DESPLIEGUE DE APLICACIONES WEB

CFGS. Diseño de Aplicaciones Web (DAW). 18-19

IES FRANCESC DE BORJA MOLL

TEMA 2. ADMINISTRACIÓN DE SERVIDORES WEB.

1. FUNCIONAMIENTO DE UN SERVIDOR WEB.

Detrás de cada página web debe existir un servidor web que ofrezca esa página a todo aquel que disponga de una conexión de red con la cual pueda acceder a la página.

La configuración del servidor web dependerá de las páginas web que ofrezca, así la configuración no será la misma si la página posee contenido estático o no, o si se necesita que modifique el contenido según interacción del usuario, o si se necesita de comunicación segura en la transición de información, o si se debe tener en cuenta el control de acceso a determinados sitios de la página.
Por lo tanto según las páginas web que se ofrezcan el servidor web deberá estar configurado para tal fin: con soporte PHP, con soporte de cifrado, con soporte de control de acceso, etc.

Un servidor web puede alojar varias páginas, sitios, dominios de Internet, pero hay que tener en cuenta que la elección del servidor web será muy importante para la configuración y administración de uno o múltiples sitios, ya que: ¿puede el servidor web modularse fácilmente? ¿se le pueden añadir o quitar características?, o por contra si se quiere añadir una funcionalidad que no posea en la instalación base se debe desinstalar e instalar de nuevo, por ejemplo: hasta ahora el servidor web solamente ofrecía páginas estáticas pero si se desea ofrecer también páginas web dinámicas, ¿qué se hace: modular o nueva instalación?

Se ha de tener que pensar que todo puede crecer y lo que ahora era un servidor web que ofrecía x número de páginas se necesita que ofrezca x*y, con ello se tendrá que prever la escalabilidad del servidor web, y también la estabilidad: ¿cómo se comporta ante múltiples conexiones simultáneas?

En la Práctica 2 y 3, aprenderemos a configurar y administrar el servidor Apache, ya que éste soporta:

- páginas web estáticas
- páginas web dinámicas
- hosts virtuales
- seguridad mediante cifrado
- autenticación y control de acceso

- modular y monitorizar archivos de registro, etc.

1.1. SERVICIO DE FICHEROS ESTÁTICOS.

Las páginas web pueden ser de dos tipos: estáticas y dinámicas.

Las páginas estáticas son aquellas que están enfocados principalmente a mostrar una información permanente, donde el navegante se limita a obtener dicha información, sin que pueda interactuar con la página Web visitada. Las Web estáticas están construidas principalmente con hipervínculos o enlaces (links) entre las páginas Web que conforman el sitio, este tipo de Web son incapaces de soportar aplicaciones Web como gestores de bases de datos, foros, consultas on line, e-mails inteligentes...

Las páginas dinámicas son aquellas que permiten crear aplicaciones dentro de la propia Web, otorgando una mayor interactividad con el navegante. Aplicaciones dinámicas como encuestas y votaciones, foros de soporte, libros de visita, envío de e-mails inteligentes, reserva de productos, pedidos on-line, atención al cliente personalizada...

1.2. CONTENIDO DINÁMICO.

Es posible que cuando de nuevo se desea volver ver una página resulta que ésta ha cambiado, por lo tanto, este sería el caso de una página dinámica.

Puede ocurrir que al acceder a una página web, dependiendo si posee una cuenta de usuario u otra, que el contenido pueda ser distinto, o que el contenido varíe dependiendo del navegador, etc. Ante un caso como éste, también se estaría hablando de una página dinámica.

Una página dinámica, necesita más recursos del servidor web que una página estática, ya que consume más tiempo de CPU y más memoria que una página estática. Además la configuración y administración del servidor web será más compleja: cuántos más módulos tengamos que soportar, más tendremos que configurar y actualizar. Esto también tendrá una gran repercusión en la seguridad del servidor web: cuántos más módulos más posibilidades de problemas de seguridad, así si la página web dinámica necesita, para ser ofrecida, de ejecución en el servidor se debe controlar que es lo que se ejecuta.

Algunos módulos con los que trabaja el servidor web Apache para poder soportar páginas dinámicas son: mod_actions, mod_cgi, mod_cgid, mod_ext_filter, mod_include, mod_ldap, mod_perl, mod_php5, mod_python.

mod_actions: Este módulo ofrece ejecutar scripts CGI basado en el tipo de medio o método de solicitud.

mod_cgi: permite ejecutar scripts de cgi.

mod_cgid: La ejecución de scripts CGI utilizando un demonio CGI externo.

mod_ext_filter: Pase el cuerpo de la respuesta a través de un programa externo antes de la entrega al cliente.

Para averiguar más sobre los módulos que proporciona Apache:

http://httpd.apache.org/docs/current/mod/

Para averiguar más sobre como configurar apache:

http://httpd.apache.org/docs/current/howto/auth.html

1.3. LA URL

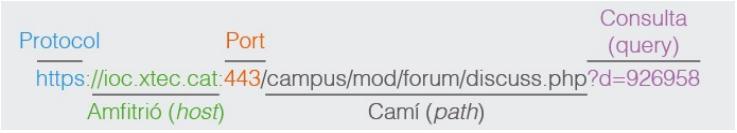
Antes de comenzar con la instalación, configuración y administración de servidores web es bueno conocer las cuestiones fundamentales del protocolo HTTP.

El protocolo de transferencia de hipertexto o HTTP (HyperText Transfer Protocol) establece el protocolo para el intercambio de documentos de hipertexto y multimedia en la web. El HTTP dispone de una variante cifrada mediante SSL llamada HTTPS. Teniendo en cuenta que HTTP es el protocolo utilizado en las comunicaciones web, conocer los aspectos básicos de este protocolo ayuda a entender algunas de las cuestiones que tienen que ver con la configuración y administración de servidores web.

Los **localizadores uniformes de recursos** (URL, Uniform Resource Locator) son el mecanismo que permite, como indica su nombre, localizar recursos en internet mediante los diferentes protocolos disponibles.

Existe también el concepto de identidad cador uniforme de recursos (URI, Uniform Resource Identifier) que, a la práctica, se considera equivalente el concepto de URL, ya que un URI equivale el conjunto de una URL más un nombre uniforme de recursos (URN, Uniform Resource Name) como el concepto URN en la práctica no se usa, los términos URL y URI son equivalentes e intercambiables en la mayor parte de los casos.

Un URL está formado por varios elementos cada uno de los cuales nos describe un aspecto diferente sobre la localización de un recurso los elementos en el caso los URL utilizados en los protocolos HTTP / HTTPS son los que presenta a continuación:



- Esquema (scheme): indica el protocolo que se utilizará para acceder al recurso especificado por el resto de la URL. Según el protocolo indicado (HTTP y HTTPS), el resto de la URL puede tener diferencias en su estructura.
- Hospedajes (host): identifica el servidor web. Puede ser un nombre de dominio o una dirección IP.
- Puerto: se trata de un elemento opcional que sirve para indicar qué puerto TCP / IP se utilizará para establecer la conexión para acceder al recurso. Con protocolo HTTP, si no se indica, coge por defecto el puerto 80 y con HTTPS utiliza por defecto el puerto 443.
- Camino (path): indica la localización del recurso dentro del servidor. El camino asignado en una URL no tiene porqué representar un camino físico de disco con el mismo nombre de directorios ya que los servidores web permiten hacer un mapeo entre caminos de URL y directorios de disco (directorios virtuales).
- Consulta (query): permite pasar parámetros adicionales útiles especialmente cuando el recurso al que accedemos es un script u otro tipo de elemento que ejecuta código en el servidor. Está formado por pares "nombre = valor", los que, en caso de haber más de uno, se separan con el carácter &.

Muchos de los elementos de una URL son opcionales y se pueden omitir si el contexto donde se utiliza la URL permite asumir unos valores por defecto.

Un ejemplo es en la barra de direcciones de un navegador, donde no es necesario escribir "Http: '/' /" al inicio de la URL, ya que es el valor por defecto para el navegador.

También ocurre en URL para hacer referencia a recursos o enlaces dentro de un documento HTML. En este caso, además, se puede omitir el nombre de host y, incluso, una parte del camino de los URL internos del documento HTML.

Podemos diferenciar entre:

URL absoluto

URL que incluye el nombre de Hospedajes (host) y el camino (path) completo del recurso. Algunos ejemplos de URL absolutos son:
https://fbmoll/wiki/Localitzador_uniforme_de_recursos

https://fbmoll/educacio/

URL relativo

URL que no incluye esquema (Scheme), el host y el puerto cuando un URL es relativo se suele llamar camino (path) y se puede dividir en caminos completos y caminos relativos.

- Caminos completos (hoja paths): comienzan siempre con el carácter / y especifican toda la secuencia de directorios que hay que recorrer hasta llegar al recurso dentro del mismo servidor de donde se ha descargado el documento HTML, partiendo del directorio raíz del sitio web. Por ejemplo, /index.html.

- Caminos relativos (relative paths): comienzan con un carácter diferente de / e indican la secuencia de directorios que hay que recorrer para llegar al recurso dentro del mismo servidor de donde se ha descargado el documento HTML, partiendo del directorio de donde se ha descargado el documento. Por ejemplo,

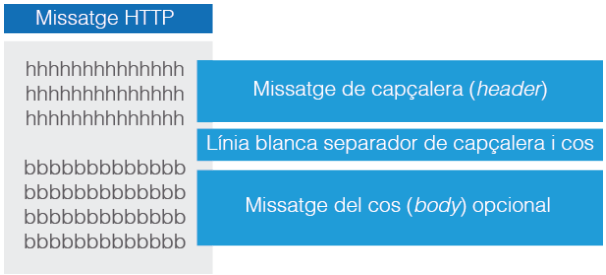
.../fbmoll/images / hdr_bg.gif.

1.4. PROTOCOLO HTTP Y HTTPS.

HTTP es un protocolo de petición / respuesta (request - response). El ciclo de comunicación entre el cliente y el servidor se basa en dos pasos donde la respuesta por parte del servidor viene precedida de una petición por parte del cliente.

Aunque las peticiones y las respuestas contienen información diferente, ambas tienen una misma estructura formada por una cabecera y un cuerpo. La cabecera contiene información sobre el mensaje (metadatos) y el cuerpo es el que lleva el contenido del mensaje. El contenido de las peticiones y respuestas puede estar vacío en algunas ocasiones, por lo tanto, puede haber peticiones o respuestas sin contenido, sólo con cabecera.

Los mensajes HTTP, tanto si son peticiones como respuestas, tienen una estructura formada por una cabecera y un cuerpo. La cabecera está formada por información alfanumérica y está separada del cuerpo por una línea en blanco. El contenido del cuerpo (que puede ser opcional) es alfanumérico o binario.



Las peticiones HTTP, en la primera línea de la cabecera, contienen la línea de petición (request line). El resto de líneas (opcionales) contienen las cabeceras de petición (request headers).

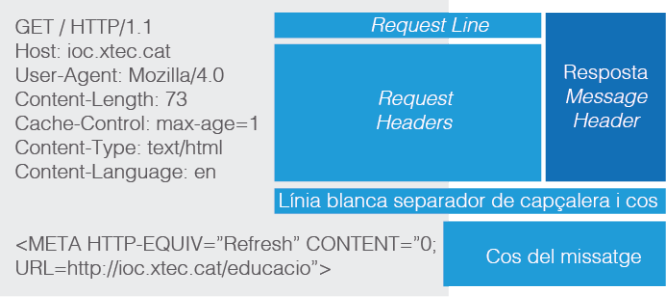
- Línea de petición: primera línea de la cabecera de una petición. Está formada por tres campos:

- Método de petición (request method): puede ser cualquiera de los métodos de petición que define el protocolo HTTP.

- URL de petición (request URL): es la URL del recurso que se pide.

- Versión HTTP: es la versión de protocolo HTTP que se utilizará (HTTP / 1.0 o HTTP / 1.1).

• Cabecera de petición: cada cabecera en una línea diferente y está formada por un par "Nombre: valor ". Si hay varios valores, estos se separan con comas.



Una petición hecha con el método POST oculta los parámetros, pero esto no implica que sea un mecanismo de seguridad. Si el protocolo utilizado es HTTP, la información se envía sin cifrar a través de red y puede ser interceptada durante el proceso de envío.

El protocolo HTTP tiene definidos numerosos métodos de petición (request methods), ya que actualmente estamos en la versión 2.0 y el protocolo ha ido evolucionando y por tanto, incorporando nuevos métodos. Los métodos son lo primero que se especifica en la línea de las peticiones. Los 8 métodos más destacados, son:

1. **GET**: el método GET se utiliza para hacer la petición de un recurso o documento en el servidor (un documento HTML, una imagen ...), especificado en el campo URL la línea de petición (request line). Las peticiones realizadas con el método GET sólo deberían hacer que el servidor vuelva algún tipo de respuesta sin que ello provoque la modificación de los datos del servidor ni de la aplicación web sobre el que se ha ejecutado con el GET.

2. **HEAD**: el método HEAD se usa para simular una petición de un recurso en el servidor. Funciona exactamente igual que el GET, con la diferencia que el servidor sólo responde con las cabeceras de la respuesta (no envía el contenido). Se utiliza por poder hacer un pronóstico de respuesta de una hipotética petición GET. Es empleado habitualmente por los navegadores para comprobar las cabeceras de un recurso determinado y así decidir si es necesario descargarlo o no en caso de tener una copia en la memoria cache local (las cabeceras nos pueden decir el tamaño del documento, la fecha de última modificación, etc.).

3. **POST**: es el método habitual para enviar los datos de formularios HTML que pueden provocar modificaciones a los datos del servidor o de la aplicación web. **Puede parecer igual que el método GET, pero tiene algunas diferencias importantes**:

- Los parámetros de un GET van concatenados a la URL de la petición y es visible en la barra de direcciones del navegador. En cambio con POST adjuntan como contenido del cuerpo de la petición y, por tanto, no son tan visibles. Como consecuencia, las peticiones GET se pueden

memorizar en los marcadores del navegador y se pueden representar, también, como URL en un enlace dentro de un documento HTML. Con POST, esto no es posible.

- Las peticiones GET se deben poder ejecutar tantas veces como se quiera sin que se produzcan modificaciones del estado o los datos del servidor. Por tanto, los navegadores permiten la ejecución repetida de peticiones GET.

Por ejemplo, actualizando una página cargada con GET o tirando atrás a una página previamente cargada con GET sin dar aviso al usuario. En cambio, con el POST es al revés. En las peticiones POST se consideran que pueden ser modificaciones, por ello, los navegadores, cada vez que se pretende repetir una petición POST, informan al usuario con un mensaje de confirmación.

4. **PUT**: el método PUT se utiliza para poder crear o reemplazar un determinado recurso o documento del servidor. Con el método PUT podemos solicitar que el contenido presente en el cuerpo de la petición se almacene en el servidor y pase a estar accesible a través de un URL que indicamos a la línea de petición. Si ya existe, se sustituiría el actual por lo que damos con el PUT. Si no existe, se crearía uno nuevo. La utilización de este método está restringido en la configuración por defecto los servidores web ya que permitiría la modificación no autorizada de sus contenidos (Si lo probamos, podemos recibir un error del tipo "405 Method Not Allowed ").

5. **DELETE**: el método DELETE se utiliza para poder eliminar un determinado recurso o documento del servidor. Con el método DELETE se puede solicitar al servidor que elimine el archivo asociado a un URL del recurso indicado a la línea de petición. La utilización de este método está restringida en la configuración para defecto los servidores web ya que permitiría la modificación no autorizada de sus contenidos (si lo probamos, podemos recibir un error del tipo "405 Method Not Allowed ").

6. **CONNECT**: este método se utiliza para poder pasar conexiones seguras SSL a través de conexiones HTTP y para poder gestionar conexiones HTTP a través de proxy (Proxies).

7. **OPTIONS**: el método OPTIONS pide al servidor qué métodos HTTP se pueden utilizar sobre el recurso identificado con una URL de la línea de petición.

8. **TRACE**: el método TRACE pide al servidor que devuelva una copia de las cabeceras de la petición tal como las ha recibido. Este método HTTP suele estar desactivado por defecto.

El protocolo **HTTP define 5 códigos de estado (status codes)** que permiten indicar varios tipo de situaciones que se pueden dar como a respuesta a una petición de un cliente.

- 1xx: son de tipo informativo, informan el cliente que la petición ha estado recibida y que el servidor continúa procesando la respuesta.
- 2xx: indican la petición ha sido correcta y se ha procesado satisfactoriamente.

- 3xx: indican alguna forma de redirección. Con un código de esta serie se da a entender el cliente que la petición es correcta pero que la respuesta se debe obtener de algún otro lugar.
- 4xx: indican que ha habido un error en el procesamiento de la petición para que el cliente ha hecho algo mal, el error ha sido causado por el cliente.
- 5xx: indican que ha habido un error en el procesamiento de la petición debido a un fallo en el servidor, el error ha sido causado por el servidor.

El **protocolo HTTPS** permite que la información viaje de forma segura entre el cliente y el servidor. Sin embargo, el protocolo HTTP envía la información sin cifrar, esto significa que cualquiera que accediese a la información transferida entre el cliente y el servidor puede ver el contenido exacto y textual de la información.

Para asegurar la información, el protocolo HTTPS requiere de certificados, siempre y cuando sean validados, la información será transferida cifrada. Pero cifrar la información requiere un tiempo de computación, por lo que será perjudicial para el rendimiento del servidor web. Así, ¿es necesario que toda la información sea transferida entre el cliente y servidor de forma cifrada?

A lo mejor solamente es necesario que sea cifrada la autenticación a dicha información, por eso en algunas páginas web puede que el servidor esté configurado para que en todo el dominio esté cifrada su información o simplemente el intento de acceso a la misma.

Un servidor web, como Apache, puede emitir certificados, pero puede que en algún navegador sea interpretado como peligroso, esto suele ser debido a que los navegadores poseen en su configuración una lista de Entidades Certificadoras que verifican, autentican y dan validez a los certificados. Los navegadores, solamente confían en quien confían. Eso no quiere decir que no se pueda crear certificados en un servidor web.

El protocolo HTTPS utiliza cifrado sobre SSL/TLS que proporcionan autenticación y privacidad. Entonces, si se necesita que la información viaje cifrada se debe emplear el protocolo HTTPS, en caso contrario el protocolo HTTP. La utilización del protocolo HTTPS no excluye ni impide el protocolo HTTP, los dos pueden convivir en un mismo dominio.

¿Cómo funcionan?

En el protocolo HTTP cuando se escribe una dirección URL en el navegador, por ejemplo http://www.debian.org/index.es.html, antes de ver la página en el navegador lo que ocurre es lo siguiente: se traduce el dominio DNS por una IP, una vez obtenida la IP se busca en ella si un servidor web aloja la página solicitada en el **puerto 80, puerto TCP asignado por defecto al protocolo HTTP**. Si el servidor web aloja la página ésta será transferida al navegador.

Sin embargo cuando se escribe en el navegador una dirección URL con llamada al protocolo HTTPS, el procedimiento es similar al anterior pero algo más complejo, así se traduce el dominio DNS por una IP. Con la IP se busca el servidor web que aloja la página solicitada **en el puerto 443**, puerto TCP asignado por defecto al protocolo HTTPS, pero ahora antes de transferir la página a tu navegador se inicia una negociación SSL, en la que entre otras cosas el servidor envía su certificado. Si el certificado es firmado por un Entidad Certificadora de confianza se acepta el certificado y se cifra la comunicación con él, transfiriendo así la página web de forma cifrada.

Se puede hacer que un servidor web, para una determinada página, espere los protocolos HTTP y HTTPS en puertos TCP distintos del 80 y 443 respectivamente. Eso sí, cuando se visite la página web, en la dirección URL se debe especificar el puerto TCP, por ejemplo: http://www.tupagina.local:8080, de esta forma el servidor web espera la petición de la página www.tupagina.local en el puerto 8080; del mismo modo en la dirección URL: https://www.tupagina.local:4333 espera la petición de la página www.tupagina.local en el puerto 443. Como se puede observar, se pueden configurar los puertos, pero se ha de tener en cuenta que cualquiera que quisiera acceder a esas páginas debería saber el puerto TCP de la solicitud. Entonces, aunque no se escriba el puerto TCP en las direcciones URL, éstas se interpretarán en el puerto 80 y 443 para el protocolo http y HTTPS respectivamente. Es decir, sería lo mismo escribir http://www.tupagina.local:80 que http://www.tupagina.local y lo mismo escribir https://www.tupagina.local:443 que https://www.tupagina.local.

1.5. TIPOS MIME.

¿Cómo se transmite un vídeo por Internet, con qué codificación?

¿Cómo sabe un navegador que al seguir un enlace de vídeo el programa que debe utilizar para reproducirlo?

El estándar Extensiones Multipropósito de Correo de Internet o MIME (Multipurpose Internet Mail Extensions), **especifica cómo un programa debe transferir archivos de texto, imagen, audio, vídeo o cualquier archivo que no esté codificado en US-ASCII**. MIME está especificado en seis RFC (Request for Comments//Petición de Comentarios):

- RFC2045
- RFC2046
- RFC2047
- RFC4288
- RFC4289
- RFC2077

Cada uno de estos RFC son notas técnicas que especifican el funcionamiento y organización sobre Internet. Cubren aspectos de las redes de computadoras, incluidos los protocolos, procedimientos, programas, presentación de documentos a través de la red, A etc.

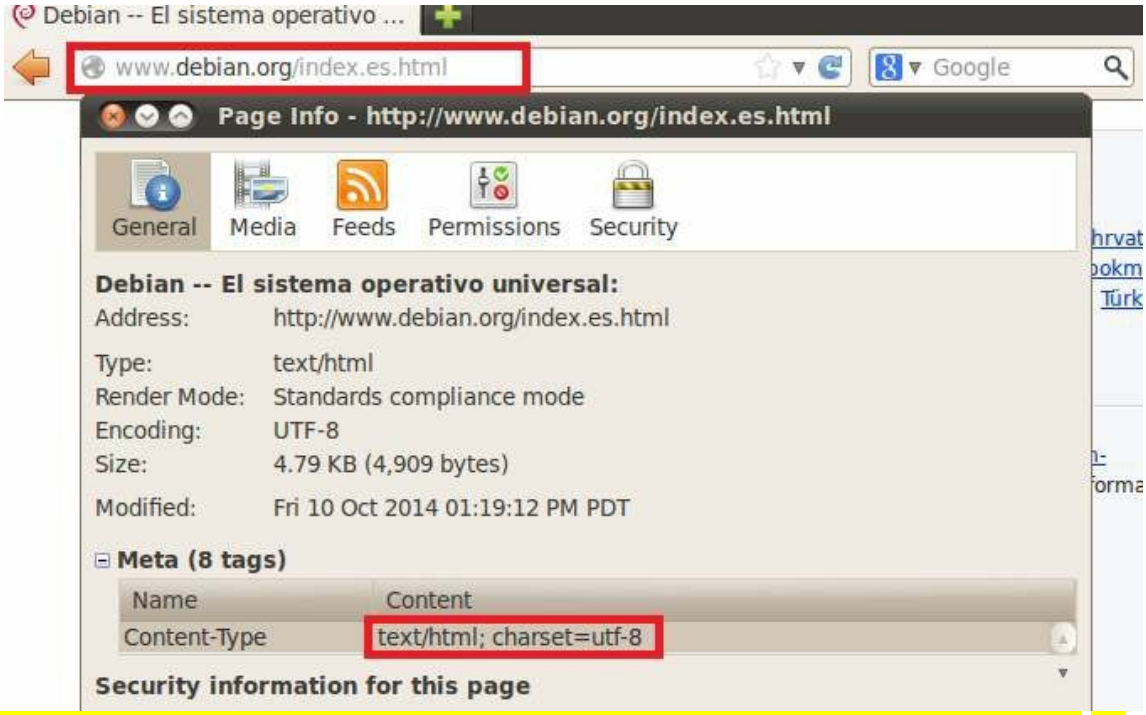
¿Cómo funciona?:

Ejemplo: transferencia de una página web.

Cuando un navegador intenta abrir un archivo, el estándar MIME le permite saber con qué tipo de archivo está trabajando para que el programa asociado pueda abrirlo correctamente. Si el archivo no tiene un tipo MIME especificado, el programa asociado puede suponer el tipo de archivo mediante la extensión del mismo, por ejemplo: un archivo con extensión .txt supone contener un archivo de texto.

Para realizar esto último, el navegador solicita la página web, y el servidor antes de transferirla confirma que la petición requerida existe y el tipo de datos que contiene. Esto último, mediante referencia al tipo MIME al que corresponde. Este diálogo es parte de las cabeceras http.

En ese diálogo, en las cabeceras respuestas del servidor existe el campo Content-Type, donde el servidor avisa del tipo MIME de la página. Con esta información, el navegador sabe como debe presentar los datos que recibe. Por ejemplo si se visita http://www.debian.org/index.es.html se puede ver como respuesta en la cabecera del servidor el campo Content-Type: text/html, indicando que el contenido de la página web es tipo texto/html.



Cada identificador de tipo MIME consta de dos partes. La primera parte indica la categoría general a la que pertenece el archivo como, por ejemplo, "text". La segunda parte del identificador detalla el tipo de archivo específico como, por ejemplo, "html". Un identificador de tipo MIME "text/html", por ejemplo, indica que el archivo es una página web estándar.

Los tipos MIME pueden indicarse en tres lugares distintos: el servidor web, la propia página web y el navegador.

- El servidor debe estar capacitado y habilitado para manejar diversos tipos MIME.
- En el código de la página web se referencia tipos MIME constantemente en etiquetas link, script, object, form, meta, así por ejemplo:
- El enlace a un archivo hoja de estilo CSS:

```
<link href="./miarchivo.css" rel="stylesheet" type="text/css">
```

- El enlace a un archivo código javascript:

```
<script language="JavaScript" type="text/javascript" src="scripts/mijavascript.js">
```

- Con las etiquetas meta podemos hacer que la página participe en el diálogo servidor-cliente, especificando datos MIME:

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
```

- El navegador del cliente también participa, además de estar capacitado para interpretar el concreto tipo MIME que el servidor le envía, también puede, en el diálogo previo al envío de datos, informar que tipos MIME puede aceptar la cabecera http_accept, así por ejemplo una cabecera http_accept tipo de un navegador sería: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

El valor */* significa que el navegador aceptará cualquier tipo MIME.

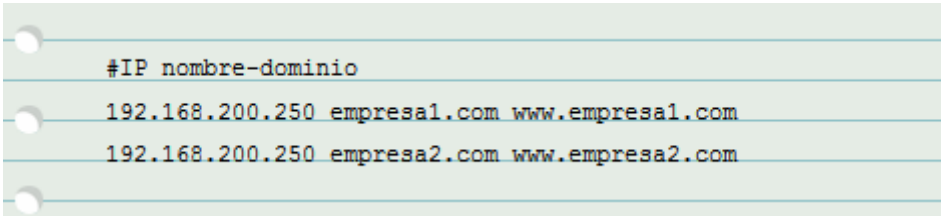
2. HOSTS VIRTUALES. CREACIÓN, CONFIGURACIÓN Y UTILIZACIÓN.

Un mismo servidor web puede alojar páginas de distintos dominios mediante la configuración de hosts virtuales o virtualhosts. Éstos **permiten que un servidor pueda alojar múltiples dominios**. Cada virtual host es único e independiente de los demás.

No obstante, todo aquello que no esté incluido en la definición de cada virtualhost se heredará de la configuración principal: httpd.conf, así si se quiere definir una directiva común en todos los virtualhost no se debe modificar cada uno de ellos introduciendo esa directiva sino que se debe definir esa directiva en la configuración principal del servidor web Apache, de tal forma que todos los virtualhost heredarán esa directiva.

Existen tres tipos de virtualhost: basados en nombre, basados en IP y basados en varios servidores principales.

Para configurar un servidor DNS para poner entradas de dominio que puedan ser utilizadas, se pueden generar estas entradas modificando el archivo /etc/hosts, añadiendo al final del mismo la IP y su nombre correspondiente. Ej:



2.1. VIRTUALHOSTS BASADOS EN NOMBRE.

Supongamos que se quiere realizar dos virtualhosts para dos empresas: empresa1 y empresa2.

La IP que debemos poner siempre en la definición de la directiva Virtualhost es la IP del servidor web, en nuestro caso: IP_Servidor_Web=192.168.200.250

Pasos a seguir para crear virtualhosts basados en nombre:

http://httpd.apache.org/docs/current/vhosts/name-based.html

2.2. VIRTUALHOSTS BASADOS EN IP.

La IP que se debe poner ahora en la definición de la directiva Virtualhost cambia, cada IP corresponde a una interfaz de red del servidor web, p.e.: IP1_Servidor_Web=192.168.200.250, IP2_Servidor_Web=192.168.200.251

Este método no aporta ventajas sobre el anterior, es más, aún puede ser más difícil de mantener si las IP del servidor web se modifican con cierta frecuencia.

Pasos a realizar para configurar un virtualhost basado en IP (parecido al caso anterior):

http://httpd.apache.org/docs/current/vhosts/ip-based.html

3. MÓDULOS.

La importancia de un servidor web radica en su:

- Estabilidad en el servicios ofrecido por el servidor.
- Disponibilidad del servicios ofrecido por el servidor.
- Escalabilidad, aumento de aplicaciones y posibilidades de mejora.

Es importante poder dotar al servidor web de nuevas funcionalidades de forma sencilla, así como del mismo modo quitárselas. Es por esto que la posibilidad que otorga el servidor web Apache mediante sus módulos sea uno de los servidores web más manejables y potentes que existen: si se necesita soporte SSL pues podemos utilizar su módulo SSL, que se necesita soporte PHP pues activamos su módulo PHP, que se necesita soporte LDAP pues módulo LDAP...

3.1. OPERACIONES SOBRE MÓDULOS.

Los módulos de Apache se pueden instalar, desinstalar, habilitar o deshabilitar, así, se puede tener un módulo instalado pero no habilitado. Esto quiere decir que aunque se instalen módulos, éstos no funcionarán hasta que se habiliten.

En la tabla siguiente se muestra un resumen de operaciones, ejemplos y comandos necesarios que se le pueden realizar a los módulos:

Operaciones sobre módulos Apache en un en un GNU/Linux Debian	
Instalar un módulo	**Ejemplo: Instalar el módulo ssl**
`apt-get install nombre-modulo`	`apt-get install libapache2-mod-gnutls`
Desinstalar un módulo	**Ejemplo: Desinstalar el módulo ssl**
`apt-get remove nombre-modulo`	`apt-get remove libapache2-mod-gnutls`
Habilitar un módulo	**Ejemplo: Habilitar el módulo ssl**
`a2enmod nombre-modulo-apache`	`a2enmod ssl`
Deshabilitar un módulo	**Ejemplo: Deshabilitar el módulo ssl**
`a2dismod nombre-modulo-apache`	`a2dismod ssl`

Para habilitar un módulo Apache, en Debian, también se puede ejecutar el comando a2enmod sin parámetros. La ejecución de este comando ofrecerá una lista de módulos a habilitar. Si se escribe el módulo en cuestión, éste quedará habilitado. Del mismo modo para deshabilitar un módulo Apache, en Debian, se puede ejecutar el comando a2dismod sin parámetros. La ejecución de este comando ofrecerá una lista de módulos a deshabilitar, se escribe el módulo en cuestión y el módulo se deshabilitará.

4. ACCESO A CARPETAS SEGURAS.

Existe la posibilidad de asegurar la información sensible que viaja entre el navegador y el servidor, pero esto repercutirá en un mayor consumo de recursos del servidor, puesto que asegurar la información implica en que ésta debe ser cifrada, lo que significa computación algorítmica.

Este cifrado es el cifrado de clave pública o asimétrico: clave pública (kpub) y clave privada (kpriv). La kpub interesa publicarla para que llegue a ser conocida por cualquiera, la kpriv no interesa que nadie la posea, solo el propietario de la misma. Ambas son necesarias para que la comunicación sea posible, una sin la otra no tiene sentido, así una información cifrada mediante la kpub solamente puede ser descifrada mediante la kpriv y una información cifrada mediante la kpriv solo puede ser descifrada mediante la kpub.

En el cifrado asimétrico se puede estar hablando de individuos o de máquinas, en nuestro caso hablamos de máquinas y de flujo de información entre el navegador (A) y el servidor web (B). Ver la siguiente tabla como ejemplo de funcionamiento del cifrado asimétrico:

A(inf) → inf cifrada → B [descifrar inf] → B(inf) = A(inf)
A(inf) → inf cifrada = [(inf)]kpubB → B [inf. cifrada]kprivB → B(inf) = A(inf)

Identificación	
A	Navegador web.
inf cifrada = [(inf)]kpubB	Información cifrada mediante la clave pública de B obtenida a través de un certificado digital.
[inf. cifrada]kprivB	Información descifrada mediante la clave privada de B.
B	Servidor web.

Como se puede apreciar, A envía la información cifrada mediante la kpubB y B la descifra mediante su clave privada (kprivB), por lo que se garantiza la confidencialidad de la información. Pero, ¿estás seguro que B es quién dice que es? ¿Es quién debe ser? ¿Cómo garantizar la autenticidad de B? Pues ya que se supone que B es quien dice ser mediante un certificado digital, se debe confiar en ese certificado.

En Internet existen autoridades de certificación (CA ó AC) que aseguran la autenticidad del certificado digital. El Servidor Web Apache permite ser CA, por lo que tiene la posibilidad de crear sus propios certificados digitales, ahora bien, es posible que el navegador web (A) no confíe en estos certificados y, por tanto avisará que la página a la cuál se intenta acceder en el servidor web representa un peligro de seguridad, ya que no existe en su lista de autoridades certificadoras de confianza.

4.1. CERTIFICADOS DIGITALES, AC Y PKI.

Un certificado digital es un documento electrónico que asocia una clave pública con la identidad de su propietario, y es emitido por autoridades en las que pueden confiar los usuarios. Éstas certifican el documento de asociación entre clave pública e identidad de un individuo o máquina (servidor web) firmando dicho documento con su clave privada, esto es, mediante firma digital. La idea consiste en que los dos extremos de una comunicación, por ejemplo cliente (navegador web) y servidor (servidor web Apache) puedan confiar directamente entre sí, si ambos tienen relación con una tercera parte, que da fe de la fiabilidad de los dos, aunque en la práctica suele interesar solamente la fiabilidad del servidor, para saber que se conecta con el servidor que se desea y no con otro servidor.

Así, la necesidad de una Tercera Parte Confiable (TPC ó TTP, Trusted Third Party) es fundamental en cualquier entorno de clave pública. La forma en que esa tercera parte avalará que el certificado es de fiar es mediante su firma digital sobre el certificado. Por tanto, podremos confiar en cualquier certificado digital firmado por una tercera parte en la que confiamos.

La TPC que se encarga de la firma digital de los certificados de los usuarios de un entorno de clave pública se conoce con el nombre de Autoridad de Certificación (AC).

El modelo de confianza basado en Terceras Partes Confiables es la base de la definición de las Infraestructuras de Clave Pública (ICP o PKIs, Public Key Infrastructures). Una Infraestructura de Clave Pública es un conjunto de protocolos, servicios y estándares que soportan aplicaciones basadas en criptografía de clave pública.

Algunos de los servicios ofrecidos por una ICP (PKI) son los siguientes:

- Registro de claves: emisión de un nuevo certificado para una clave pública.
- Revocación de certificados: cancelación de un certificado previamente emitido.
- Selección de claves: publicación de la clave pública de los usuarios.
- Evaluación de la confianza: determinación sobre si un certificado es válido y qué operaciones están permitidas para dicho certificado.
- Recuperación de claves: posibilidad de recuperar las claves de un usuario.

Las ICP (PKI) están compuestas por:

- Autoridad de Certificación (AC): realiza la firma de los certificados con su clave privada y gestiona la lista de certificados revocados.
- Autoridad de Registro (AR): es la interfaz hacia el mundo exterior. Recibe las solicitudes de los certificados y revocaciones, comprueba los datos de los sujetos que hacen las peticiones y traslada los certificados y revocaciones a la AC para que los firme.

4.2. MÓDULO SSL PARA APACHE.

El método de cifrado SSL/TLS utiliza un método de cifrado de clave pública para la autenticación del servidor.

El módulo ssl es quien permite cifrar la información entre navegador y servidor web. Si se desea saber más acerca del módulo SSL:

http://httpd.apache.org/docs/current/ssl/

5. AUTENTICACIÓN Y CONTROL DE ACCESO.

Puede que interese impedir el acceso a determinadas páginas ofrecidas por el servidor web, así. Para este tipo de casos se tiene que pensar en la autenticación y el control de acceso.

Cuando se autentica en una web se suele transferir la información de autenticación a una base de datos, que puede existir en la misma máquina que el servidor web o en otra totalmente diferente. Suelen emplearse bases de datos SQL o LDAP para la autenticación de usuarios, siendo OpenLDAP una de las alternativas más empleadas.

HTTP proporciona un método de autenticación básico de usuarios: basic. Este método ante una petición del cliente (navegador web) al servidor cuando se solicita una URL, mostrará un diálogo pidiendo usuario y contraseña. Una vez autenticado el usuario, el cliente volverá a hacer la petición al servidor pero ahora enviando el usuario y contraseña, en texto claro (sin cifrar) proporcionados en el diálogo. Es recomendable entonces, si se emplea este método, que se haga combinado con conexión SSL (HTTPS).

5.1. AUTENTICAR USUARIOS EN APACHE MEDIANTE LDAP.

Para que Apache pueda autenticar usuarios mediante LDAP, se ha de tener habilitados los módulos: ldap y authnz_ldap.

Una vez que se haya instalado OpenLDAP (ver por internet manual para instalarlo), se procede a configurar la autenticación de usuarios mediante LDAP:

1. Habilitar el soporte LDAP para Apache2:

```
a2enmod authnz_ldap
/etc/init.d/apache2 restart
```

2. Configurar el virtualhost autenticacion-ldap-apache como sigue:

```
<VirtualHost *:80>
        DocumentRoot /var/www/autenticacion-ldap
        ServerName www.empresa-proyecto.panel-de-control.com
        ServerAlias www.autenticacion-ldap.empresa-proyecto.com
        <Directory /var/www/autenticacion-ldap>
                AllowOverride All
        </Directory>
        ErrorLog /var/log/apache2/error-autenticacion-ldap.log
        LogLevel warn
        CustomLog /var/log/apache2/access-autenticacion-ldap.log combined
</VirtualHost>
```

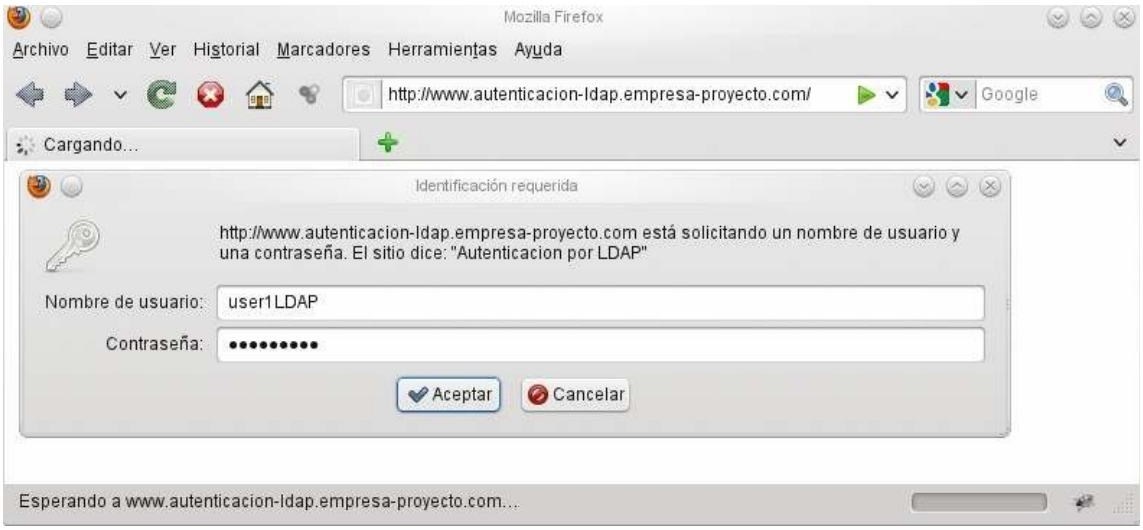
La directiva AllowOverride All es necesaria para habilitar ficheros .htaccess.

3. Crear el fichero /var/www/autenticacion-ldap/.htaccess que permite configurar la autenticación ldap para el virtualhost anterior:

```
AuthName "Autenticacion por LDAP"
AuthType Basic
AuthBasicProvider ldap
AuthzLDAPAuthoritative on
AuthLDAPUrl ldap://127.0.0.1/ou=usuarios,dc=proyecto,dc=com?uid
Require ldap-user user1LDAP
```

La directiva Require ldap-user admin permite la autenticación al usuario user1LDAP, todos los demás usuarios tienen el acceso denegado.

4. Accede a la URL: www.empresa-proyecto.panel-de-control.com ó www.autenticacion-ldap.empresa-proyecto.com



6. MONITORIZACIÓN DEL ACCESO: ARCHIVOS DE REGISTRO (LOGS).

Tan importante como es configurar un servidor web lo es mantener y comprobar su correcto funcionamiento, y para ello se debe de ayudar de los logs o archivos de registro que permiten revisar y estudiar su funcionamiento.

Apache permite mediante diversas directivas crear archivos de registro que guardarán la información correspondiente a las conexiones con el servidor. Esta información es guardada en formato CLF (Common Logon Format) por defecto. Ésta es una especificación utilizada por los servidores web para hacer que el análisis de registro entre servidores sea mucho más sencillo, de tal forma que independientemente del servidor web utilizado se pueda emplear el mismo método de análisis de registro, ya sea mediante lectura, mediante programas ejecutables (scripts) o mediante programas propios de análisis de registro.

En un archivo de registro en formato CLF cada línea identifica una solicitud al servidor web. Esta línea contiene varios campos separados con espacios. Cada campo sin valor es identificado con un guión (-). Los campos empleados en una configuración por defecto de Apache2 son los definidos en la siguiente tabla:

Ejemplo log Apache en formato CLF		
192.168.200.100 - - [05/May/2011:17:19:18 +0200] "GET /index.html HTTP/1.1" 200 20		
Campos (especificadores)	**Definición**	**Ejemplo**
host (%h)	Identifica el equipo cliente que solicita la información en el navegador.	192.168.200.100
ident (%l)	Información del cliente cuando la máquina de éste ejecuta identd y la directiva IdentityCheck está activada.	
authuser (%u)	Nombre de usuario en caso que la URL solicitada requiera autenticación HTTP.	
date (%t)	Fecha y hora en el que se produce la solicitud al servidor. Va encerrado entre corchetes. Este campo tiene su propio formato: [dia/mes /año:hora:minuto:segundo zona]	[05/May/2011:17:19:18 +0200]
request (%r)	Petición del cliente, esto es, la página web que está solicitando. En el ejemplo: /index.html, esto es, dentro de la raíz del dominio que se visite la página	/index.html
status (%s ó %>s)	Identifica el código de estado HTTP de tres dígitos que se devuelve al cliente.	200
Bytes (%b)	Sin tener en cuenta las cabeceras HTTP el número de bytes devueltos al cliente.	20

Cada campo tiene su especificador, el cual se emplea en las directivas de Apache para indicar que campo queremos registrar.

6.1. DIRECTIVAS PARA ARCHIVOS DE REGISTRO.

El contexto de aplicación de todas las directivas que se indican a continuación en la siguiente tabla puede ser el de la configuración principal del servidor así como el de la configuración de los host virtuales.

Directivas	Definicion
`TransferLog`	Directiva que define el nombre del archivo de registro o al programa al que se envía la información de registro. Emplea los especificadores asignados por la directiva `LogFormat`.
`LogFormat`	Directiva que define el formato del archivo de registro asignado con la directiva `TransferLog`
`ErrorLog`	Directiva que permite registrar todos los errores que encuentre Apache. Permite guardar la información en un archivo de registro o bien en `syslog`
`CustomLog`	Directiva similar a la directiva `TransferLog`, pero con la particularidad que permite personalizar el formato de registro empleando los especificadores anteriormente vistos.
`CookieLog`	Directiva que define el nombre del archivo de registro donde registrar información sobre cookies

La tabla siguiente muestra la sintaxis y el uso de las anteriores directivas:

Sintaxis y uso de directivas para archivos de registro

Directiva TransferLog	
Sintaxis	`TransferLog nombre_fichero_archivo_registro | tubería_para_enviar_al_programa_la_información_de_registro`
Uso	`TransferLog logs/acceso_a_empresal.log`
Directiva LogFormat	
Sintaxis	`LogFormat nombre_fichero_archivo_registro [opcional_alias] [opcional_alias]` permite definir un logformat con un nombre de tal forma que cuando hacemos referencia al nombre lo hacemos al logformat vinculado.
Uso	`LogFormat logs/acceso_a_empresal.log`
Directiva ErrorLog	
Sintaxis	`ErrorLog nombre_fichero_archivo_registro`
Uso	`ErrorLog logs/acceso_a_empresal.log`
Directiva CustomLog	
Sintaxis	`CustomLog nombre_fichero_archivo_registro| tubería_para_enviar_al_programa_la_información_de_registro [variable_de_entorno_opcional]`
Uso	`CustomLog logs/acceso_a_empresal.log`
Directiva CookieLog	
Sintaxis	`CookieLog nombre_fichero_archivo_registro`
Uso	`CookieLog logs/acceso_a_empresal.log`

En GNU/Linux se puede comprobar en tiempo real desde un terminal en el equipo que guarda los logs -que puede ser el propio equipo servidor web- que es lo que ocurre cuando se accede a una página web observando el contenido de los archivos de registro mediante el comando: tail -f nombre_archivo_de_registro.log.

6.2. ROTACIÓN DE LOS ARCHIVOS DE REGISTRO (I).

Como los archivos de registro a medida que pasa el tiempo van incrementando su tamaño, debe existir una política de mantenimiento de registros para que éstos no consuman demasiados recursos en el servidor, así es conveniente rotar los archivos de registro, esto es, hay que depurarlos, comprimirlos y guardarlos. Básicamente hay dos opciones para rotar los registros: rotatelogs un programa proporcionado por Apache, o logrotate, una utilidad presente en la mayoría de los sistemas GNU/Linux.

No se debe olvidar que la información recopilada en los ficheros log se debe conservar al menos durante 1 año por eventuales necesidades legales, de este modo, además de rotarlos se opta habitualmente por comprimir logs.

Uso de rotatelogs

```
CustomLog "|ruta_rotatelogs ruta_log_a_rotar numero_segundos|tamaño_máximoMB" alias_logformat
```

Ejemplos

Rotar el archivo de registro access.log cada 24horas

```
CustomLog "|/usr/sbin/rotatelogs /var/log/apache2/access.log 86400" common
```

Rotar el archivo de registro access.log cada vez que alcanza un tamaño de 5 megabytes

```
CustomLog "|/usr/sbin/rotatelogs /var/logs/apache2/access.log 5M" common
```

Rotar el archivo de registro error.log cada vez que alcanza un tamaño de 5 megabytes y el archivo se guardará con el sufijo de formato : YYYY-mm-dd-HH_MM_SS (Año-Mes-Día-Hora_Minutos_Segundos)

```
ErrorLog "|/usr/sbin/rotatelogs /var/logs/errorlog.%Y-%m-%d-%H_%M_%S 5M" common
```

Los ficheros rotados por intervalo de tiempo, lo harán siempre y cuando en el intervalo de tiempo definido existan nuevos datos.

Por defecto, si no se define formato mediante ningún modificador % para guardar los archivos de registro, el sufijo nnnnnnnnnn (10 cifras) se agrega automáticamente y es el tiempo en segundos traspasados desde las 24 horas (medianoche).

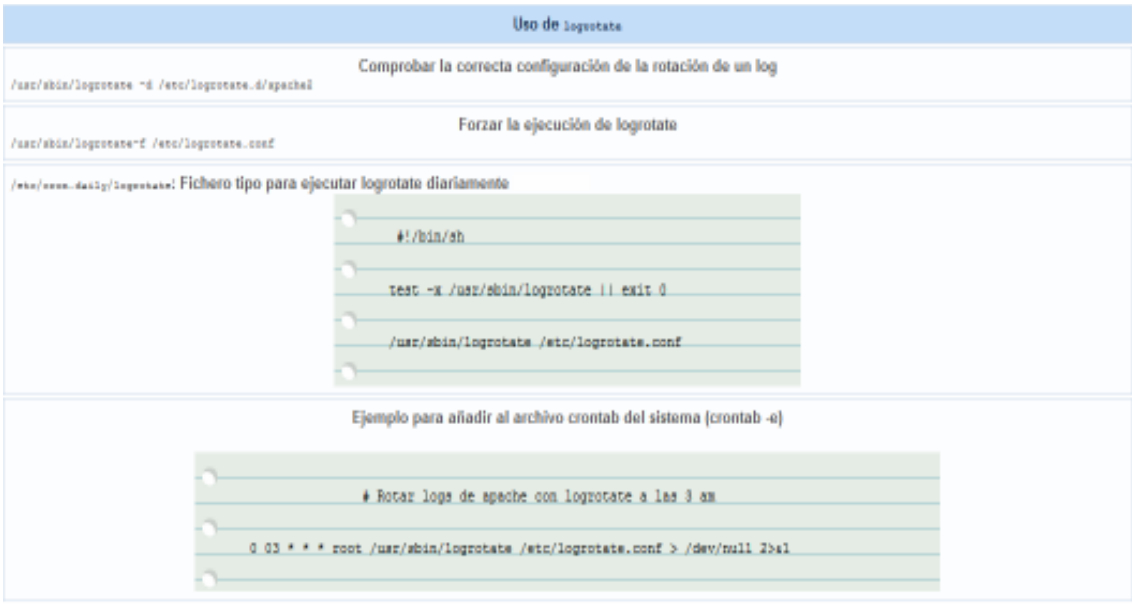
6.2.1. ROTACIÓN DE LOS ARCHIVOS DE REGISTRO (II).

El programa logrotate rota, comprime y envía archivos de registro a diario, semanalmente, mensualmente o según el tamaño del archivo.

En Debian se puede encontrar los siguientes archivos de configuración para logrotate:

- /etc/logrotate.conf: Define los parámetros globales, esto es, los parámetros por defecto de logrotate. Se puede encontrar un archivo tipo en el siguiente enlace: logrotate.conf.

- /etc/logrotate.d/apache2: Define para apache2 el rotado de logs, todos aquellos parámetros que no se encuentren aquí recogen su valor del fichero /etc/logrotate.conf. Se puede encontrar un archivo tipo en el siguiente enlace: logrotate.d/apache2



7. DESPLIEGUE DE APLICACIONES SOBRE SERVIDORES WEB.

Normalmente las aplicaciones sobre servidores web necesitan de los siguientes elementos para su correcto funcionamiento: soporte php y soporte sql.

El servidor web puede tener soporte php, pero el soporte sql debe ser ofrecido por otro servidor al que pueda acceder el servidor web. Este servidor con soporte sql puede estar configurado en el mismo equipo que el servidor web o en otro.

El procedimiento suele ser el siguiente:

1. Se descarga la aplicación.
2. Se configura para que sea visible a través del servidor web.
3. Suele traer una página de instalación que verifique si el servidor web cumple los requisitos para la instalación de la aplicación.
4. Es necesaria antes de finalizar el proceso de instalación autenticarse al servidor sql con un usuario con permisos para crear/modificar una base de datos. Puede que previamente se tenga que crear la base de datos para que el proceso de instalación genere las tablas necesarias en la misma.
5. Se pide un usuario y contraseña para poder acceder a la aplicación web.
6. Fin de la instalación.