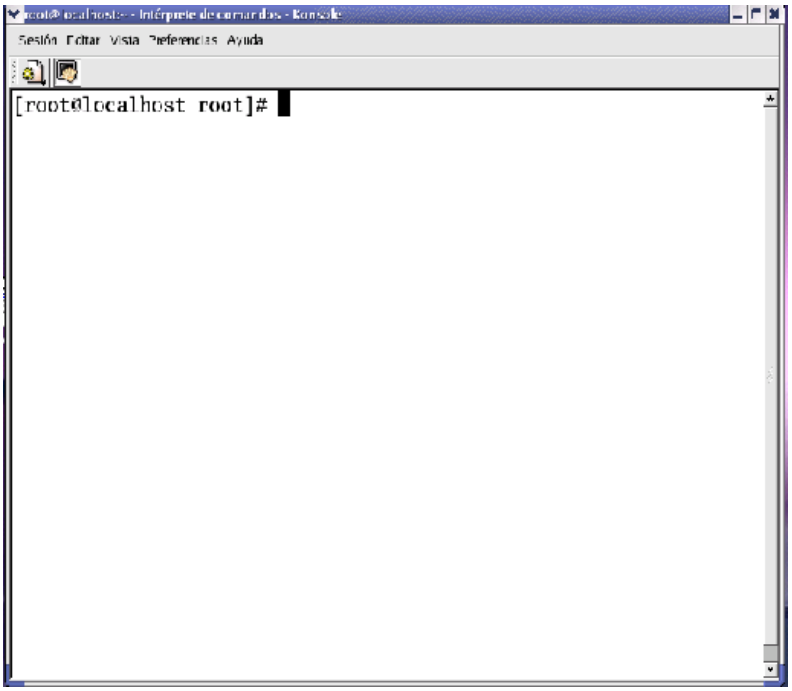


Figura N° 2. Terminal o Ventana de Consola de Red Hat 8.0.

4. Cámbiese a la ruta o al directorio /usr/bin (Teclee cd /usr/bin)

   [root@localhost root]# cd /usr/bin

5. Teclee lo siguiente: perl /root/swp/swp/setup.pl, después del prompt:

   [root@localhost bin]# perl /root/swp/swp/setup.pl

6. A continuación aparecerán las siguientes instrucciones en la ventana de consola:

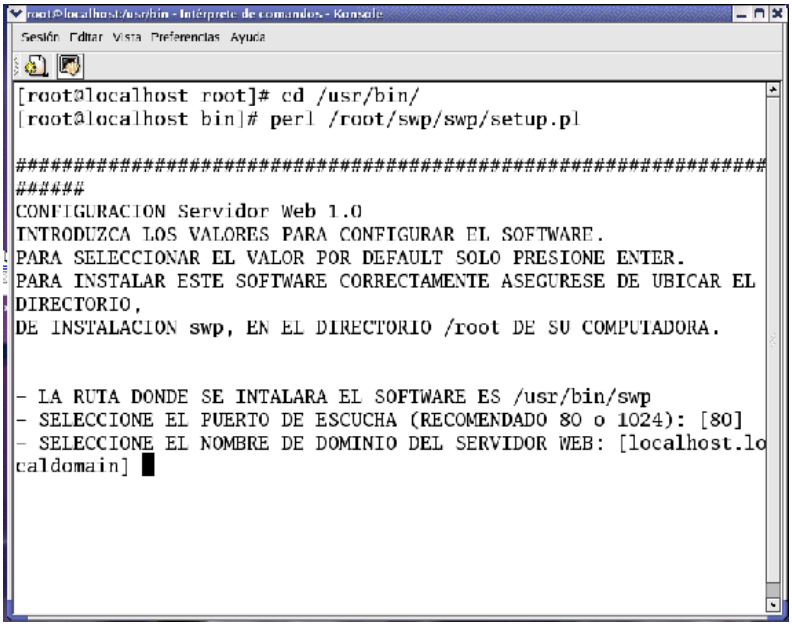


Figura N° 3. Proceso de Instalación, solicitud de parámetros.

El texto que aparece se lee de la siguiente manera:

##
CONFIGURACION Servidor Web 1.0
INTRODUZCA LOS VALORES PARA CONFIGURAR EL SOFTWARE.
PARA SELECCIONAR EL VALOR POR DEFAULT SOLO PRESIONES ENTER.
PARA INSTALAR ESTE SOFTWARE CORRECTAMENTE ASEGURESE DE UBICAR EL DIRECTORIO,
DE INSTALACION swp, EN EL DIRECTORIO /ROOT DE SU COMPUTADORA.

LA RUTA DONDE SE INTALARA EL SOFTWARE ES /usr/bin/swp
SELECCIONE EL PUERTO DE ESCUCHA (RECOMENDADO 80 o 1024): [80]
SELECCIONE EL NOMBRE DE DOMINIO DEL SERVIDOR WEB: [localhost.localdomain]

Se le solicitaran 3 parámetros. Para aceptar el valor por default o guardar los cambios, simplemente presione la tecla Enter.

- La ruta donde se instala el software: este valor no se puede cambiar, es estándar, esta es la ruta donde el software será instalado.

- El puerto de escucha: el numero del puerto donde el Servidor Web esperara las conexiones entrantes. El valor por default es el puerto 80, sin embargo es recomendable que si se va a cambiar este valor únicamente, se cambie por el valor del puerto 1024. Estos valores son los estándares para puertos de escucha de Servidores Web.
- El nombre de domino: El identificador de la computadora servidor en que residirá el software. El valor por default es identificado automáticamente, gracias a la configuración del sistema operativo.

7. Una vez introducidos los valores solicitados, se procede a crear los directorios necesarios. Si ocurriera un error al crear un directorio, aparecería un mensaje indicando el error y el nombre del directorio sobre el que ocurrió el error. De lo contrario, el proceso se realiza satisfactoriamente.

```
CREANDO DIRECTORIOS
```

8. A continuación de la creación de los directorios, se procede a la copia de los archivos necesarios para el funcionamiento del software. Si no ocurre ningún error en la copia de los archivos en la pantalla aparecerá lo siguiente:

```
COPIANDO LOS ARCHIVOS NECESARIOS
COPIANDO /root/swp/swp/setup.pl => /usr/bin/swp/
COPIANDO /root/swp/swp/sep.pl => /usr/bin/swp/
COPIANDO /root/swp/swp/mime.types => /usr/bin/swp/
COPIANDO /root/swp/swp/Readme.txt => /usr/bin/swp/
COPIANDO /root/swp/swp/server.cfg => /usr/bin/swp/
COPIANDO /root/swp/swp/temp.txt => /usr/bin/swp/
COPIANDO /root/swp/swp/usuarios.txt => /usr/bin/swp/
COPIANDO /root/swp/swp/pas.txt => /usr/bin/swp/
COPIANDO /root/swp/swp/html/index.html => /usr/bin/swp/html
COPIANDO /root/swp/swp/html/imagenes/logob.jpg => /usr/bin/swp/html/imagenes
COPIANDO /root/swp/swp/lib/swp/Config.pm => /usr/bin/swp/lib/swp
COPIANDO /root/swp/swp/lib/swp/Daemon.pm => /usr/bin/swp/lib/swp
COPIANDO /root/swp/swp/lib/swp/Logs.pm => /usr/bin/swp/lib/swp
COPIANDO /root/swp/swp/lib/swp/Server2.pm => /usr/bin/swp/lib/swp
COPIANDO /root/swp/swp/lib/swp/Modulos.pm => /usr/bin/swp/lib/swp
COPIANDO /root/swp/swp/lib/swp/Server.pm => /usr/bin/swp/lib/swp
```

COPIANDO /root/swp/swp/lib/swp/modulos/mod_cgi.pm => /usr/bin/swp/lib/swp/modulos

Estos son los archivos que se copian en el proceso de instalación. La ruta que sigue a la palabra COPIANDO es la ruta o ubicación de donde se copia el archivo. La ruta después de los caracteres => indica la ubicación hacia donde se copio el archivo. Al igual que con los directorios, si ocurriese un error al realizar la copia, aparecería el mensaje con el error ocurrido y el nombre del archivo respectivo.

9. A continuación de la copia de los archivos, el programa procede a realizar la configuración del Servidor Web. En esta parte es donde se asocian los parámetros introducidos por el usuario con los parámetros establecidos en el archivo de configuración server.cfg. Si por algún motivo no el programa no pudiera actualizar el archivo, aparecería un mensaje de error indicando que ha ocurrido un error y que debe editar dicho archivo manualmente. Si desea conocer la configuración de su Servidor Web, puede abril dicho archivo, aunque es aconsejable no realizar ningún tipo de modificación en dicho archivo, ya que de esta forma no se garantiza el correcto funcionamiento del servidor. El mensaje que aparecerá es el siguiente:

   CONFIGURANDO /usr/bin/swp/server.cfg

Lo que se visualizaría en la ventana de consola se presenta a continuación.

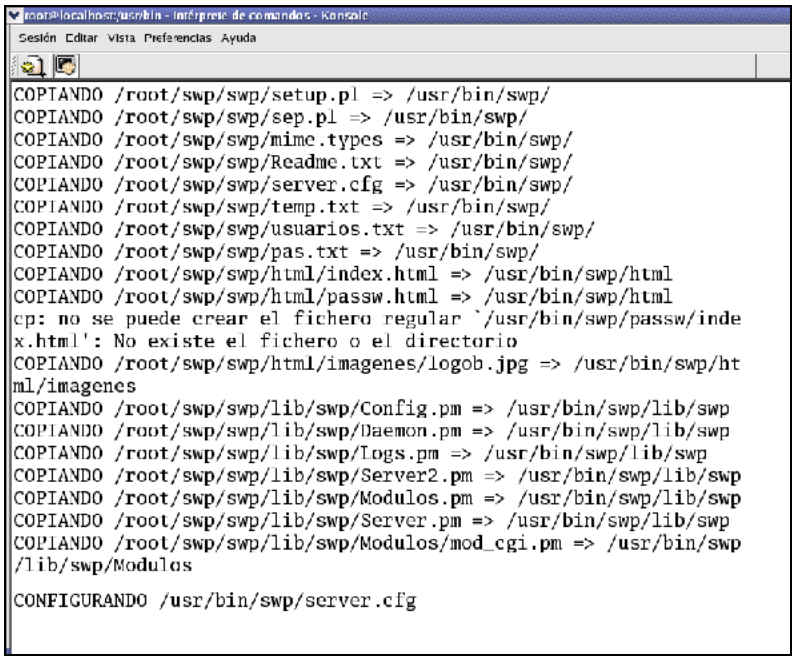


Figura N° 4. Proceso de Instalación, copia y configuración.

10. Finalmente, copie el directorio Interfaz al directorio raíz de documentos, es decir al directorio *html* en la ruta */usr/bin/swp/html*. En este momento, el servidor esta instalado y configurado, de este modo se puede proceder a operarlo.

3.2 Comprobación de la Instalación

Una vez haya realizado satisfactoriamente la instalación del software del Servidor Web, puede comprobar si este esta funcionando correctamente. Siga los siguientes pasos:

1. Inicie el Servidor Web 1.0 (consulte la sección 4.1).
2. Abra una ventana de navegador, por ejemplo, el navegador Mozilla. El icono del navegador Mozilla es el marcado con un recuadro en la Figura N° 5.

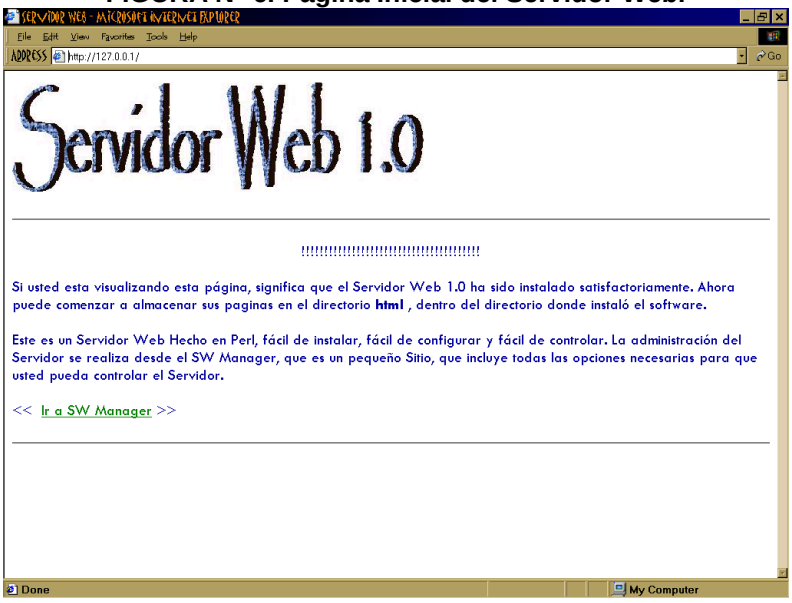
Figura N° 5. Icono del Navegador Web Mozilla.

3. En la barra de direcciones, en el campo *location* digite la siguiente dirección y presione Enter:

   http://127.0.0.1

Si el Software esta instalado correctamente, debe aparecer en su navegador la pagina de Inicio del Servidor Web, esta se muestra en la Figura N° 6.

FIGURA N° 6. Página Inicial del Servidor Web.



4. Si no logra visualizar esa pagina quiere decir que existió algún error durante la instalación.

4. CONTROL DEL SERVIDOR WEB

4.1 Operaciones Básicas

Existen 3 operaciones básicas para el control del funcionamiento del Servidor Web 1.0; inicio, finalización y reinicio. Estas se ejecutan desde una ventana de consola o terminal, similar a la ventana de comando en Windows. A continuación se describe cada operación.

4.1.1 Inicio

Para iniciar el Servidor Web siga los siguientes pasos:

1. Inicie una sesión en una ventana de consola.
2. Cámbiese al directorio /usr/bin, para esto teclee lo siguiente, después del prompt:

```
[root@localhost root]# cd /usr/bin
```

3. A continuación, teclee lo siguiente después del prompt para iniciar el Servidor Web:

```
[root@localhost bin]# perl /usr/bin/swp/sep.pl start
```

donde:

[root@localhost bin]#	*es el promt de la consola, similar a C:/ de Windows*
perl	*es el comando para ejecutar un script o programa de Perl*
/usr/bin/swp/sep.pl	*la ruta completa donde reside el script a ejecutar*
start	*el parámetro pasado al script que indica al servidor que inicie*

4. Presione la tecla Enter, y aparecerá en la ventana de consola el siguiente mensaje:

```
...Iniciando Servidor Web 1.0
```

En la Figura N° 7 se observa lo que aparece en la consola.

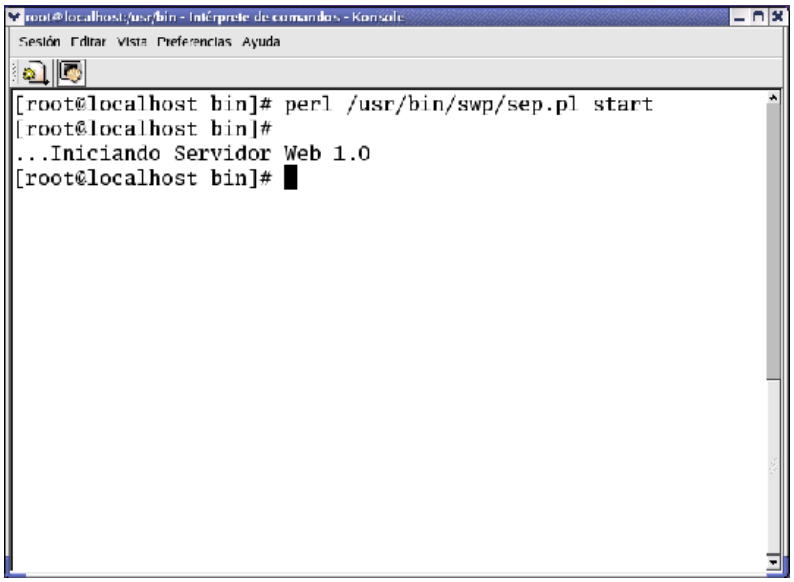


Figura N° 7. Pantalla de Inicio del Servidor Web 1.0.

5. Esto indica que el Servidor Web esta listo para recibir conexiones entrantes. Si cierra la ventana de consola, el proceso que ejecuta el Servidor Web será eliminado.

4.1.2 Finalización

Para detener o finalizar el funcionamiento del Servidor Web siga los siguientes pasos:

1. Active la sesión de la ventana de consola, donde inicio el Servidor Web.

2. Teclee lo siguiente después del prompt para detener el Servidor Web:

   [root@localhost bin]# perl /usr/bin/swp/sep.pl stop

   donde:

[root@localhost bin]#	*es el promt de la consola, similar a C:/ de Windows*
perl	*es el comando para ejecutar un script o programa de Perl*
/usr/bin/swp/sep.pl	*la ruta completa donde reside el script a ejecutar*
stop	*el parámetro pasado al script que indica al servidor finalizar*

3. Presione la tecla Enter, y aparecerá en la ventana de consola el siguiente mensaje:

    Proceso Eliminado 1093

Donde el numero 1093 (para ejemplificar) representa el pid (Process ID) o el id del proceso en el sistema operativo. De este modo el Servidor Web deja de funcionar. Esto indica que el Servidor Web esta detenido. En la Figura N° 8 se observa lo que aparece en la consola.

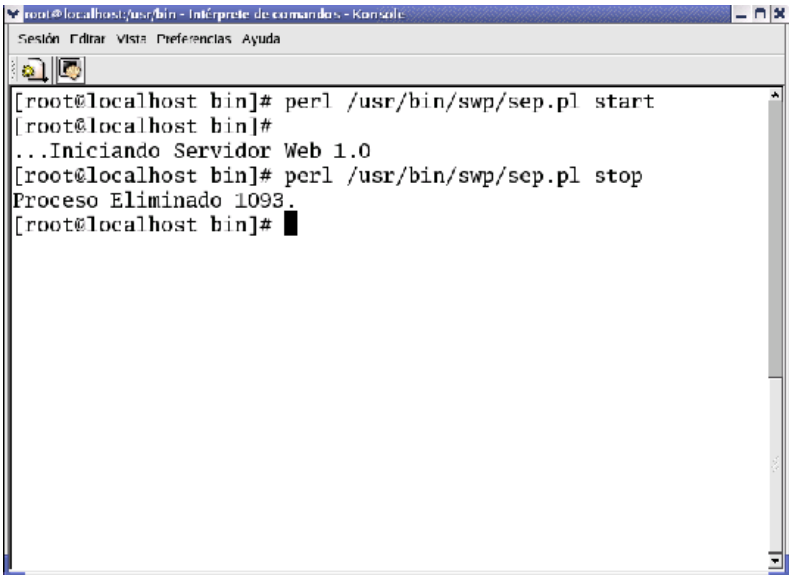


Figura N° 8. Pantalla de Finalización del Servidor Web 1.0.

4.1.3 Reinicio

Para reiniciar el funcionamiento del Servidor Web siga los siguientes pasos:

1. Active la sesión de la ventana de consola, donde inicio el Servidor Web.

2. Teclee lo siguiente después del prompt para reiniciar el Servidor Web:

    [root@localhost bin]# perl /usr/bin/swp/sep.pl restart

donde:

[root@localhost bin]#	*es el promt de la consola, similar a C:/ de Windows*
perl	*es el comando para ejecutar un script o programa de Perl*
/usr/bin/swp/sep.pl	*la ruta completa donde reside el script a ejecutar*
restart	*el parámetro pasado al script que indica al servidor reiniciar*

3. Presione la tecla Enter. Empezara a ejecutarse el proceso de finalización del servidor y a continuación empezara a ejecutarse el proceso de inicio del Servidor Web.

4.2 Modo Protección

El modo protección es aquel en el que se restringe el acceso al contenido del Servidor Web únicamente a los usuarios registrados, es decir, que poseen un nombre de usuario y un password asignados por el administrador del Servidor Web. Cuando el modo protección esta activado, cada vez que un usuario solicita una pagina almacenada en el servidor, se despliega la pagina de autenticación para que, este ingrese la información correspondiente. Para obtener información sobre la activación y desactivación del modo protección, consulte la sección 5.1.3 en la cual se describe como ejecutar estos procesos desde la interfaz grafica.

4.3 Autenticación de Usuarios

La autenticación de usuarios es el proceso en el cual, los usuarios tienen que ingresar un nombre de usuario y un password, en la pagina de autenticación del Servidor Web, cuando el modo protección esta activado. En la Figura N° 9 se muestra la Pagina de autenticación del Servidor Web 1.0

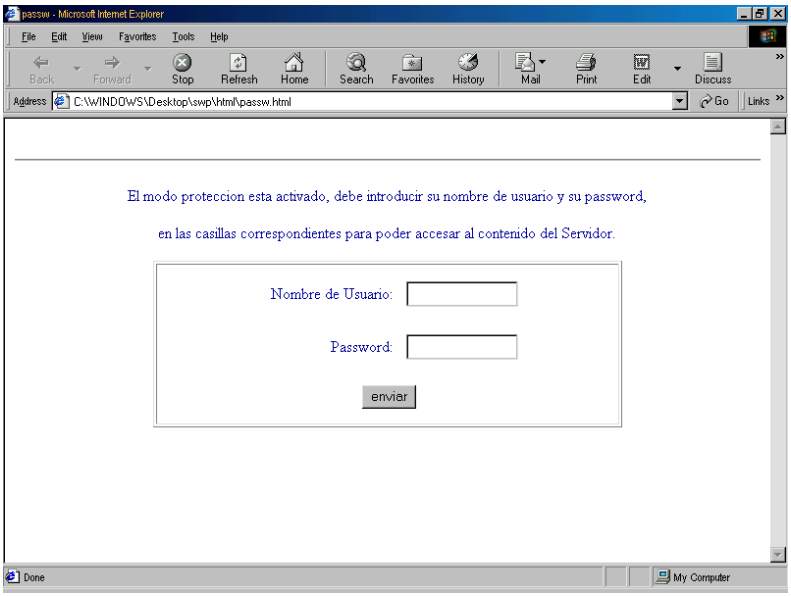


Figura N° 9. Página de Autenticación del Servidor Web 1.0.

Cuando el modo protección esta activado, todas las personas que deseen accesar al contenido del Servidor Web 1.0, únicamente deben ingresar en la casilla *Nombre de Usuario*, su Usuario; en la casilla *Password*, su contraseña; y a continuacion presionar el boton *enviar*. De esta manera, el Servidor Web compara si la información proporcionada es correcta. Si fue así, la pagina solicitada, aparecerá en el navegador. De lo contrario, la pagina que aparcera, será la pagina de autenticación. Hasta que no se compruebe que los datos están correctos la pagina de autenticación continuara apareciendo en el navegador.

Para poder ser un usuario registrado, se debe comunicar al administrador del Servidor Web 1.0, para el ingreso de la información correspondiente. Refiérase a la sección 5.1.2.1, en la que se describe el proceso para registrar y adicionar usuarios en el Servidor Web.

4.4 Almacenar Archivos en el Servidor

Entre los archivos fuente del Servidor Web 1.0, existe el directorio o carpeta html. Este es el directorio raíz de documentos del servidor. En esta carpeta es donde deben almacenarse todos los archivos, documentos o paginas web para que el servidor pueda accesar a ellos cuando sean solicitados.

4.5 Agregar Modulos

El Servidor Web 1.0 esta realizado bajo el esquema modular. Es decir las funciones que el servidor realiza, las hace a través de módulos, que son archivos de Perl de extensión .pm (de perl module). Actualmente el único modulo que forma parte del servidor es el modulo que implementa el manejo de CGI. Si se quiere agregar nuevas funcionalidades o soporte para bases de datos o lenguajes de programación al servidor, se deben seguir los siguientes pasos:

1. Copiar el modulo en el directorio módulos
2. editar el archivo de configuración server.cfg en parte de módulos básicos:

   AddModule mod_nombremodulo

3. editar el archivo de configuración server.cfg en parte de manejadores de archivos:

   AddType tipo_archivo extensión

4. Reiniciar el Servidor Web

4.5 Archivos de Información

Existen dentro del directorio de archivos fuente del Servidor Web, dos archivos de utilidad para el administrador del Servidor Web, ya que le proporcionan información sobre los accesos por parte de los clientes, y los errores en la

atención de peticiones del Servidor Web. Por otra parte, se encuentra el archivo en el que se encuentra registrada toda la información de configuración del Servidor Web. Estos archivos se describen a continuación.

- *access.log:* este es un archivo en texto plano ubicado en la ruta */usr/bin/swp/log*. En el se registra toda la información de acceso al Servidor Web tal como, la fecha y hora del acceso, el tipo de petición (GET o POST) y el objeto solicitado (nombre de archivo o pagina web).

- *error.log*: archivo de texto plano ubicado en la ruta */usr/bin/swp/log*. En el se registra la información sobre los errores que ocurren durante el proceso de inicio y atención de peticiones del Servidor Web. La información que se registra es el error, el script y la línea del mismo donde ocurre el error.

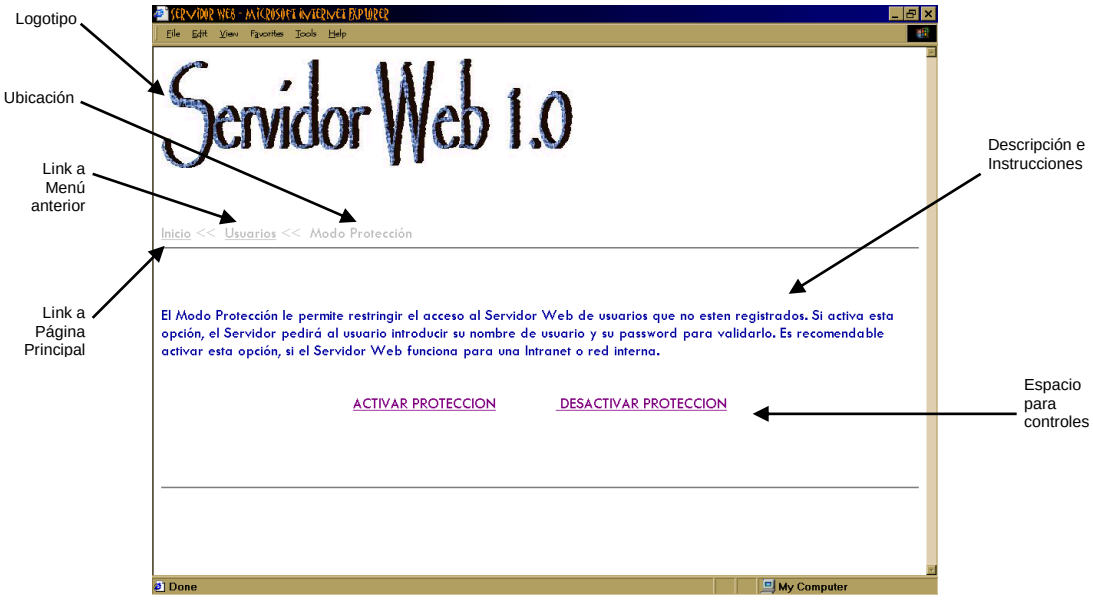
- *server.cfg*: archivo de texto plano ubicado en la ruta */usr/bin/swp*. En el se registra la información sobre la configuración actual del Servidor Web. La ruta de instalación, el puerto de escucha, la ruta del directorio raíz de documentos, entre otra, es la información que se registra en dicho archivo. Es recomendable no cambiar los valores de los parámetros que ya están establecidos en este archivo.

5. ADMINISTRACIÓN DE USUARIOS Y CONTROL DE MODO PROTECCIÓN

Las opciones de administración de usuarios y control del modo de protección, planteadas en los requerimientos de la interfaz del Servidor Web, están incluidas y se ejecutan desde un Sitio. Dicho sitio consta de una página principal que contiene todas las opciones del Sitio y desde donde se cargan las páginas de ejecución de las operaciones.

En cada página se describe la función que se realiza y contiene los controles necesarios para realizar la operación. Todas las páginas incluyen el logotipo del Servidor Web y poseen un link hacia la página de presentación. En caso de ser un sub menú, un link hacia la opción principal de la que se derivan. De igual manera se indica la ubicación dentro del sitio. En la Figura N° 10 se describe el formato general de las páginas del Sitio Web.

FIGURA N° 10. Formato General de las Páginas Web.



El Sitio Web esta formado por archivos de extensión html, y no necesita más que un navegador para visualizarse. Para ingresar, el usuario debe ejecutar la página de presentación, desde la cual tendrá acceso a todas las opciones disponibles. El sitio se ubica en el directorio Interfaz dentro del directorio raíz de documentos del Servidor Web, *html*, ya que dicho sitio ejecuta sus procesos, a través del Servidor Web, es decir para ejecutar el sitio, el Servidor Web debe estar iniciado.

5.1 Descripción del Sitio Web

5.1.1. Presentación del Sitio

Esta es la página principal del Sitio Web que incorpora una breve descripción del software y los requerimientos del mismo. La página contiene las opciones disponibles para la administración y operación del Servidor Web. En la Figura N°11 se muestra la página de presentación del Sitio Web administrador.

FIGURA N° 11. Página de Presentación del Sitio Web.

Las opciones que la página de presentación contiene son las siguientes:

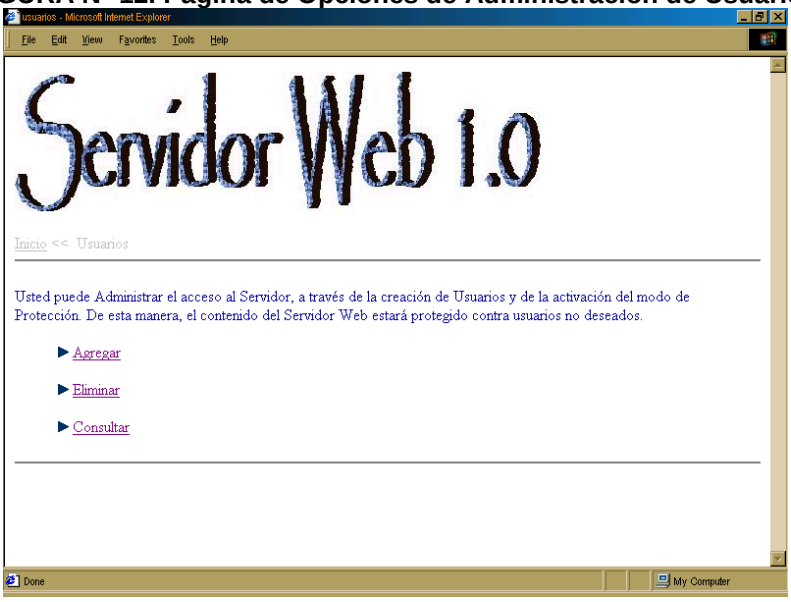
- Usuarios: Administra los Usuarios del Servidor Web y activa el Modo de Protección.
- Modo Protección: Activa y desactiva el Modo de Protección del Servidor Web.

5.1.2. Usuarios

Esta opción se incluye para gestionar la administración y el control de los usuarios del Servidor Web. A través de ella se pueden adicionar, eliminar y consultar los

usuarios registrados para accesar al contenido del Servidor Web, cuando el modo de protección esta activado. En la Figura N° 12 se observa la página de Usuarios.

FIGURA N° 12. Página de Opciones de Administración de Usuarios



Esta página incluye un sub menú con las opciones disponibles para administrar los usuarios. A continuación se describe cada una de ellas.

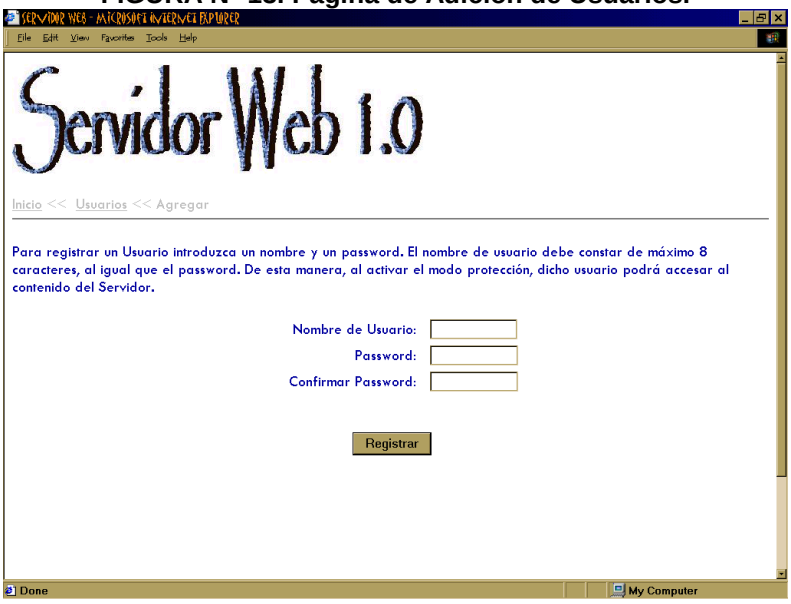
5.1.2.1 Adición

Con esta opción se pueden registrar usuarios del Servidor Web. La información que se solicita es la siguiente:

- Nombre de Usuario: Cadena de 8 caracteres máximo que identifica al usuario.
- Password: Identificador de 8 caracteres que validará al usuario cuando accede al servidor.
- Confirmar Password: Se utiliza para asegurarse de que la cadena que se introdujo en password sea la correcta.

En la Figura N° 13 se observa la página de adición de usuarios. Al presionar el botón Registrar, la información introducida en los campos se envía al programa que se encarga de realizar este proceso.

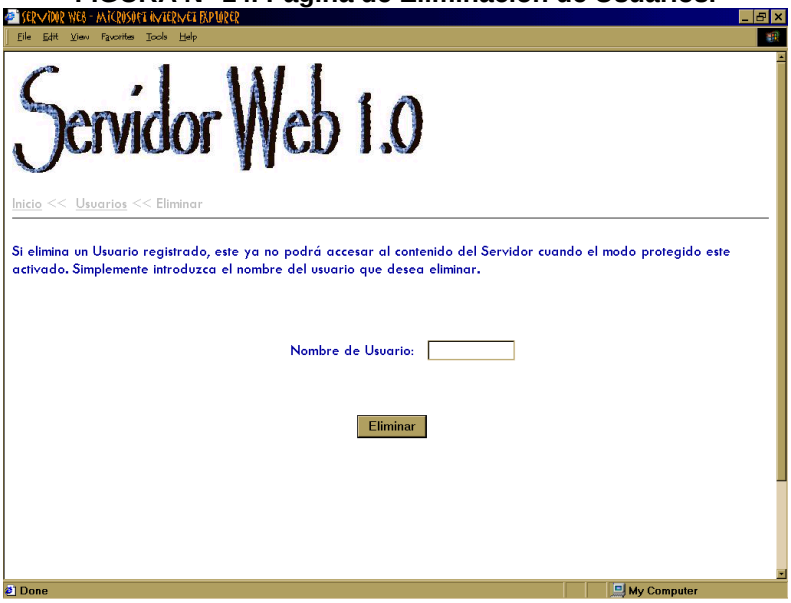
FIGURA N° 13. Página de Adición de Usuarios.



5.1.2.2 Eliminación

La eliminación de usuarios registrados se realiza simplemente introduciendo en la caja de texto, el nombre del usuario que se desea eliminar. Al presionar el botón Eliminar, se invoca el programa que ejecuta este proceso. En la Figura N° 14 se muestra la página de eliminación.

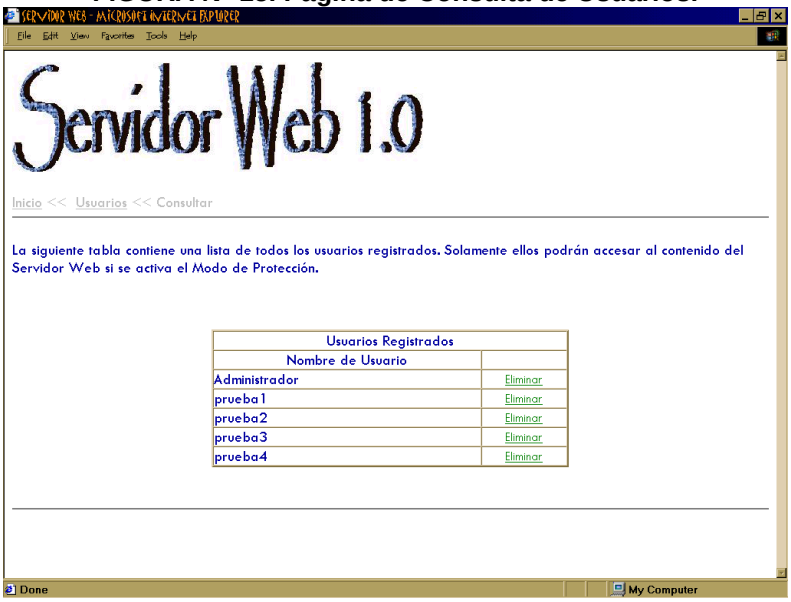
FIGURA N° 14. Página de Eliminación de Usuarios.



5.1.2.3 Consultar

A través de esta opción se pueden visualizar en una tabla, todos los usuarios registrados que pueden accesar al contenido del Servidor Web cuando el modo de protección esta activado. En la Figura N° 15 se observa la página de consulta.

FIGURA N° 15. Página de Consulta de Usuarios.

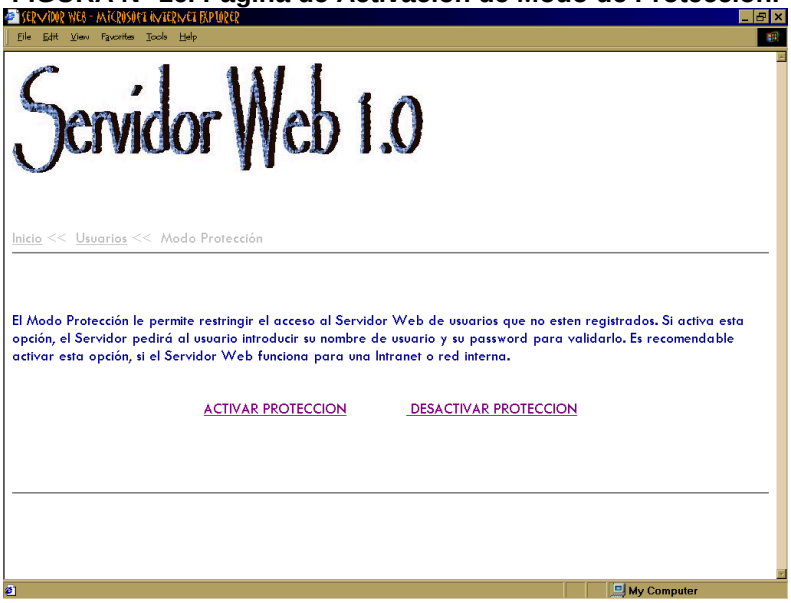


Como muestra la figura, la tabla posee dos columnas; una donde se muestra el nombre del usuario y la otra que incluye un link hacia la página de eliminación de usuarios.

5.1.3. Modo Protección

En esta página se define la protección contra usuarios no deseados o prohibidos, es decir, que se restringe el acceso al contenido del Servidor Web únicamente para los usuarios que estén registrados en el archivo correspondiente. La información contenida en dicha página se muestra en la Figura N° 16.

FIGURA N° 16. Página de Activación de Modo de Protección.



Cuando se elige el link ACTIVAR PROTECCION, se ejecuta el programa que activa el Modo de Protección del Servidor Web. De esta manera sólo los usuarios que posean un nombre de usuario y un password registrado podrán accesar al contenido del mismo.

Por otra parte, cuando se escoge el link DESACTIVAR PROTECCION, el Modo Protección es desactivado y cualquier usuario puede accesar al contenido del Servidor Web.

6. CONFIGURACIÓN DEL FIREWALL

A continuación se presenta un guía para la implementación de un firewall a través del uso de las herramientas que el Sistema Operativo Linux Red Hat 8.0, proporciona.

Un firewall o cortafuegos evita que los virus invadan una computadora y evita que los usuarios no autorizados accedan a un Servidor. El firewall está ubicado entre la computadora y la red. Determina los servicios a los que pueden acceder los usuarios remotos en la red. Un firewall que haya sido configurado debidamente puede aumentar la seguridad de un sistema. Se recomienda configurar un firewall para cualquier sistema con una conexión de Internet.

6.1 Firewalls e iptables

Linux contiene utilidades avanzadas para filtrado de paquetes, el proceso de controlar los paquetes de red cuando entran, se mueven o salen de un sistema dentro del kernel. Los kernels anteriores al 2.4 trabajaban con *ipchains* para efectuar el filtrado de paquetes y usaban listas de reglas que se aplicaban a los paquetes en cada paso del proceso de filtrado. La presentación del kernel 2.4 trajo consigo iptables (también llamado *netfilter*), que es parecido a ipchains pero con mejoras en el funcionamiento y en el control disponible a la hora de filtrar paquetes. De esta manera se aconseja el uso de iptables para implementar el firewall.

6.2 Filtrado de paquetes

El tráfico se mueve a través de una red en paquetes. Un paquete de red es una colección de datos en diferentes tamaños y formatos. Para enviar un fichero por red, la computadora emisor debe en primer lugar partirlo en diferentes paquetes usando las reglas del protocolo de red. Cada uno de estos paquetes contiene una parte pequeña de los datos del fichero. Cuando recibe la transmisión, la computadora receptor reensambla los paquetes y construye de nuevo el fichero el fichero.

Cada paquete contiene información que le ayuda a navegar por la red y moverse hacia su destino. El kernel de Linux contiene la característica interna de filtrado de paquetes, que le permite aceptar algunos de ellos en el sistema mientras que intercepta y para a otros. El filtro de red kernel 2.4 tiene tres *tablas* internas o *listas de reglas*. Son las siguientes:

- *filtro*: Esta es la tabla por defecto para manejar paquetes de red.
- *nat:* Esta tabla se usa para alterar paquetes que crean una nueva conexión.
- *mangle:* Esta tabla se usa tipos específicos de alteración de paquetes.

Cada una de estas tablas tiene un grupo de *cadenas* internas que corresponden a las acciones llevadas a cabo por el filtro de red en el paquete. Las cadenas internas para la tabla filtro son las siguientes:

- *INPUT:* Esta cadena sirve solo para paquetes recibidos por medio de una interfaz de red.
- *OUTPUT:* Esta cadena sirve para paquetes enviados por medio de la misma interfaz de red que recibió los paquetes.
- *FORWARD:* Esta cadena sirve para paquetes recibidos en una interfaz de red y enviados en otra.

Las cadenas internas para la tabla *nat* son las siguientes:

- *PREROUTING:* Esta cadena altera paquetes recibidos por medio de una interfaz de red cuando llegan.
- *OUTPUT*: Esta cadena altera paquetes generados localmente antes de que sean dirigidos por medio de una interfaz de red.
- *POSTROUTING:* Esta cadena altera paquetes antes de que sean enviados por medio de una interfaz de red.

Las cadenas internas para la tabla *mangle* son las siguientes:

- *PREROUTING:* Esta cadena altera paquetes recibidos por medio de una interfaz de red antes de que sean dirigidos.
- *OUTPUT*: Esta cadena altera paquetes generados localmente antes de que sean dirigidos por medio de una interfaz de red.

Cada paquete de red recibido o enviado de un sistema Linux está sujeto a al menos una tabla. Un paquete puede que sea verificado contra muchas, muchas reglas dentro de la lista de reglas antes de llegar al final de una cadena. La estructura y propósito de estas reglas puede variar, pero normalmente buscan identificar un paquete que viene de o se dirige a una dirección IP en particular o un conjunto de direcciones al usar un determinado protocolo y servicio de red. Independientemente de su destino, cuando un paquete cumple una regla en particular en una de las tablas, se asignan a un *objetivo (target)* particular, o una acción a aplicárseles. Si la regla especifica un objetivo ACCEPT para un paquete que la cumpla, el paquete se salta el resto de las verificaciones de la regla y se permite que continúe hacia su destino. Si una regla especifica un objetivo DROP, el paquete *se deja caer*, significando esto que no se permite que el paquete acceda al sistema y no se envía ninguna respuesta de vuelta al servidor que envió el paquete. Si una regla especifica un objetivo REJECT, el paquete se deja caer, pero se envía un mensaje de error al emisor.

Cada cadena tiene una política ACCEPT, DROP, o REJECT sobre el paquete, o si no, puede enviarlo al espacio de usuario con QUEUE. Si ninguna de las reglas de la cadena se aplican al paquete, entonces el paquete se trata de acuerdo a la política por defecto de las cadenas. El comando *iptables* permite configurar estas listas de reglas, así como configurar nuevas cadenas y tablas para ser usadas en si situación particular

6.3 Herramienta de configuración del Firewall

Durante la instalación de Red Hat Linux en la pantalla de configuración del firewall, como se observa en la Figura N° 17, se ha dado la posibilidad de escoger el nivel de seguridad alto, medio o ninguno o de permitir determinados dispositivos, servicios de entrada y puertos. A continuación se describen los pasos a seguir.

Figura N° 17. Configuración de Firewall.

1. Seleccione el nivel de seguridad deseado desde el menú desplegable.

- **Alto**

Si elige Alto, su sistema no aceptará conexiones (que no sean parámetros por defecto) que usted no haya definido específicamente. Por defecto, solo las siguientes conexiones están permitidas:

- Respuestas de DNS
- DHCP — de modo que cualquier interfaz de la red que use DHCP se puede configurar correctamente

Si elige Alto, su firewall no permitirá lo siguiente:

- Modo activo FTP (modo pasivo FTP, usado por defecto en la mayoría de clientes sí debería funcionar)
- transferencias de ficheros IRC DCC
- RealAudio
- Clientes remotos del sistema X Window

Si va a conectar su sistema a internet, pero no desea ejecutar un servidor, ésta es la opción más segura. Si necesita servicios adicionales, puede elegir Customize para permitir servicios específicos a través del firewall.

- **Medio**

Si elige Medio, su firewall no permitirá que máquinas remotas tengan acceso a ciertos recursos de su sistema. Por defecto, el acceso a los siguientes recursos no está permitido:

- Puertos por debajo del 1023 — los puertos reservados standard, usados por la mayoría de servicios de sistema, tales como FTP, SSH, telnet, HTTP, y NIS.

- El puerto de servidor NFS (2049) — NFS se deshabilita tanto para servidores remotos como para clientes locales.
- El modo de pantalla local del sistema X Window para clientes X remotos.
- El puerto de servidor X Font (por defecto, xfs no se escucha en la red; está deshabilitado en el servidor fuente).

Si quiere permitir recursos tales como RealAudio a la vez que bloquea el acceso a los servicios normales del sistema, elija Medio. Seleccione Customize para permitir servicios específicos a través del firewall.

- **Ningún Firewall**

Ningún firewall proporciona acceso completo a su sistema y no realiza comprobaciones de seguridad Comprobación de seguridad es la deshabilitación del acceso a ciertos servicios. Esto debería estar seleccionado únicamente si usted está conectado a una red de confianza (no Internet) o si desea hacer más configuraciones de firewall en otro momento.
Elija Customize para añadir dispositivos de confianza o para permitir servicios de entrada adicionales.

2. Seleccione los Dispositivos de confianza

Al seleccionar cualquiera de los Dispositivos de confianza se permite el acceso a su sistema a todo el tráfico de ese dispositivo; queda excluido de las reglas del firewall. Por ejemplo, si está ejecutando una red local, pero está conectado a Internet por medio de un acceso remoto PPP, puede comprobar eth0 y el tráfico proveniente de su red local será permitido. Seleccionar eth0 como de confianza significa que todo el tráfico a través de Ethernet está permitido, pero la interfaz ppp0 sigue teniendo un firewall. Si desea restringir el tráfico en una interfaz, leave it unchecked. No es recomendable que haga cualquier dispositivo conectado a redes públicas, como Internet, un Dispositivo de confianza.

4. Seleccione las opciones de los servicios a los que permitirá Entradas

Activar estas opciones permite que los servicios especificados pasen a través del firewall. Nota, durante la instalación de la estación de trabajo, la mayoría de estos servicios *no* están instalados en el sistema.

- **DHCP**

Si permite preguntas y respuestas DHCP de entrada, está permitiendo que cualquier interfaz de red que use DHCP determine sus direcciones IP. Normalmente DHCP está activado. Si DHCP no está activado, su computadora no podrá obtener una dirección IP.

- **SSH**

*S*ecure *SH*ell (SSH) es un conjunto de herramientas para conectarse y ejecutar comandos en una máquina remota. Si desea utilizar herramientas SSH para acceder a su máquina a través de un firewall, active esta opción. Para acceder a su máquina remotamente, utilizando herramientas SSH, necesita tener instalado el paquete openssh-server

- **Telnet**

Telnet es un protocolo para conectarse a máquinas remotas. Las comunicaciones Telnet son cifradas y no proporcionan seguridad ante el snooping de red. No se recomienda permitir el acceso Telnet de entrada. Si quiere permitir el acceso de retorno Telnet, tendrá que instalar el paquete *telnet-server*.

- **WWW (HTTP)**

Apache (y otros servidores Web) utilizan el protocolo HTTP para servir páginas web. Si está planeando hacer su Servidor Web accesible para todos, active esta opción. No se requiere esta opción para visualizar páginas localmente o para desarrollar páginas web. Tendrá que instalar el paquete apache si quiere servir páginas web. Al activar WWW (HTTP) no se abrirá un puerto para HTTPS. Para activar HTTPS, especifíquelo en el campo Otros puertos.

- **Mail (SMTP)**

Si quiere permitir la entrega de correo a través de su firewall, de modo que hosts remotos puedan conectarse directamente a su máquina para entregar correo, active esta opción. No necesita activarla si recoge el correo desde su servidor de ISP utilizando POP3 o IMAP, o si usa una herramienta como por ejemplo fetchmail. Tenga en cuenta que un servidor SMTP que no esté configurado adecuadamente puede permitir que máquinas remotas usen su servidor para enviar correo basura.

- **FTP**

El protocolo FTP se utiliza para transferir ficheros entre máquinas en red. Si quiere hacer su servidor FTP accesible para todos, acytive esta opción. Necesita instalar el paquete wu-ftpd (y posiblemente el anonftp) para que esta opción sea de utilidad.

- **Otros puertos**

Puede permitir el acceso a puertos que no están listados aquí, al listarlos en el campo Otros puertos. Utilice el formato siguiente: puerto:protocolo. Por ejemplo, si quiere permitir acceso IMAP a través de su firewall, puede especificar imap:tcp. También puede especificar explícitamente puertos numéricos; para permitir paquetes UDP en el puerto 1234 a través del firewall, escriba 1234:udp. Para especificar varios puertos, sepárelos con comas.

Debe tener el servicio iptables activado y Ejecutándose para activar el nivel de seguridad.

Después de la instalación, puede cambiar el nivel de seguridad de su sistema mediante el uso de la Herramienta de configuración del nivel de seguridad. Para iniciar la aplicación, seleccione Menú principal (en el panel) => Configuración del sistema => Seguridad o escriba el comando redhat-config-securitylevel desde un indicador de comandos de shell o ventana de consola. A continuación siga los

pasos detallados anteriormente. En la Figura N° 18, se muestra la pantalla de la Herramienta de configuración del nivel de seguridad.

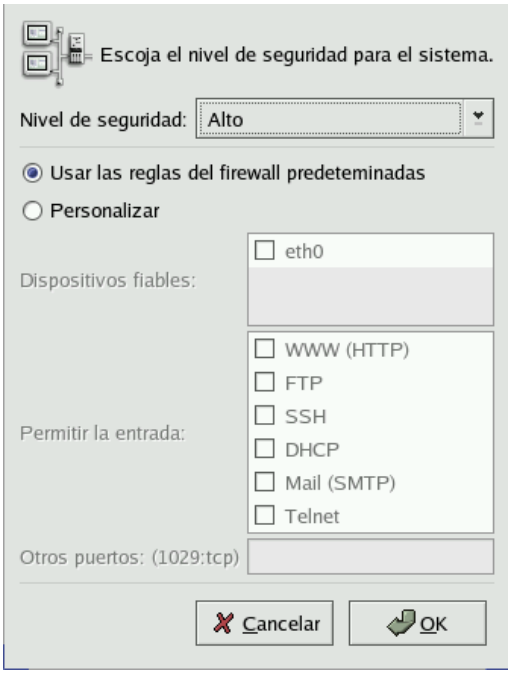


Figura N° 18. Herramienta de configuración del Nivel de Seguridad.

6.4 Activación del servicio iptables

Las reglas de firewall sólo estarán activas si se está ejecutando el servicio iptables. Para arrancar manualmente el servicio, use el comando:

```
/sbin/service iptables restart
```

Para asegurarse de que se ha iniciado al arrancar el sistema, escriba el comando:

```
/sbin/chkconfig --level 345 iptables on
```

También puede usar la aplicación *Servicecon***f** para activar iptables.

6.4.1 Opciones usadas en comandos iptables

Las reglas que permiten a los paquetes ser filtrados por el kernel se ponen en ejecución ejecutando el comando iptables. Cuando use el comando iptables, debe especificar las opciones siguientes:

- *Packet Type:* éste dicta qué tipo de paquetes filtra el comando.
- *Packet Source or Destination:* éste dicta qué paquetes filtra el comando basándose en el origen o destino del paquete.
- *Target:* éste dicta qué acción se lleva a cabo en paquetes que cumplen los criterios mencionados anteriormente.

Las opciones usadas con la regla dada iptables deben Estar agrupadas lógicamente, basándose en el propósito y en las condiciones de la regla general, para que la regla sea válida.

Tablas

Un aspecto muy potente de iptables es que se pueden utilizar múltiples tablas para decidir el destino de un paquete en particular, dependiendo del tipo de paquete que se esté monitorizando y de qué es lo que se va a hacer con el paquete. Gracias a la naturaleza extensible de *iptables* se pueden crear tablas especializadas que se almacenarán en el directorio */etc/modules/<kernel-version>/kernel/net/ipv4/netfilter* para objetivos específicos.

La tabla por defecto, llamada *filter*, contiene las cadenas estándar por defecto para INPUT, OUTPUT, y FORWARD. Esto es parecido a las cadenas estándar que se utilizan con *ipchains*. Además, por defecto, *iptables* también incluye dos tablas adicionales que realizan tareas de filtrado específico de paquetes. La tabla *nat* se

puede utilizar para modificar las direcciones de origen y destino grabadas en un paquete, y la tabla *mangle* permite alterar los paquetes de forma especializada.
Cada tabla contiene cadenas por defecto que realizan las tareas necesarias basándose en el objetivo de la tabla, pero se pueden configurar fácilmente nuevas cadenas en el resto de las tablas.

Estructura

Muchos comandos *iptables* tienen la siguiente estructura:

```
iptables [-t <table-name>] <command> <chain-name> <parameter-1> \
        <option-1> <parameter-n> <option-n>
```

En este ejemplo, la opción *<table-name>* permite al usuario seleccionar una tabla diferente de la tabla filter por defecto que se usa con el comando. La opción *<command>* es el centro del comando, dictando cuál es la acción específica a realizar, como pueda ser añadir o borrar una regla de una cadena particular, que es lo que se especifica en la opción *<chain-name>*. Tras *<chain-name>* se encuentras los pares de parámetros y opciones que realmente definen la forma en la que la regla funcionará y qué pasará cuando un paquete cumpla una regla.

En la estructura de un comando *iptables*, es importante recordar que, al contrario que la mayoría de los comandos, la longitud y complejidad de un comando *iptables* puede cambiar en función de su propósito. Un comando simple para borrar una regla de una cadena puede ser muy corto, mientras que un comando diseñado para filtrar paquetes de una subred particular usando un conjunto de parámetros específicos y opciones puede ser mucho más largo. Al crear comandos *iptables* puede ser de ayuda reconocer que algunos parámetros y opciones pueden crear la necesidad de utilizar otros parámetros y opciones para especificar algo de los requisitos de la opción anterior. Para construir una regla válida, esto deberá continuar hasta que todos los parámetros y opciones que

requieran otro conjunto de opciones hayan sido satisfechos. Teclee iptables -h para ver una lista detallada de la estructura de los comandos *iptables*.

Comandos

Los comandos le dicen a *iptables* que realice una tarea específica. Solamente un comando se permite por cada cadena de comandos *iptables*. Excepto el comando de ayuda, todos los comandos se escriben en mayúsculas. Los comandos de *iptables* son los siguientes:

- -A — Añade la regla iptables al final de la cadena especificada. Este es el comando utilizado para simplemente añadir una regla cuando el orden de las reglas en la cadena no importa.
- -C — Verifica una regla en particular antes de añadirla en la cadena especificada por el usuario. Este comando puede ser de ayuda para construir reglas *iptables* complejas pidiéndole que introduzca parámetros y opciones adicionales.
- -D — Borra una regla de una cadena en particular por número (como el 5 para la quinta regla de una cadena). Puede también teclear la regla entera e *iptables* borrará la regla en la cadena que corresponda.
- -E — Renombra una cadena definida por el usuario. Esto no afecta a la estructura de la tabla. Tan solo le evita el problema de borrar la cadena, creándola bajo un nuevo nombre, y reconfigurando todas las reglas de dicha cadena.
- -F — Libera la cadena seleccionada, que borra cada regla de la cadena. Si no se especifica ninguna cadena, este comando libera cada regla de cada cadena.

- -h — Proporciona una lista de estructuras de comandos útiles, así como una resumen rápido de parámetros de comandos y opciones.
- -I — Inserta una regla en una cadena en un punto determinado. Asigne un número a la regla a insertar e iptables lo pondrá allí. Si no especifica ningún número, *iptables* posicionará su comando al principio de la lista de reglas.

- -L — Lista todas las reglas de la cadena especificada tras el comando. Para ver una lista de todas las cadenas en la tabla *filter* por defecto. La sintaxis siguiente deberá utilizarse para ver todas la lista de todas las reglas de una cadena específica en una tabla en particular:

  iptables -L *<chain-name>* -t *<table-name>*

- -N — Crea una nueva cadena con un nombre especificado por el usuario.
- -P — Configura la política por defecto para una cadena en particular de tal forma que cuando los paquetes atraviesen la cadena completa sin cumplir ninguna regla, serán enviados a un objetivo en particular, como puedan ser ACCEPT o DROP.
- -R — Reemplaza una regla en una cadena en particular. Deberá utilizar un número de regla detrás del nombre de la cadena para reemplazar esta cadena. La primera regla de una cadena se refiere a la regla número 1.
- -X — Borra una cadena especificada por el usuario. No se permite borrar ninguna de las cadenas predefinidas para cualquier tabla.
- -Z — Pone ceros en los contadores de byte y de paquete en todas las cadenas de una tabla en particular.

Parámetros

Una vez que se hayan especificado algunos comandos de *iptables*, incluyendo aquellos para crear, añadir, borrar, insertar, o reemplazar reglas de una cadena en particular, se necesitan parámetros para comenzar la construcción de la regla de filtrado de paquetes.

- -c Resetea los contadores de una regla en particular. Este parámetro acepta las opciones PKTS y BYTES para especificar qué contador hay que resetear.
- -d Configura el nombre de la máquina destino, dirección IP o red de un paquete que cumplirá la regla. Cuando se especifique una red, puede utilizar dos métodos diferentes para describir la máscara de red, como 192.168.0.0/255.255.255.0 o 192.168.0.0/24.
- -f Aplica esta regla solo a los paquetes fragmentados.

Usando la opción ! después de este parámetro, únicamente los paquetes no fragmentados se tendrán en cuenta.

- -i Configura las interfaces de entrada de red, como eth0 o ppp0, para ser usadas por una regla en partículas. Con *iptables*, este parámetro opcional solo debería de ser usado por las cadenas INPUT y FORWARD cuando se utilice junto con la tabla *filter* y la cadena PREROUTING con las tablas *nat* y *mangle*. Este parámetro proporciona varias opciones útiles que pueden ser usadas antes de especificar el nombre de una interfaz:
    - ! — Dice a este parámetro que no concuerde, queriendo decir esto que las interfaces especificadas se excluirán de esta regla.
    - + — Carácter comodín usado para hacer coincidir todas las interfaces que concuerden con una cadena en particular. Por ejemplo, el parámetro -i eth+ aplicará esta regla a todas las interfaces Ethernet de su

sistema excluyendo cualquier otro tipo de interfaces, como pueda ser la ppp0.

Si el parámetro -i se utiliza sin especificar ninguna interfaz, todas las interfaces estarán afectadas por la regla.

- -j Dice a *iptables* que salte a un objetivo en particular cuando un paquete cumple una regla en particular. Los objetivos válidos que se usarán tras la opción -j incluyen opciones estándar, ACCEPT, DROP, QUEUE, y RETURN, así como opciones extendidas que están disponibles a través de módulos que se cargan por defectos con el paquete RPM de iptables de Red Hat Linux, como LOG, MARK, y REJECT, así como otras. Mire la página del manual de iptables para obtener más información sobre este y otros muchos objetivos, incluyendo reglas de ejemplo que los utilizan, así como objetivos que pueden ser usados solamente en una tabla en particular. En lugar de especificar la acción objetivo, puede también dirigir un paquete que cumpla la regla hacia una cadena definida por el usuario fuera de la cadena actual. Esto le permitirá aplicar otras reglas contra este paquete, y filtrarlo mejor con respecto a otros criterios. Si no especifica ningún objetivo, el paquete se mueve hacia atrás en la regla sin llevar a cabo ninguna acción. A pesar de todo, el contador para esta regla se sigue incrementando en uno, a partir del momento en el que el paquete se adecua a la regla especificada.
- -o Configura la interfaz de red de salida para una regla en particular, y solo puede ser usada con las cadenas OUTPUT y FORWARD en la tabla filter y la cadena POSTROUTING en las tablas nat y mangle. Estas opciones de los parámetros son los mismos que para los de la interfaz de red de entrada (opción -i).

- -p Configura el protocolo IP para la regla, que puede ser icmp, tcp, udp, o all(todos), para usar cualquier protocolo. Además, se pueden usar otros protocolos menos usados de los que aparecen en /etc/protocols. Si esta opción se omite al crear una regla, la opción all es la que se selecciona por defecto.
- -s Configura el origen de un paquete en particular usando la misma sintaxis que en el parámetro de destino (opción -d).

UNIVERSIDAD DON BOSCO
FACULTAD DE INGENIERIA
ESCUELA DE INGENIERIA EN COMPUTACION

MANUAL DEL PROGRAMADOR
"DISEÑO Y ELABORACIÓN DE UN SOFTWARE PROTOTIPO PARA LA CREACIÓN DE UN SERVIDOR WEB, ORIENTADO A LA MEDIANA EMPRESA DE EL SALVADOR"

CIUDADELA DON BOSCO DICIEMBRE 2003

INDICE

1. DESCRIPCIÓN DEL SERVIDOR WEB

Un Servidor Web es un programa que funciona sobre una máquina en red, esperando conexiones de otras maquinas llamadas clientes, para servir documentos solicitados desde un navegador, los cuales son almacenados en sus dispositivos de almacenamiento masivo o discos duros. El Servidor Web gestiona y administra el acceso a dichos documentos por parte de otros servidores u otros clientes.

El Servidor Web 1.0 es un software para servir paginas en Internet, escrito en el lenguaje de programación Perl. Esta hecho bajo el esquema de la programación mediante el uso de módulos, lo que hace su expansión o adición de nuevas funcionalidades mucho más fácil y eficiente.

La estructura de archivos que conforman al Servidor Web 1.0, consta de scripts de Perl, módulos de Perl y los archivos de información.

2. PERL

2.1 Aspectos Teóricos

Perl es un lenguaje de programación de todo propósito y fácilmente extensible que se ha vuelto muy popular en los últimos años. Tanto la especificación del lenguaje como su implementación son libres, lo que ha hecho que Perl sea utilizado como lenguaje independiente y sea embebido en una gran cantidad de proyectos con muy diversas naturalezas. Perl es un lenguaje de computadora interpretado.

En Perl las instrucciones terminan en punto y coma (;). Todo lo que este después del signo de numero (#), en la misma línea es un comentario.

2.1.1 Tipos de Variables

Existen 3 tipos básicos de variables en Perl.

1. Escalares: las variables escalares empiezan por $. Un escalar puede tener números, strings o cadenas de caracteres, referencias y descriptores. Ejemplo:

```
$a = 5;
$b ="xxx";
```

2. Arreglos: las variables arreglos empiezan por @. Los elementos de un arreglo son escalares que empiezan por $. Los subíndices empiezan por 0. El escalar $a no tiene que ver nada con $a[ ]. Ejemplo:

```
@a = (95, 7, 'fff' );
```

3. Hashes o arreglos asociativos: las variables hash empiezan por %. Para crear un elemento de un hash se requiere una lista de 2 valores el primer elemento es la clave y el segundo es el valor, si la clave es un string sencillo se puede omitir las comillas. Ejemplo:

```
%a = ( 'x', 5, 'y', 3);
```

2.1.2 Variables Predefinidas

Perl pone a disposición un conjunto bastante rico de variables especiales, es decir, variables gestionadas directamente por el intérprete que contienen parámetros muy útiles.

- **Variables que afectan arreglos**

$[: es el suscrito base de los arreglos [default es 0].

$": el separador de elementos cuando se interpola un arreglo en un string de comilla doble [default es espacio].

- **Variables utilizadas en archivos**

$.: contiene el último número de línea leído.

$/: terminación de registro de entrada [default es '\n'].

$|: si es diferente de cero se vacía el buffer de salida después de print o write

Variables usadas con patrones

$&: contiene el último string que hizo match.

$+: contiene el string que coincidió con el ultimo paréntesis que hizo match.

$1, $2, $3...: Memoria de los matches de los paréntesis.

- **Variables usadas en impresión**

$,: separador de campo para la salida, aplicable a print.

$: separador de registro para la salida, aplicable a print

- **Variables relacionadas con procesos**

$0: el nombre del script de Perl.

$!: número de error o string con el texto del error.

$<: uid real del proceso actual.

$>: uid efectivo del proceso actual.

$(: gid real del proceso actual.

$): gid efectivo del proceso actual.

- **Variables diversas**

$_: contiene el contenido del último registro leído de un fichero.

*$^*: contiene el nombre de la cabecera de formato o report definido para la salida actual.

$$: número de proceso que se esta ejecutando actualmente.

$&: representa el valor de la último cadena de caracteres comparada exitosamente.

@ARGV: contiene los parámetros pasados al Perl script.

%ENV: array asociativo que contiene las variables de entorno bajo el que se ejecuta un Perl script.

osname: Esta variable contiene el nombre del sistema operativo.

2.1.3 Entrada/Salida

La instrucción básica para mostrar el valor de las variables en pantalla es *print*. Lo realmente particular del lenguaje PERL es el método que usa para la entrada de datos desde el teclado. Para asignar un valor desde el teclado a una variable, se asigna a la variable la representación del teclado <STDIN> (STandarDINput).

2.1.4 Programación Orientada a Objetos en Perl

Perl también soporta la Programación Orientada a Objetos (POO). Empezando por el principio, una clase es una colección de variables y de funciones que acceden a esas variables. Un objeto es una instanciación particular de una clase. En Perl, casi todos los módulos son, en realidad, objetos.

Constructores y Destructores

Cuando existe una jerarquía de herencia, es común un diseño que usa una delegación planificada en las clases antecesoras. Esto es: el código de un inicializador llama a los correspondientes inicializadores de las clases antepasadas y añade las correspondientes inicializaciones de la clase. El proceso de destrucción es recíproco. Este esquema es posiblemente más usual que el reemplazamiento completo del método por uno nuevo. Lo habitual es que el nombre de un constructor sean *new*.

2.1.5 Modulos en Perl

Un módulo Perl es un fichero de texto (con el sufijo .pm) que contiene código Perl. El fichero debe colocarse en uno de varios directorios estándar en los que el compilador busca (la variable de entorno PERL5LIB gobierna la búsqueda, también puedes usar la opción -I del compilador). Esa lista esta disponible en un programa Perl a través de la variable @INC. El compilador sustituye cada :: por el separador de caminos. Así la orden: use Text::ParseWords; se traduce por el fichero Text/ParseWords.pm. En nereida, por ejemplo, el directorio exacto es

/usr/lib/Perl5/5.00503/Text/ParseWords.pm. El compilador Peal abre el primer fichero que case o coincida con Text/ParseWords.pm y evalúa el texto en su interior. Si falla, la compilación termina con un mensaje de error. En otro caso, el compilador busca en el módulo por una rutina denominada import, y si existe la ejecuta. Cuando esta termina la compilación continúa en el fichero original, justo después de la línea en la que aparece la sentencia *use*.

Métodos para la inclusión de módulos y bibliotecas

Hay tres métodos para incluir archivos de código: *do, require y use*. Los archivos incluidos con los dos primeros son bibliotecas, mientras que los incluidos con use son módulos.

3. MODULOS ESTANDAR DE PERL UTILIZADOS

Data::Dumper

Se utiliza para volcar estructuras de datos en Perl para imprimir o evaluar.

File::Copy

Copia archivos o manejadores de archivos.

HTTP::Daemon

Implementa una clase simple de un servidor http. Entre los métodos más importantes están:

accept()

Realiza una llamada al sistema para monitorear en el puerto. Retorna un objeto *HTTP::Daemon::ClientConn* que hace referencia a una conexión con un cliente.

url

Retorna u cadena URL que puede ser utilizada para accesar al servidor raíz.

get_request()

Lee datos desde el cliente y activa un objeto *HTTP::Request* el cual es retornado.

send_response( )

Escribe un objeto *HTTP::Response* al cliente como respuesta.

send_error( )

Envía un error como respuesta a una solicitud del cliente.

timeout( )

Establece un valor de timeout para el puerto.

HTTP::Response

Clase que encapsula respuestas http. Construye un objeto HTTP::Response, describiendo una respuesta con un código y un mensaje opcional.

HTTP::Request

Clase que encapsula requisiciones o solicitudes http, que consisten de una línea de requisición, algunos encabezados y el contenido.

IPC::Open3

Abre un proceso para lectura, escritura y manejo de errores

Sys::Hostname

Obtiene el hostname de la computadora.

Config

Accesa a la información de configuración de Perl.

Cwd

Obtiene la ruta del directorio actual de trabajo.

Errno

Constantes de Errores del Sistema.

Exporter

Implementa un método por default para importar métodos.

Env

Importa variables de entorno como escalares o arreglo.

POSIX

Interface de Perl para es estándar IEEE 1003.1

Strict

Verifica que todas las variables hayan sido inicializadas correctamente.

Vars

Predeclarar nombres de variables globales.

4. FUNCIONES PREDEFINIDAS UTILIZADAS

A continuación se describen algunas de las funciones incluidas en la distribución de Perl que se han utilizado en la programación de los script del Servidor Web 1.0.

bless: cambia el tipo de una referencia a otro tipo.

binmode: Abre un file en la modalidad binaria; tiene que ser invocado después de ejecutar open.

chdir: nos permite cambiar de directorio dentro de la jerarquía de directorios.

chop: recorta y retorna el último carácter de una cadena.

chown: cambia el propietario de los ficheros dados.

close : cierra un fichero.

defined: sirve para comprobar si existe una variable, formato, subrutina.

delete: borra un valor de un array asociativo a través de su clave.

die: imprime en la salida del error estándar un mensaje pasado como parámetro cuando ocurre un error en la ejecución de una sentencia.

eof: retorna verdadero si el final del fichero dado.

eval: evalúa la expresión pasada como si se tratase de un pequeño programa Perl.

exec: ejecuta lo que pasemos como parámetro y sale del programa.

exit: hace que salgamos del Perl script devolviendo al sistema operativo el valor pasado como argumento.

exists: prueba si existe la key en el hash.

fileno: devuelve el descriptor del manejador del fichero pasado como parámetro.

fork: realiza una llamada fork para crear un proceso.

join: une las cadenas pasadas como argumento con un separador también pasado como argumento.

keys: devuelve todas las claves de un array asociativo.

kill: Envía una señal a un proceso. Se puede usar como kill 'STOP' proceso; para las señales, véase man 7 signal.

last: la ejecución interrumpe el ciclo actual.

length: devuelve la longitud en caracteres del parámetro pasado.

local: declara como locales las variables pasadas como argumentos.

mkdir: crea un directorio en el camino dado.

my: define variables locales.

open: abre el fichero dado asociándole un manejador de fichero especificado también como parámetro.

pop: retorna y borra el ultimo elemento del array dado.

print: muestra en la salida standard o en el fichero especificado la expresión dada.

push: añade el valor dado al final del array pasado como parámetro.

read: lee un determinado numero de caracteres desde el fichero pasado como argumento.

require: sirve para incluir código externo en nuestro guión.

return: devuelve un valor desde una subrutina.

seek: sitúa un puntero a fichero en un lugar determinado.

select: sirve para seleccionar el manejador de fichero que sera utilizado por defecto para la salida de los comandos o funciones que no especifiquen un determinado manejador de fichero como parámetro.

shift: devuelve el primer valor del array dado borrándolo posteriormente.

split: divide una cadena en subcadenas según el separador especificado.

substr: extrae un string de otro.

system: igual que exec pero no se sale del Perl script.

undef: vuelve el argumento, no definido.

values: devuelve todos los valores del array asociativo dado.

write: escribe un registro con formato en el fichero asociado a ese formato.

DATA: este descriptor se refiere a todo lo que tiene el archivo del programa después de la línea __END__, antes de la línea __END__ esta el programa Perl; la línea __END__ es opcional.

5. ESTRUCTURA DE ARCHIVOS

5.1 Scripts

Los scripts desarrollados para el Servidor Web 1.0 consisten en archivos de código en Perl de extensión.pl. a continuación se describen.

setup.pl

este archivo de código es el que se encarga de instalar y configurar el software del Servidor Web.

sep.pl

Este constituye el programa principal, a través del cual se controla el funcionamiento de las operaciones básicas del Servidor Web, inicio, reinicio y finalización del mismo. Después de iniciar el servidor, este programa pasa el control al modulo Server, para el proceso de atención de peticiones.

5.2 Módulos

Los módulos desarrollados para el Servidor Web 1.0 consisten en archivos de código en Perl, de extensión .pm (de Perl Module). Cada uno de ellos se ha definido mediante el uso de la palabra reservada package, que indica que ese archivo es un modulo de Perl. Cada modulo implementa un proceso, además que dentro de los mismos se pueden desarrollar métodos o funciones pueden ser invocados por ellos mismos o por otros scripts u otros módulos.

Para agregar un modulo al Servidor Web, solamente es necesario copiar el archivo.pm en el directorio de módulos, Módulos. Posteriormente, editar el archivo de configuración, con el modulo agregado y el tipo de archivo que lo manejara. A continuación se describe cada modulo desarrollado.

Config

Modulo elaborado para leer y accesar a la información de configuración del Servidor Web, registrada en el archivo server.cfg. Este es el modulo de lectura de configuración. Este implementa los siguientes métodos.

Método load_config()

Carga los parámetros de configuración del Servidor Web.

Método load_mime_types()

Carga el tipo y la extension de un archivo.

Logs

Este modulo el acceso y la escritura en los archivos de logeo con el fin de registrar la información sobre los accesos al Servidor Web y los errores ocurridos en el proceso de atención de peticiones.

Server

Modulo que implementa el método que crea e inicia el Servidor Web, constituye la estructura que controla el Servidor Web. Además implementa un método que obtiene la solicitud del cliente, además de los métodos para construir y enviar el mensaje de respuesta al cliente.

Método Server()

Constituye la función principal, crea el Servidor Web, inicia el lazo de espera de conexiones entrantes.

Método envio()

Construye y envía el mensaje de respuesta para el cliente, a través de llamadas a los métodos new_response() y load_page()

Método new_response()

Adjunta al mensaje de respuesta para el cliente, la información de cabecera.

Método load_page()

Carga la pagina solicitada y adjunta al mensaje de respuesta para el cliente, la información de contenido.

Server2

Modulo que se encarga de encontrar y retornar el archivo o pagina solicitada.

Método find_file()

Encuentra el archivo o pagina solicitada y envía el resultado al método load_page()

Método get_extention()

Obtiene el tipo y la extensión del archivo y el manejador del mismo en caso de que sea un script cgi.

mod_cgi

Modulo que implementa el manejo de scripts CGI, de extensión.pl o.cgi.

Método get_new_env()

Actualiza el arreglo asociativo %ENV, con la información del entorno.

5.3 Archivos de Información

Los archivos de información son archivos de texto plano en los que se registra información sobre la configuración, tipos de datos e información de logeo del Servidor Web. Estos archivos también los constituyen aquellos que el servidor utiliza para realizar algunas de sus funciones. A continuación se describe cada uno de ellos.

server.cfg

Este es el archivo en el que se encuentra registrada la configuración actual del Servidor Web. Esta información es actualizada por el programa de instalación, cuando configura el software y es accesado por la librería o modulo de lectura de configuración.

mime.types

Archivo en que se encuentran registrados los tipos de datos y la extensión correspondiente que el Servidor Web soporta.

pid.file

Este archivo almacena el pid o el id del proceso que ejecuta o el numero de proceso que tiene en el sistema operativo el Servidor Web. Este archivo es actualizado cada vez que se inicia el Servidor Web, para almacenar el numero de proceso, y cada vez que se finaliza, para leer el numero del proceso y eliminarlo.

access.log

Archivo que registra información sobre los accesos al Servidor Web por parte de los clientes.

error.log

En este se registra toda la información referente a los errores que ocurren durante los procesos que el Servidor Web ejecuta.

usuarios.txt

En el se almacenan los nombres de usuario y las contraseñas de los usuarios registrados para accesar al Servidor Web, cuando el modo protección esta activado. Por otra parte, la primera línea de este archivo, es la bandera que indica si el Servidor Web esta protegido o no. Si lo esta el valor corresponde a una letra A, de lo contrario el valor es una D.

pas.txt

Este archivo guarda temporalmente la contraseña del usuario que desea accesar al Servidor Web, cuando el modo protección esta activado.

temp.txt

archivo que almacena temporalmente la pagina que un cliente ha solicitado cuando el modo protección esta activado.

Readme.txt

Este es un archivo de información general sobre el Servidor Web.

Index.html

Es la pagina de inicio o pagina por default del Servidor Web, se ubica dentro del directorio html, que es el directorio raíz de documentos, es decir, el directorio donde se deben almacenar los archivos o paginas web, para que estas puedan ser servidas a los clientes.

Passw.html

Es la pagina de autenticación del Servidor Web, en la cual el cliente debe ingresar su nombre de usuario y su password, si el modo protección esta activado. Se ubica en el directorio raíz de documentos, html.

6. CODIGO DEL PROGRAMA

6.1 Programa de Instalación

```
#!/usr/bin/perl -w
# setup.pl - script de instalación del servidor web 1.0

use strict;
use vars qw/$VERSION $SEP/;
use Config;
use Sys::Hostname;
use File::Copy;

$VERSION = "Servidor Web 1.0";
BEGIN {
   $SEP = '/'; }
my $OStype = $Config{'osname'};
my %defaults = (InstallLocation => (is_unix($OStype) && $>) ?
                        (getpwuid $>)[7] .$SEP.'swp' :
                        ($Config{installscript} || '') .$SEP.'swp',
                        Port => 80,
                        Hostname => hostname(),
                        UserRoot => '/home',
                        UserHtmlDir => 'public_html',
                        );
my %actual;
print <<EO_HEADER;
#########################################################################
CONFIGURACION $VERSION
INTRODUZCA LOS VALORES PARA CONFIGURAR EL SOFTWARE.
PARA SELECCIONAR EL VALOR POR DEFAULT SOLO PRESIONE ENTER.
PARA INSTALAR ESTE SOFTWARE CORRECTAMENTE ASEGURESE DE UBICAR EL DIRECTORIO,
DE INSTALACION swp, EN EL DIRECTORIO /root DE SU COMPUTADORA.
EO_HEADER

# RUTA DE INSTALACION
print "- LA RUTA DONDE SE INTALARA EL SOFTWARE ES $defaults{InstallLocation}\n";
$actual{InstallLocation} = $defaults{InstallLocation};
```

```
# PUERTO DE ESCUCHA
$actual{Port} = Input("- SELECCIONE EL PUERTO DE ESCUCHA (RECOMENDADO 80 o 1024)", $defaults{Port});

# EL NOMBRE DE LA COMPUTADORA
$actual{Hostname} = Input("- SELECCIONE EL NOMBRE DE DOMINIO DEL SERVIDOR WEB", $defaults{Hostname});

$actual{UserRoot} = $defaults{UserRoot};
$actual{UserHtmlDir} = $defaults{UserHtmlDir};

#CHEQUEO DE INSTALACION COMO USUARIO ROOT
my $Uid = $>;
if (is_unix($OStype) && ($actual{Port} == 80) && ($Uid != 0)) {
      print "** USTED DEBE TENER PRIVILEGIOS DE USUARIO ROOT PARA INSTALAR SOFTWARE.\n";
            print "O, PUEDE INTENTAR CON EL PUERTO > 1024.\n";
      exit;
}

# CREACION DE LOS DIRECTORIOS
print "\nCREANDO DIRECTORIOS\n";
mkdir("$actual{InstallLocation}", 0755) || print "IMPOSIBLE CREAR EL DIRECTORIO $actual{InstallLocation}  - $!\n";
mkdir("$actual{InstallLocation}${SEP}lib", 0755) || print "IMPOSIBLE CREAR EL DIRECTORIO $actual{InstallLocation}/lib
- $!\n";
mkdir("$actual{InstallLocation}${SEP}lib${SEP}swp", 0755) || print "IMPOSIBLE CREAR EL DIRECTORIO
$actual{InstallLocation}/lib/PerlWebServer - $!\n";
mkdir("$actual{InstallLocation}${SEP}lib${SEP}swp${SEP}Modulos", 0755) || print "IMPOSIBLE CREAR EL
DIRECTORIO $actual{InstallLocation}/lib/PerlWebServer/Module - $!\n";
mkdir("$actual{InstallLocation}${SEP}logs", 0755) || print "IMPOSIBLE CREAR EL DIRECTORIO
$actual{InstallLocation}/logs - $!\n";
mkdir("$actual{InstallLocation}${SEP}html", 0755) || print "IMPOSIBLE CREAR EL DIRECTORIO
$actual{InstallLocation}/html - $!\n";
mkdir("$actual{InstallLocation}${SEP}html${SEP}imagenes", 0755) || print "IMPOSIBLE CREAR EL DIRECTORIO
$actual{InstallLocation}/html/images - $!\n";

# COPIANDO ARCHIVOS
print "COPIANDO LOS ARCHIVOS NECESARIOS\n";
            my $file;
            ######## dentro de principal ############
            $file="/root/swp/swp/setup.pl";
            print "COPIANDO $file => $actual{'InstallLocation'}${SEP}\n";
            system("cp $file $actual{InstallLocation}${SEP}setup.pl -R");
            $file="/root/swp/swp/sep.pl";
            print "COPIANDO $file => $actual{'InstallLocation'}${SEP}\n";
            system("cp $file $actual{InstallLocation}${SEP}sep.pl -R");
            $file="/root/swp/swp/mime.types";
            print "COPIANDO $file => $actual{'InstallLocation'}${SEP}\n";
            system("cp $file $actual{InstallLocation}${SEP}mime.types -R");
            $file="/root/swp/swp/Readme.txt";
            print "COPIANDO $file => $actual{'InstallLocation'}${SEP}\n";
            system("cp $file $actual{InstallLocation}${SEP}Readme.txt -R");
            $file="/root/swp/swp/server.cfg";
            print "COPIANDO $file => $actual{'InstallLocation'}${SEP}\n";
            system("cp $file $actual{InstallLocation}${SEP}server.cfg -R");
```

```
                $file="/root/swp/swp/temp.txt";
                 print "COPIANDO $file => $actual{'InstallLocation'}${SEP}\n";
                 system("cp $file $actual{InstallLocation}${SEP}temp.txt -R");
                $file="/root/swp/swp/usuarios.txt";
                 print "COPIANDO $file => $actual{'InstallLocation'}${SEP}\n";
                 system("cp $file $actual{InstallLocation}${SEP}usuarios.txt -R");
                $file="/root/swp/swp/pas.txt";
                 print "COPIANDO $file => $actual{'InstallLocation'}${SEP}\n";
                 system("cp $file $actual{InstallLocation}${SEP}pas.txt -R");
                 ######## dentro de html ############
                 $file="/root/swp/swp/html/index.html";
                 print "COPIANDO $file => $actual{'InstallLocation'}${SEP}html\n";
                 system("cp $file $actual{InstallLocation}${SEP}html/index.html -R");
                 ######## dentro de images ############
                 $file="/root/swp/swp/html/imagenes/logob.jpg";
                 print "COPIANDO $file => $actual{'InstallLocation'}${SEP}html/imagenes\n";
                 system("cp $file $actual{InstallLocation}${SEP}html/imagenes/logob.jpg -R");
                 ######## dentro de lib ############
                 $file="/root/swp/swp/lib/swp/Config.pm";
                 print "COPIANDO $file => $actual{'InstallLocation'}${SEP}lib/swp\n";
                 system("cp $file $actual{InstallLocation}${SEP}lib/swp/Config.pm -R");
                 $file="/root/swp/swp/lib/swp/Daemon.pm";
                 print "COPIANDO $file => $actual{'InstallLocation'}${SEP}lib/swp\n";
                 system("cp $file $actual{InstallLocation}${SEP}lib/swp/Daemon.pm -R");
                 $file="/root/swp/swp/lib/swp/Logs.pm";
                 print "COPIANDO $file => $actual{'InstallLocation'}${SEP}lib/swp\n";
                 system("cp $file $actual{InstallLocation}${SEP}lib/swp/Logs.pm -R");
                 $file="/root/swp/swp/lib/swp/Server2.pm";
                 print "COPIANDO $file => $actual{'InstallLocation'}${SEP}lib/swp\n";
                 system("cp $file $actual{InstallLocation}${SEP}lib/swp/Server2.pm -R");
                 $file="/root/swp/swp/lib/swp/Modulos.pm";
                 print "COPIANDO $file => $actual{'InstallLocation'}${SEP}lib/swp\n";
                 system("cp $file $actual{InstallLocation}${SEP}lib/swp/Modulos.pm -R");
                 $file="/root/swp/swp/lib/swp/Server.pm";
                 print "COPIANDO $file => $actual{'InstallLocation'}${SEP}lib/swp\n";
                 system("cp $file $actual{InstallLocation}${SEP}lib/swp/Server.pm -R");
                                 ######## dentro de Modulos ############
                 $file="/root/swp/swp/lib/swp/Modulos/mod_cgi.pm";
                 print "COPIANDO $file => $actual{'InstallLocation'}${SEP}lib/swp/Modulos\n\n\n";
                 system("cp $file $actual{InstallLocation}${SEP}lib/swp/Modulos/mod_cgi.pm -R");

# ACTUALIZANDO EL ARCHIVO DE CONFIGURACION
print "CONFIGURANDO $actual{InstallLocation}${SEP}server.cfg\n";
if (open(CFG, "> $actual{InstallLocation}${SEP}server.cfg")) {
   $actual{SEP} = $SEP;
   while (my $line = <DATA>) {
            $line =~ s/\%\%([^\%]+)\%\%/$actual{$1}/g;
            print CFG $line;
   }
   close(CFG);
} else {
   print <<EOF;
```

```
ERROR - IMPOSIBLE ESCRIBIR EN EL ARCHIVO DE CONFIGURACION.
PORFAVOR COPIE EL ARCHIVO 'server.cfg' A $actual{InstallLocation} Y EDITELO.
EOF
   exit(0);
}
print "\nPARA INICIAR EL SERVIDOR PUEDE DIGITAR:\n$actual{InstallLocation}${SEP}sep.pl start\n\n";

# INPUT, SUBRUTINA PARA OBTENER LA INFORMACION DEL USUARIO
sub Input {
   my ($msg,$def) = @_;
   $def ||= '';
   print "$msg: [$def] ";
   my $input = <STDIN>;
   chop($input);
   if ($input eq "") {
            return $def;
   } else {
            return $input;
   }
}

#SUBRUTINA PARA VERIFICAR EL SISTEMA OPERATIVO
sub is_unix {
   my $type = shift || $Config{'osname'};
   return 0  unless $type;
   return ($type =~ /n[iu]x$/i);
}

# ARCHIVO DE CONFIGURACION
__DATA__
#####################################################################
# Este es el Archivo de configuracion para el Servidor Web.

<Scope Server>
# Ruta de Instalacion del Servidor
ServerRoot            %%InstallLocation%%

# Archivo de Procesos
PIDFile    __ServerRoot__%%SEP%%pid.file

# Archivos de Logeo
AccessLog __ServerRoot__%%SEP%%logs%%SEP%%access.log
ErrorLog  __ServerRoot__%%SEP%%logs%%SEP%%error.log

# Numero del puerto de escucha
ServerPort %%Port%%
</Scope>

#####################################################################
# Nombre del Host
<Scope %%Hostname%%>
```

```
# Ruta de Directorio Raiz
DocumentRoot __ServerRoot__%%SEP%%html

# Definicion de la pagina de inicio
DirectoryIndex index.html index.htm index.cgi

# Agregar Modulos basicos
AddModule mod_cgi

# Manejadores de Archivos
AddType cgi_script     cgi pl
</Scope>
```

6.2 Programa Principal

```
#!/usr/bin/perl -w
# sep.pl
# Script de control que inicia o detiene el Servidor Web de acuerdo a un argumento recibido

use strict;

# Identificando el directorio actual para cargar las librerias
BEGIN {
   my $actual_path = $0;
   $actual_path =~ s/sep\.pl//;
   eval "use lib '${actual_path}lib';
        use swp::Server;
        use swp::Config;
        use swp::Server2;
        ";
   die "Error al Cargar las Librerias\n$@"  if ($@);
}

# Cargando la configuracion
my $config = new swp::Config();

# estableciendo una pausa para los procesos hijos finalizados
local $SIG{CHLD} = sub { wait; };

# Identificacion del argumento recibido e inicio o finalizacion del
# servidor web, de acuedo al mismo
   if (my $arg = shift(@ARGV)) {
            if ($arg =~ /^start/i) {
               my $pid1 = fork;
               exit(0)  if ($pid1);
               my $pid2 = fork;
               exit(0)  if ($pid2);
               my $server = new swp::Server();
               local (*PID);
               if (open(PID,">> ". $config->pidfile)) {
                        print PID "$$\n";  # Almacena el pid (process id) del nuevo proceso del servidor
                        close(PID);
```

```
            print "\n...Iniciando Servidor Web 1.0\n";
            $server->server();
    } else {
            print "Error - Imposible escribir en el Archvio pid $!";
            exit(1);
    }
} elsif ($arg =~ /^stop/i) {
    local (*PID);
    if (open(PID,"< ". $config->pidfile)) {
            while (my $line = <PID>) {
                chop($line);
                if (! kill 9, $line) {
                        print "Imposible eliminar el proceso $line!\n";
                } else {
                        print "Proceso Eliminado $line.\n";
                }
            }
            close(PID);
            open(PID,"> ". $config->pidfile);
            close(PID);
    } else {
            print "Error - Imposible Leer el Archivo pid $!";
            exit(1);
    }
} elsif ($arg =~ /^restart/i) {
    local (*PID);
    if (open(PID,"< ". $config->pidfile)) {
            while (my $line = <PID>) {
                chop($line);
                if (! kill 9, $line) {
                        print "Imposible eliminar el proceso $line!\n";
                } else {
                        print "Proceso Eliminado $line.\n";
                }
            }
            close(PID);
            open(PID,"> ". $config->pidfile);
            close(PID);
    } else {
            print "Error - Imposible Leer el Archivo pid $!";
    }
    my $pid1 = fork;
    exit(0)  if ($pid1);
    my $pid2 = fork;
    exit(0)  if ($pid2);
    my $server = new swp::Server();
    {
            local (*PID);
            if (open(PID,">> ". $config->pidfile)) {
                print PID "$$\n";
                close(PID);
                print "\n...Iniciando Servidor Web 1.0\n";
```

```
                    $server->server();
                } else {
                    print "Error - Imposible escribir en el Archvio pid $!";
                    exit(1);
                }
            }
        }
    }

exit(0);
```

6.3 Módulo Config

```
package swp::Config;
#librerÃa para cargar la configuracion del Servidor Web 1.0

use strict;
use Exporter;
use Cwd;
use Config;
use swp::Logs;
use vars qw(@ISA @EXPORT $AUTOLOAD $band_escribir);
use swp::Server2;
use Data::Dumper;

@ISA = ('Exporter');
$band_escribir = 0;
my $basedir = $0;
$basedir =~ s/(sep\.pl)//;
my $arc_conf = "server.cfg";
my $arc_mime = "mime.types";
my $arc_pid = "pid.file";

my %_scope_hash;
my $_default_host = '';
my %_handlers;
my %_handler_ext;
my $modulos_cargados = 0;
my $mimes_cargados = 0;
$Global::CONFIG_LOADED = 1;

#constructor de la clase
sub new {
  my $pkg = shift;
  my %hash = (_permitted => {}, _scope => 'server');
  my $ref = \%hash;
  bless $ref, $pkg;
  my %args = @_;
  $band_escribir = 1  if (defined $args{'write'});
  $ref->load_config();
```

```
  $ref->load_mime_types()  unless ($mimes_cargados);
  return $ref;
}

#metodo para cargar la configuracion del servidor
sub load_config {
  my $self = shift;
  local (*IN);
  my $dir = $basedir;
  my $file_name = ($dir) ? $dir.$SEP.$arc_conf : $arc_conf;
  if (-f $file_name && -r $file_name) {
     if (open(IN,$file_name)) {

                while (my $linea = <IN>) {
                   $linea =~ s/^\s//o;
                   $linea =~ s/#.*$//o;
                   $linea =~ s/\s+$//o;
                   next  unless ($linea);
                   while ($linea =~ /__\w+__/o) {
                            $linea =~ s/__(\w+)__/$self->$1()/e || die "Error en Archivo de configuraciÃ³n:\nValor desconocido
en la lÃnea ". $. .": $linea\n";
                   }
                   if ($linea =~ /^<Scope\s+([^>]+)>$/io) {
                            $self->scope($1);
                   } elsif ($linea =~ /^<\/Scope>$/io) {
                            $self->scope('server');
                   } elsif ($linea =~ /^(\w+)\s+(.*)$/o) {
                            my ($key,$val) = ($1,$2);
                            if ($val =~ /\s/o) {
                               $self->add_attribute($key, [split(/\s+/, $val)]);
                            } else {
                               $self->add_attribute($key, $val);
                            }
                   } else {
                            die "Error en Archivo de ConfiguraciÃ³n:\nSintaxis desconocida en la lÃnea ". $. .": $linea\n";
                   }
                }
                $modulos_cargados = 1;
                close(IN);
     } else {
                die "Imposible abrir el Archivo de ConfiguraciÃ³n: $file_name $!\n";
     }
  } else {
     die "Imposible abrir el Archivo de ConfiguraciÃ³n: $file_name $!\n";  } }

#cargar los tipos mime
sub load_mime_types {
  my $self = shift;
  foreach my $dir (cwd(),$self->ServerRoot) {
    if (-f $dir.$SEP.$arc_mime) {
      $arc_mime = $dir.$SEP.$arc_mime;
      last;
```

```
    }
  }
  local (*IN);
  if (open(IN,$arc_mime)) {
    while (my $linea = <IN>) {
      chomp($linea);
      $linea =~ s/^\s+//o;
      $linea =~ s/\#.*$//o;
      next  unless ($linea);
      $linea =~ s/\s+$//o;



      if ($linea =~ /^(\S+)\s+(.*)$/o) {
              my ($key,$val) = ($1,$2);
              foreach (split(/\s+/, $val)) {
                 next  unless (/\S/);
                 $mime{$_} = $key;
              }
      }
    }
    $mimes_cargados = 1;
    close(IN);
  } else {

  }
}

sub scope {
    my ($self) = shift;
    $_default_host ||= $self->{_scope}  if ($self->{_scope} ne 'server');
    return (@_) ? $self->{_scope} = lc(shift) : $self->{_scope};
}

# interpreta add_attribute
sub add_attribute {
  my ($self,$key,$val) = @_;
  return  unless (defined $key);
  $key = lc $key;
  if ($key eq 'addmodule' and (! $modulos_cargados)) {
      $self->add_module($val);
  } elsif ($key eq 'addtype' and (! $modulos_cargados)) {
      my $tmp = shift(@$val);
      $self->handler_type($val,$tmp);
  } else {
    $self->{_permitted}{$self->scope}{$key} = 1;
    $_scope_hash{$self->scope}{$key} = $val;

    if ($self->scope eq 'server') {
              no strict 'refs';
              ${"Global::". uc $key} = $val;
```

```
    }
  }
}
# interpretar add_module
sub add_module {
    my $self = shift;
    my $mod = shift or return;
    my $obj;
    my $code = qq{
            require swp::Modulos::$mod;
            \$obj = new swp::Modulos::$mod();
    };
    eval $code;
    if ($@) {
            logError("Error al cargar el Modulo: $mod $@");
            return;
    }

    if ($obj->init() == M_OK) {
            $self->handler($obj->handler_type,$obj);
    }
}
#definir si existe un manejado
sub handler {
    my ($self,$type,$obj) = @_;
    return (defined $obj) ? $_handlers{$self->scope}{$type} = $obj :
            $_handlers{$self->scope}{$type} || undef;
}
#encontrar el tipo de manejador
sub handler_type {
    my $self = shift;
    my $ext = shift;
    if (my $new_type = shift()) {
            foreach (@$ext) {
               $_handler_ext{$self->scope}{$_} = $new_type;
            }
    }
    return $_handler_ext{$self->scope}{$ext} || 'text';
}

sub mime_type {
  my ($self,$ext) = @_;
  $ext ||= '';
  $ext =~ s/^\.//;
  return (defined $mime{$ext}) ? $mime{$ext} : 'text/html';
}



#determinar el sistema operativo
sub OSTYPE {
```

```
  return $Config{'osname'};
}

#confirmar si el sistema operativo es linux
sub is_unix {
    my $self = shift;
    return ($Config{'osname'} =~ /n[ui]x$/i) ? 1 : 0;
}
1;
```

6.4 Módulo Logs

```
package swp::Logs;
#librerÃa que define y carga los archivos de logeos y accesos

use strict;
use Exporter;
use Cwd;
use vars qw(@ISA @EXPORT);

@ISA = ('Exporter');
@EXPORT = qw(logAccess logError);
my $accesslog = '';
my $errorlog = '';

#obtener la ruta del archivo
sub get_files {
    if (defined $Global::SERVERROOT) {
            $accesslog ||= $Global::ACCESSLOG;
            $errorlog ||= $Global::ERRORLOG;
    } elsif (defined $Global::CONFIG_LOADED) {
            require swp::Config;
            my $cfg = new swp::Config();
            $accesslog ||= $cfg->AccessLog;
            $errorlog ||= $cfg->ErrorLog;
    } else {
    }
}

#si el archivo access.log es obtener su ruta y escribir en el
sub logAccess {
    get_files();
    logCustom($accesslog,@_);
}

#si el archivo es error.log obtener su ruta y escribir en el
sub logError {
    get_files();
    logCustom($errorlog,@_);
}
 #abrir el archivo y escribir
```

```
sub logCustom {
   my ($file,@args) = @_;
   local (*OUT);
   if (open(OUT,">> $file")) {
            print OUT join("\n",@args),"\n";
            close(OUT);
   }
}
1;
```

6.5 Módulo Server

```
package swp::Server;
#librerÃa principal que implementa el Servidor Web 1.0

use strict;
use swp::Logs;
use swp::Server2;
use POSIX;
#use Cwd;
use Socket;
use vars qw($AUTOLOAD);
my $rere=0;
my $server;

# senal de espera para terminar procesos hijos
local $SIG{CHLD} = sub { wait; }  if (defined $Global::UNIX);

# constructor de la clase
sub new {
   my $pkg = shift;
   my %hash = (_config => undef,
                        _methods => {'GET'=>1,'POST'=>1,'HEAD'=>1},
                        );
   my $self = \%hash;
   bless $self, $pkg;
   $self->config;
   return $self;
}

# Cargando el archivo de configuracion, retornando un objeto de la clase swp::Config
sub config {
   my $self = shift;
   unless (defined $self->{_config}) {
            require swp::Config;
            $self->{_config} = new swp::Config();
   }
   return $self->{_config};
}
```

```
# Obtener el nombre y la version del servidor web
sub token_producto {
    return $server->token_producto;
}

# Informacion estandar del servidor web
sub page_footer {
    my $self = shift;
    return "<hr>$CRLF". $self->token_producto ."$CRLF";
}


# Creando un nuevo objeto de la clase estandar de perl HTTP::Response
sub new_response {
    my ($self, $req_page) = @_;
    require HTTP::Response;
    my $resp_head = new HTTP::Response;
    my $req_page_ext = $req_page || '';
    $req_page_ext =~ s/^[^\.]+//;
    $resp_head->code('200');
    $resp_head->message('OK');
    $resp_head->protocol('HTTP/1.1');
    $resp_head->content_type($self->config->mime_type($req_page_ext));
    $resp_head->date(time);
    $resp_head->server($self->token_producto);
    $resp_head->header('Connection', 'close')  unless (defined $Global::UNIX);
    return $resp_head;
}
# Preparando una respuesta a la peticion y cargar la pagina solicitada
sub envio {
    my ($self, $req_head, $sock) = @_;
    my $meth = $req_head->method;
    return undef  unless (defined $self->{_methods}{$meth});
    my ($host,$port) = split(':', ($req_head->header('Host') || ''), 2);
    $host ||= $self->config->default_host();
    my $old_scope = $self->config->scope();
    my $host2 = $self->config->scope($self->config->has_scope($host) || $self->config->default_host());
    my $resp_head = $self->new_response($req_head->url->path);
    my $peer = $sock->peername();
    my ($peer_port,$peer_iaddr) = unpack_sockaddr_in($peer);
    $req_head->header('Peer_addr', inet_ntoa($peer_iaddr));
    my $text = $self->load_page($req_head,$resp_head);
    $resp_head->content_length(length($text));
    $resp_head->content($text)  if ($text);
    if ($meth eq 'HEAD') {
            $resp_head->header('Connection', 'close');
            $resp_head->content(undef);
    }
    $self->config->scope($old_scope);
    return $resp_head;
}
```

```
# Funcion principal que crea el servidor web e inicia el lazo para la espera de conexiones
sub server {
   my $self = shift;
   die "Es Necesaria una ConfiguraciÃ³n para iniciar el Servidor!"  unless ($self->config);
   unless (defined $server) {
            require swp::Daemon;
            $server = new swp::Daemon(LocalPort => $self->config->ServerPort,
                                                      LocalAddr => $self->config->default_host,
                                                      Listen => 5,
                                                      Reuse => 1
                                                      );
   }
   # Aceptando una conexion
   while (my $new_sock = $server->accept()) {
                        # estableciendo el timeout en segundos
                        $new_sock->timeout(150);
            while (my $request_obj = $new_sock->get_request()) {

# inicia proceso de autenticacion si esta activada
my @valb=split /;/,$request_obj->as_string();
my $ix;
my $vb;
my $c1="";
my $c2="";
my $c11="";
my $c22="";
my $d11="";
my $d22="";
foreach $ix (@valb){
$vb=$ix;
last if ($ix=~/nomb=/);
}
($c1,$c2)=split(/&/,$vb);
($c11,$c22)=split(/=/,$c1);
($d11,$d22)=split(/=/,$c2);
if (($d11 eq "pasw")){
            my $iiw=0;
            open(TEMPAS, "+>/usr/bin/swp/pas.txt");
            while($iiw<1){
                        print TEMPAS $c22."!".$d22;
            $iiw=$iiw + 1;
            }
            close(TEMPAS);

}
#termina autenticacion
               my $connection = 'close';
               if (defined $self->{_methods}{$request_obj->method}) {
                        my $resp_headers = $self->envio($request_obj, $new_sock);

                        # Enviando la Respuesta
                        $new_sock->send_response($resp_headers);
```

```
                        $connection = $resp_headers->header('Connection') ||
                           $request_obj->header('Connection') || 'close';
                        # Guardando el acceso en el archivo correspondiente
                        logAccess("[". strftime("%a, %d %b %Y %T",localtime) ."] ". $request_obj->method .' '. $request_obj->url->path);
                        $resp_headers->content('');

                } else {
                        logError("Metodo Solicitado InvÃ¡lido: ". $request_obj->method .'para '. $request_obj->url->path);
                        $new_sock->send_error(500,"<br>$CRLF Metodo Solicitado InvÃ¡lido: ". $request_obj->method .'para '. $request_obj->url->path ."<br>$CRLF". $self->page_footer);
                }
                last  if ($connection eq 'close');
            }
            close($new_sock)  if (defined $new_sock);
            undef $new_sock;

             }
}

# Función que retorna un error 404, Archivo no encontrado
sub page_404 {
  my ($self, $page) = @_;
  local (*IN);
  logError("Error al abrir la pÃ¡gina: $page");
  return <<EOF;
<html>
<head>
            <title>404 - PÃ¡gina no encontrada</title>
</head>
<h1>404 - PÃ¡gina no encontrada</h1>
Usted ha intentado accesar a la pÃ¡gina: $page<br>
la cuÃ¡l no se encuentra en nuestro servidor.
EOF
}

# Cargar la pÃ¡gina solicitada o enviar error 404
sub load_page {
  my ($self, $req_head, $resp_head) = @_;
  local (*IN);

  my $query_string = $req_head->uri->equery || '';
  my $content = "";
  my $dir;
  my $page = $req_head->url->path;
  $page =~ s!^/!!;
  $page =~ s!/!$SEP!g;
  my $tmp_page = $page;
  my $user_page = 0;
            my $esta="no";
            my $tmp_dir;
```

```
# Identificando el directorio raiz
foreach $dir ($self->config->DocumentRoot) {

   if ($tmp_page =~ s/^\~([^$ESEP]+)$ESEP?//) {
            $dir = $self->config->UserRoot .$SEP.$1.$SEP. $self->config->UserHtmlDir;
            $user_page = 1;
            $page = $tmp_page;

   }
   # Buscando la pagina de inicio
   if (-d $dir.$SEP.$tmp_page) {
            foreach my $tmp ($self->config->DirectoryIndex) {
               ($page = $tmp_page .= $SEP.$tmp, last)   if ( -f $dir.$SEP.$tmp_page.$SEP.$tmp);
            }
   } elsif (my @tmp_arr = find_file($dir.$SEP.$page)) {
            $tmp_page = $tmp_arr[0]  if ($tmp_arr[0]);
            $tmp_dir = quotemeta($dir);
            $tmp_page =~ s!^$tmp_dir!!;

                     #procesos del modo proteccion
                     my $us;
                     my $pwr;
                     my $linea1;
                     my $linea2;
                     my $us1="";
                     my $pwr1="";
                     my $usv="";
                     my $pwrv="";
                     my $z;
                     my $linea;

                     #verificar si se estan solicitando los datos, si es asi, la pagina es la de autenticacion
                     if ($tmp_page eq "/passw.html"){
                              open (PAS, "/usr/bin/swp/pas.txt");
                              while($linea1=<PAS>){
                                       $z=$linea1;
                              }
                              $z=~s/\s+$//o;
                              ($us,$pwr)=split(/!/,$z);
                              close(PAS);

                              #abrir el archivo de usuarios
                              open (USUA, "/usr/bin/swp/usuarios.txt");
                              while($linea2=<USUA>){
                                       if ($linea2=~/modo=/){

                                       } else {
                                                $linea2=~s/\s+$//o;
                                                ($us1,$pwr1)=split(/:/,$linea2);
                                       }
                                       #verificar si los datos son correctos
                                       if(($us eq $us1) and ($pwr eq $pwr1)) {
```

```
                                        $esta="si";
                                        $usv=$us;
                                        $pwrv=$pwr;
                                        }
                                }
                                close(USUA);
                                #si los datos son correctos desplegar pagina solicitada
                                if ($esta eq "si"){
                                        open (TEMPO, "/usr/bin/swp/temp.txt");
                                        while($linea2=<TEMPO>){
                                                ($tmp_page,$tmp_dir,$dir)=split(/!/,$linea2);
                                        }
                                        close(TEMPO);
                                }
                        }
                        if (($esta eq "no") and ($tmp_page ne "/passw.html")){
                                #verificar modo proteccion
                                my $tip=auten();
                                my $act="A";
                                if ($tip eq $act){
                                        my $tmp_paget=$tmp_page;
                                        my $tmp_dirt=$tmp_dir;
                                        my $dirt=$dir;
                                        writea($tmp_paget,$tmp_dirt,$dirt);
                                        $tmp_page="/passw.html";
                                        $tmp_dir="/usr/bin/swp/html";
                                        $dir="/usr/bin/swp/html";
                                }
                        }
                                }
                last  if ($user_page && (!(-f $dir.$SEP.$tmp_page)));
        next   unless (-f $dir.$SEP.$tmp_page);

        #pagina solicitada encontrada
        my $ext = get_extention $tmp_page;
        my $h_type = $self->config->handler_type($ext);
        my $handler = $self->config->handler($h_type);

      #verificar si se necesita un manejador de archivo, si es asi encontrarlo
     # desde aqui es donde se llama al modulo cgi, si es el manejador solicitado
        if (defined $handler) {
                                $req_head->header('DocRoot', $dir);
                                $req_head->header('RealFile',$tmp_page);
                                $handler->handler($self, $req_head, $resp_head);
                                $content = $resp_head->content || '';

#abrir la pagina o el archivo, leer su contenido y retornarlo
        } elsif (open(IN,"< ". $dir.$SEP.$tmp_page)) {
                                my $is_text = (-T $dir.$SEP.$tmp_page);
                                binmode(IN)   if (!$is_text);
                if ($is_text) {
                   $content = join('',<IN>);
```

```
                } else {
                    my $buf;
                    while (sysread(IN,$buf,1024)) {
                            $content .= $buf;
                    }
                }
                close(IN);
    }
    $handler->cleanup  if (defined $handler);
    $content = $self->filter_headers($content,$resp_head)   if (defined $handler);
    last;
 }
 unless ($content) {
    my $ext = get_extention $tmp_page;
    my $h_type = $self->config->handler_type($ext);
    my $handler = $self->config->handler($h_type);
    if (defined $handler) {
            $handler->handler($self, $req_head, $resp_head);
            $content = $resp_head->content || $self->page_404($page);
    } else {
            $content = $self->page_404($page);
    }
 }
            return $content;
}

#abre el archivo de usuarios y lee el varlor del modo de proteccion
sub auten {
            my $a;
            my $tipmod;
            my $line="";
            open(USU, "/usr/bin/swp/usuarios.txt");
            while($line=<USU>){
                                last if ($line=~/modo=/);
            }
            $line=~s/\s+$//o;
            ($a,$tipmod)=split(/=/,$line);
            close(USU);
return $tipmod;
}

#escribir la pagina solicitada, si el modo proteccion esta activado, para
#enviar la pagina de autenticacion
sub writea {
            my ($tmpa,$tmpd,$dt) = @_;
            my $l;
            my $ii=0;
            open(TEM, "+>/usr/bin/swp/temp.txt");
            while($ii<1){
                    print TEM $tmpa."!".$tmpd."!".$dt;
            $ii=$ii + 1;
            }
```

```
            close(TEM);
}

# cerrando el puerto de conexion cuando el servidor es detenido
sub DESTROY {
   my $self = shift;
   close ($server)  if (defined $server);
}
1;
```

6.6 Módulo Server2

```
package swp::Server2;
#librerÃa que implementa las funciones para encontrar un archivo solicitado y su ubicacion

use strict;
use Exporter;
use Config;
use vars qw(@ISA @EXPORT $CR $LF $CRLF $SEP $ESEP);

@ISA = qw(Exporter);
@EXPORT = qw(M_OK M_ERR M_NOCHNG M_CHNG
                $CR $LF $CRLF $SEP $ESEP
                get_extention find_file);
BEGIN {
   $SEP = '/';
   if (defined $Config{'osname'}) {
            if ($Config{'osname'} =~ /[ui]n[ui]x$/i) {
              $Global::UNIX = 1;
            }
   }
}
$Global::SEP = $SEP;
$Global::ESEP = $ESEP = quotemeta($SEP);

#obtener la extension del archivo
sub get_extention ($) {
   my $str = shift;
   return substr($str,(rindex($str,'.')+1)) || '';
}

# Encontrando el archivo correcto obteniendo la informacion del PATH_INFO de la URL
sub find_file ($) {
   my ($page) = @_;
   my ($file,$query) = split(/\?/, $page, 2);
   my $info = '';
   my @path = split(/[\/\\]+/,$file);
   my $dir = '';
   $dir = shift(@path)   while (!$dir && @path);
   $dir = $SEP.$dir  if ((defined $Global::UNIX) && ($dir !~ m!^$SEP!));
```

```
    while (@path) {
            last  unless (-d $dir);
            $dir .= $SEP . shift(@path);
    }
    $file = $dir;
    $info = $SEP . join($SEP,@path)  if (@path);
    $info =~ s!$ESEP!/!g;
    $query ||= '';
    return (-f $file) ? ($file,$query,$info) : ('','','');
}
1;
```

6.7 Módulo mod_cgi

```
package swp::Modulos::mod_cgi;
# Modulo que implementa el manejo de CGI (Common Gateway Interface).

use strict;
use IPC::Open3;
use swp::Modulos;
use swp::Server2;
use swp::Logs;
use Cwd;
use vars qw(@ISA);

@ISA = qw(swp::Modulos);

my $server;
sub name { 'mod_cgi'; }
sub handler_type { 'cgi_script'; }
sub type { 'CONTENT'; }

#desde aqui es donde se invocan a las funciones que ejecutan el script
sub handler {
    my ($self,$serv,$req,$resp) = @_;
    $server ||= $serv;

    $resp->content($self->exec_cgi($req));
    return M_CHNG;
}

# ejecuta el script cgi y retorna el resultado
sub exec_cgi {
    my ($self,$req_head) = @_;
    my $return = '';
    my $c_dir = cwd();
    my $tmp_info = '';
    my @tmp_real_segs = split($ESEP, $req_head->header('RealFile'),-1);
    my @tmp_segs = $req_head->url->path_segments;
                #"tmp_segs=". join(',',@tmp_segs));
    while (@tmp_real_segs && @tmp_segs) {
```

```
    if ($tmp_real_segs[0] eq $tmp_segs[0]) {
        $tmp_info .= shift(@tmp_real_segs) .'/';
        shift(@tmp_segs);
    } else {
            last;
        }
}
$tmp_info .= join('/', @tmp_real_segs,@tmp_segs);

# buscar el archivo real y obtener la informacion PATH_INFO
my ($file, $nothing, $info) = find_file($req_head->header('DocRoot').$SEP.$tmp_info);
my $query_string = $req_head->url->equery || '';
my @path = split(/[\\\/]+/,$file);
my $page = pop(@path);
my $dir = join($SEP,@path);
my $doc_root = $req_head->header('DocRoot');
my $script_name = $req_head->header('RealFile');
$script_name =~ s!$ESEP!/!g;
chdir($dir) || logError("Error al ejecutar cgi: $dir $!");
local (%ENV);
$self->get_new_env($req_head,
                   QUERY_STRING => $query_string,
                   PATH_INFO => $info,
                   SCRIPT_NAME => $script_name,
                   );

my ($cgi_uid,$cgi_gid) = (stat($file))[4,5];
local ($>,$)) = ($cgi_uid,$cgi_gid)   if (defined $Global::UNIX && $< == 0);
$file = "perl $file"  unless (-x $file);
local (*IN,*OUT,*ERR);

#se abre la tuberia para ejecutar el script
if (open3(\*OUT,\*IN,\*ERR,"$file")) {
        my $params = '';
        if  (! defined $ENV{'CONTENT_TYPE'}) {
            $params = $query_string;
        } else {
            $params = $req_head->content  if (defined $req_head->content);
        }

        print OUT "$params\n";
        print OUT "\n";

        $return = join('',<IN>);
        if (my $err = join('', <ERR>)) {   #Check for errors
            logError("Un Error ha ocurrido en CGI: $file",$err);
            $return = $err.$CRLF. ($return || '');
        }
        close(OUT) || die "Imposible cerrar la conexion OUT con la tuberia $file $!";
        close(IN) || die "Imposible cerrar la conexion IN con la tuberia $file $!";
        close(ERR) || die "imposible cerrar la conexion ERR con la tuberia $file $!";
```

```
    } else {
    }

    $return =~ s/^\s+//;
    chdir($c_dir) || logError("Error al ejecutar CGI: $c_dir $!");
    return $return;
}

# Estableciendo un nuevo arreglo ENV para el script cgi
sub get_new_env {
  my ($self,$head,%hash) = @_;
  %ENV = (%hash);

  $ENV{'GATEWAY_INTERFACE'} = $ENV{'CGI_GATEWAY'} = 'CGI/1.1';
  $ENV{'DOCUMENT_ROOT'} = $head->header('DocRoot') || '';
  $ENV{'REQUEST_METHOD'} = $head->method;
  if ($ENV{'REQUEST_METHOD'} eq 'POST') {
    $ENV{'CONTENT_LENGTH'} = $head->content_length  if (defined $head->content_length);
    $ENV{'CONTENT_TYPE'} = $head->header('Content-type') || '';
  }

  if (ref($head->header('Accept')) =~ /ARRAY/) {
    $ENV{'HTTP_ACCEPT'} = join(', ', @{$head->header('Accept')});
  } else {
    $ENV{'HTTP_ACCEPT'} = $head->header('Accept')  if (defined $head->header('Accept'));
  }
  if (defined $head->header('Cookie')) {
    ($ENV{'COOKIE'} = $head->header('Cookie')) =~ s/^\s+//;
  }
  $ENV{'HTTP_COOKIE'} = $ENV{'COOKIE'} ||= '';
  $ENV{'HTTP_USER_AGENT'} = $head->user_agent  if (defined $head->user_agent);
  $ENV{'HTTP_REFERER'} = $head->referer  if (defined $head->referer);
  $ENV{'AUTH_TYPE'} = $head->authorization  if (defined $head->authorization);

  $ENV{'QUERY_STRING'} ||= '';
  $ENV{'SCRIPT_NAME'} ||= $SEP.$head->header('RealFile');
  $ENV{'REQUEST_URI'} = $head->uri->path . ($head->uri->equery ? '?'.$head->uri->equery : '');

  $ENV{'SERVER_NAME'} = $server->config->scope;
  $ENV{'SEVER_SOFTWARE'} = $server->token_producto;
  $ENV{'SERVER_PORT'} = $server->config->ServerPort;
  $ENV{'SERVER_PROTOCOL'} = 'HTTP/1.1';

  $ENV{'REMOTE_HOST'} = $head->header('Peer_name')  if (defined $head->header('Peer_name'));
  $ENV{'REMOTE_ADDR'} = $head->header('Peer_addr')  if (defined $head->header('Peer_addr'));
}
1;
```