

Caso práctico



A la empresa BK Programación le ha surgido un nuevo proyecto: una empresa con varias sucursales quiere montar una aplicación web por sucursal.

Ada, la directora, considera que para afrontar este proyecto y atender así la demanda ofrecida, deben configurar un nuevo equipo servidor. Para tal fin se reúne con María:



-Hola María -dijo Ada-, nos han ofrecido un nuevo proyecto relacionado con servicios web, pienso que podemos afrontarlo, pero quería saber tu opinión: ¿con la infraestructura que tenemos ahora ves necesario el montaje de otro equipo servidor dedicado a este proyecto o con lo que tenemos nos arreglamos?

-Pienso -dijo María- que tal como estamos ahora, sí o sí, independientemente de los recursos que consuma este nuevo proyecto necesitamos la configuración de otro equipo servidor. Además debemos configurar dos entornos: el de pruebas y el de producción. ¿Para cuándo sería el proyecto?

-El proyecto debemos entregarlo con fecha final dentro de tres meses.

-Entonces, creo que si todo sigue su cauce normal no tendremos ningún tipo de problema para la ejecución del proyecto. ¿Qué recursos humanos habías pensado y dispones para destinar al proyecto?

-Ahora disponemos de todo el personal de la empresa y cuento contigo y con Juan para que os coordinéis las funciones de este proyecto.

-Pues por mí, no veo objeción al mismo.

-Bien -asintió Ada-, entonces no se hable más, tendremos que configurar otro equipo servidor y aceptamos el proyecto.

Así, la empresa BK Programación envió un presupuesto a la empresa del proyecto, ésta lo aprobó y comenzó el trabajo.

Para afrontar el nuevo proyecto al que se enfrenta BK Programación se acuerda en una reunión en la que asistieron: Ada, María y Juan, quien sería destinado al nuevo proyecto y las funciones a realizar en el mismo. Así, en dicha reunión se determinó que María sería la encargada del montaje, configuración y administración del nuevo equipo servidor y Juan el encargado de coordinar con el resto del personal la creación y funcionamiento de las aplicaciones web del proyecto.

María, entonces, se puso manos a la obra y determinó el siguiente escenario de trabajo para el equipo servidor de este proyecto (que iremos montando):

- ✓ Sistema Operativo: Ubuntu 14.04 LTS
- ✓ Servidor Web: Apache (apache2)
 - ✦ Configuración de Red:
 - ✦ Servidor Web: 192.168.200.250
 - ✦ Cliente de pruebas (desde donde se lanza el navegador): 192.168.200.100

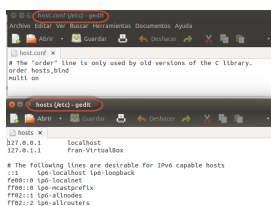
Citas para pensar

Albert Einstein: "Se debe hacer todo tan sencillo como sea posible, pero no más sencillo."

Hay que tener en cuenta que en el escenario las IP empleadas son **IP privadas**, sin existencia en Internet, por lo cual siempre que se haga referencia a las mismas a través de nombre de dominios, deberá existir un **servidor DNS** que las resuelva en local o bien en su defecto deberán existir las entradas correspondientes en el fichero del sistema local **/etc/hosts.conf**.

Host.conf, archivo de configuración para el **cliente DNS** que permite especificar entre otros parámetros:

- El orden de los sistema de resolución de nombres
- Sistema a utilizar en la resolución de nombres de dominio (**archivo hosts** o servicio DNS)
- Si se permiten o no varias direcciones IP en el fichero hosts



1.- Funcionamiento de un servidor Web.

Caso práctico

Para poder llevar a buen fin el proyecto, María, reúne al equipo destinado al mismo, ya que quiere que todo el personal tenga claro los requisitos, entregables y fechas de ejecución del proyecto. Así, en esta reunión informativa para todo el equipo destinado al proyecto, trató los siguientes temas:

1. Recursos del equipo servidor.
2. Conectividad del equipo servidor.
3. Servidor web empleado: El porqué de su elección y funcionamiento.
4. Posibilidades del servidor web empleado.
5. Requisitos de las aplicaciones web del proyecto.
6. Entregables y fechas.



Reflexiona

¿Alguna vez te has parado a pensar qué existe detrás de una página web? ¿Por qué al escribir un nombre en un navegador puedes visionar una página web? ¿Por qué no tienes acceso a determinadas páginas? ¿De qué modo puedes impedir el acceso a determinados sitios de una página: por directorio, por usuario? ¿Cómo se puede establecer una comunicación segura en una transición bancaria? ...

Recomendación

A partir de ahora vamos a trabajar en la configuración del servidor web (Apache), es conveniente hacer una copia del fichero original del mismo. Podemos hacerla de diferentes maneras, por ejemplo:

```
cd /etc/apache2/  
sudo cp apache2.conf apache2.conf.old
```

Hoy en día utilizamos Internet como una herramienta común: para el trabajo, para el ocio... Pero sin duda el elemento fundamental que usamos no es otro que el navegador, gracias al cual podemos sacar partido a todo lo que se encuentra en Internet: comprar entradas para el cine, acceder a nuestra cuenta bancaria, averiguar el tiempo que hará el fin de semana... pero nada de esto tendría sentido si detrás de cada página web a la que accedemos no existiera un servidor web, el cual permite que la página esté accesible 24x7 (24 horas al día y 7 días a la semana, es decir, siempre).

Detrás de cada página web debe existir un servidor web que ofrezca esa página, bien a los internautas, a los trabajadores de una empresa -por tratarse de una página web interna, de la empresa, no accesible a Internet-, o 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.

Pero ¿un servidor web pueda alojar varias páginas web o solamente una? Es más, ¿puede alojar varios sitios o dominios o solamente uno, esto es, permite host virtuales? Pues, 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 ser modular -fácilmente se le pueden añadir o quitar características-, o por la contra si queremos añadirle una funcionalidad que no posea en la instalación base debemos desinstalarlo e instalarlo de nuevo, por ejemplo: hasta ahora el servidor web solamente ofrecía páginas estáticas pero queremos ofrecer también páginas web dinámicas, qué hacemos: modular o nueva instalación.

También tenemos que pensar que todo puede crecer y lo que ahora era un servidor web que ofrecía x número de páginas necesitamos que ofrezca x*y, con lo cuál tenemos que prever la escalabilidad del servidor web, y también la estabilidad: ¿cómo se comporta ante múltiples conexiones simultáneas?

De nada servirá tener instalado un servidor web sin saber como se va a comportar ofreciendo el servicio, con lo cuál será muy importante previamente y durante el funcionamiento del servidor establecer unas pruebas de funcionamiento del mismo y registrar lo acontecido.

Por todo lo anteriormente comentado veremos como configurar y administrar el servidor Apache (apache2), ya que soporta: páginas web estáticas, dinámicas, hosts virtuales, seguridad mediante cifrado, autenticación y control de acceso, modularización y monitorización de archivos de registro.



1.1.- Servicio de ficheros estáticos.

Reflexiona

¿Es necesario que todas las páginas web se modifiquen constantemente? ¿Un blog sería útil si el contenido no sufre cambios? ¿Y un manual? ¿Si actualizamos un manual la página deja de ser estática?

Todas aquellas páginas web que durante el tiempo no cambian su contenido no necesariamente son estáticas. Una página estática puede modificarse, actualizando su contenido y seguir siendo estática, ¿entonces? Entonces debemos diferenciar cuando accedemos a una página web entre código ejecutable en el lado del servidor y en el lado del cliente -equipo que solicita la página mediante el cliente web (navegador)-. Si al acceder a una página web no es necesaria la intervención de código en el lado del servidor -por ejemplo código PHP- o en el lado del cliente -por ejemplo `javascript`- entonces entenderemos que la página es estática, si por el contrario es necesaria la intervención en el lado del servidor y/o en el lado del cliente entenderemos que la página es dinámica.



Ofrecer páginas estáticas es simple, puesto que solamente se necesita que el servidor web disponga de soporte `html/xhtml/css` o incluso solamente `html/xhtml`. En cuanto a configuración y administración del servidor es el caso más simple: solamente se necesita un soporte mínimo base de instalación del servidor Apache, esto es, no se necesita por ejemplo soporte PHP. En cuanto a rendimiento del servidor, sigue siendo el caso más beneficioso: no necesita de ejecución de código en el lado del servidor para visionar la página y tampoco necesita ejecución de código en el lado del cliente, lo que significa menos coste de `CPU` y memoria en el servidor y en el cliente, y por lo tanto una mayor rapidez en el acceso a la información de la página.

Para poder ofrecer páginas estáticas mediante el servidor Apache simplemente copias la página en la ruta correspondiente donde quieres que se visiona la página. Así por ejemplo cuando se instala Apache en un GNU/Linux Debian 6 se crean una serie de rutas en el equipo servidor similar a la estructura siguiente.

Rutas de interés en la instalación de Apache (apache2)

Rutas de interés en la instalación de Apache (apache2) en un GNU/Linux Debian	
<pre>/etc/apache2/ ├── apache2.conf ├── conf.d ├── envvars ├── httpd.conf ├── magic ├── mods-available ├── mods-enabled ├── ports.conf ├── sites-available └── sites-enabled</pre>	<pre>/etc/apache2/sites-available/ ├── default └── default-ssl /var/www/html └── index.html /etc/apache2/mods-available/mime.conf /etc/apache2/apache2.conf</pre>

En la instalación de Apache se crea una página web en `/var/www/index.html` referenciada a través del archivo [default \(/etc/apache/sites-available/default\)](#) (0.59 KB), éste contiene la configuración por defecto, generada en la instalación de Apache, para esa página. Si solamente quieres servir una página web la forma más fácil de hacerlo sería sustituyendo la página `index.html`, referenciada en `default`, por la página que quieres servir, por ejemplo `empresa.html`. Puedes comprobarlo siguiendo el procedimiento:

1. Abres el navegador en la página por defecto creada en la instalación de Apache: `index.html`.
2. Sustituyes los archivos en el servidor. Ten en cuenta que la página a servir debe siempre poseer el nombre `index.html`.
3. Pulsas F5 en el navegador para actualizar la página y la página que verás será la tuya.

Siempre en el directorio `/var/www/` si lo que quieres es servir otra página, por ejemplo `empresa.html`, simplemente no le cambies como antes el nombre, deja el que posee la página. Ahora podrás ver dos páginas en el servidor: la página `index.html` y la página `empresa.html`. Si lo que quieres es servir más páginas pues, como antes, simplemente vas subiendo al servidor las páginas e incluso podrías organizarlas en carpetas.

Para saber más

Te proponemos que hagas un viaje por la página web de documentación de Apache.

[Página web oficial de documentación de Apache \(apache 2.2\)](#)

1.2.- Contenido dinámico.

Citas para pensar

Miguel de Unamuno: "El progreso consiste en el cambio."

Muchas veces seguro que te encuentras visitando una página web y la información te parece tan interesante que procedes y guardas en **Favorito** la direcciónURL para una posterior visión, pero cuando de nuevo deseas ver la página resulta que lo que estás viendo no tiene nada que ver o es distinto de lo que esperabas, ¿qué ha ocurrido? Pues puede que la página haya cambiado su contenido o que la página que visitas posee contenido no estático, dinámico, dependiente del código ejecutado en el servidor o en el cliente al acceder a la página.

Imagínate que accedes a una página web y dependiendo si posees una cuenta de usuario u otra el contenido es distinto, o que presionas en una imagen de la página y se produce un efecto en la misma, o que el contenido cambia dependiendo del navegador. De cualquier forma la página ha sido modificada mediante una interacción con el usuario y/o el navegador, por lo tanto nos encontramos con una página dinámica.

Como bien puedes pensar, 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 másmódulos 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 debemos 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`.



Para saber más

En el siguiente enlace a la página de Apache puedes ampliar la información que te proporcionamos sobre los módulos.

[Página web oficial de documentación de Apache \(apache 2.2\) sobre módulos.](#)

Autoevaluación

Abres el navegador y solicitas una página a un servidor web: ¿cuál de las siguientes acciones indica que la página solicitada no es dinámica?

- ☐ La página tiene un panel de control, al cual accedes mediante tu usuario y tu contraseña, los cuales nunca cambias. La página entonces establece comunicación con una base de datos y te permite el acceso a tu perfil, distinto del perfil del administrador de la página.
- ☐ Al pasar el puntero por encima de una imagen, ésta se redimensiona y al salir vuelve al tamaño original.
- ☐ Cuando visitas la página con distintos navegadores aparece un comentario de alerta indicando el navegador con el cual estás accediendo a la página.
- ☐ La página solicitada es un manual sobre el Servidor Apache, y está totalmente escrita en código HTML y CSS.

No es correcta. Si la página enseña perfiles distintos según el usuario que acceda a la misma, la página es dinámica.

No es correcta. Si la página web se modifica por la interacción del usuario la página es dinámica.

No es correcta. Si la página web muestra acciones distintas según el navegador que solicita la página, la página es dinámica.

Efectivamente ésta opción es cierta. Aquellas páginas cuyo contenido no depende de la interacción del usuario, del navegador o un sistema gestor de bases de datos son páginas estáticas.

Solución

1. **Incorrecto** ([Retroalimentación](#))
2. **Incorrecto** ([Retroalimentación](#))
3. **Incorrecto** ([Retroalimentación](#))
4. **Opción correcta** ([Retroalimentación](#))



1.3. Directivas del archivo de configuración

Debes conocer

Las directivas presentes en los [ficheros de configuración](#) pueden ser de aplicación para todo el servidor, o puede que su aplicación se limite solamente a determinados directorios, ficheros, hosts, o URLs. Podemos usar las secciones de configuración y los ficheros `.htaccess` para modificar el ámbito de aplicación de las directivas de configuración.

ServerRoot

```
#ServerRoot "/etc/apache2"
```

Que indica el directorio raíz de la instalación de Apache. No se refiere al directorio donde colocaremos las páginas web. Ten en cuenta que está comentada porque es la que usa por defecto.

Conviene recordar que al realizar la instalación manual, antes de compilar el servidor lo configuraríamos con la siguiente línea, y estaríamos indicando que es `/www` `./configure --prefix=/www --enable-shared=max --enable-wmodule=rewrite --enable-module=so`

Si nos fijamos, el directorio coincide con el valor de `--prefix`.

Esta directiva solo se modificaría en caso de mover el servidor Apache a otra ubicación en la estructura de directorios y como iremos viendo habría que cambiar más. Lo mejor es elegir bien desde el principio dónde instalaremos Apache.

En el caso que estamos viendo, no aparece ninguna otra ruta, pero para configurar rutas relativas al directorio de instalación de Apache, una vez especificado con la directiva anterior, podríamos usar `%ServerRoot%` en lugar de la ruta completa.

PidFile

PidFile establece la ruta al archivo en el que el servidor graba su ID de proceso (pid). Por defecto, el PID se coloca en `%ServerRoot%/logs`

En el archivo vemos que aparecen unas constantes que están definidas en el archivo `/etc/apache2/envvars`

Para ver los valores reales puedes editar el archivo, nosotros usaremos ejemplos concretos porque son más claros.

```
PidFile logs/httpd.pid
```

No se recomienda cambiar la ruta si no se sabe muy bien lo que se está haciendo.

Timeout

Son los segundos que se esperan las respuestas durante la comunicación. Por defecto es 300 segundos y se recomienda no cambiarlo.

```
Timeout 300
```

KeepAlive, MaxKeepAliveRequests y KeepAliveTimeout

Determina si el servidor va a permitir que cada conexión haga más de una petición. El problema de activarlo es que un único cliente puede consumir demasiados recursos y saturar el servidor por lo que en caso de establecerlo a on se recomienda configurar cuidadosamente `KeepAliveTimeout`, generalmente a un nivel bajo (en la versión 2.2 por defecto era 15 y se ha bajado a 5).

```
KeepAlive Off
```

`MaxKeepAliveRequests` determina el número de peticiones que podrá realizar cada conexión. Evidentemente solo tiene sentido si `KeepAlive` está activada.

```
MaxKeepAliveRequests 100
```

`KeepAliveTimeout` determina el tiempo que el servidor esperará antes de atender una nueva petición del mismo cliente en la misma conexión.

```
KeepAliveTimeout 5
```

Estas directivas son un buen ejemplo de elementos que pueden hacer que nuestro servidor no funcione como esperamos una vez en producción. Si no probamos con un número de peticiones superior al máximo no podremos comprobar si el funcionamiento es el esperado.

IfModule

Es un contenedor que permite establecer determinadas opciones solo si se ha cargado un módulo determinado. Si se escribe `!` (cierre de exclamación) antes del nombre del módulo se ejecutan las opciones si no se ha cargado el módulo.

Esta directiva no aparece en la configuración principal porque se ha desplazado al archivo de configuración de cada módulo, pero se mantiene aquí porque su uso es importante.

```
<IfModule mod_mime_magic.c>
    MIMEMagicFile conf/magic
</IfModule>
```

StartServers

Esta directiva aparece en el archivo `/etc/apache2/mods-enabled/mpm_event.conf`

```
<IfModule mpm_event_module>
    StartServers 2
    MinSpareThreads 25
    MaxSpareThreads 75
    ThreadLimit 64
    ThreadsPerChild 25
    MaxRequestWorkers 150
    MaxConnectionsPerChild 0
</IfModule>
```

Apache crea y destruye servidores automáticamente según el tráfico que tenga que atender en cada momento. Apache es muy eficiente en esto por lo que no deberíamos preocuparnos demasiado de este y de los siguientes parámetros. Además deberían ir dentro de un `IfModule` según al que queramos aplicarlo.

Esta directriz establece cuantos servidores se crearán al arrancar.

```
StartServers 5
```

Al final de la explicación de la directiva podemos ver que se usan otras asociadas a esta para determinar cuántos servidores o hilos (hebras) se deben mantener a la espera.

Listen

Esta directiva se encuentra en `ports.conf`

Este elemento indica al servidor en qué dirección y puerto debe escuchar las peticiones http que lleguen además de los de por defecto. El puerto estándar que un servidor web reciba peticiones http es el 80 por lo que la línea que nos encontramos es

```
Listen 80
```

En nuestro caso hemos instalado Apache en una máquina virtual así que está configurado para escuchar en la interfaz de loopback (127.0.0.1). Si queremos que el servidor sea visto desde la máquina host debemos configurarlo situando la dirección IP que configuramos para la tarjeta de red que añadimos como solo-anfitrión. La dirección IP y el puerto se separan por dos puntos.

Listen 192.168.56.101:80

Si configuramos Apache para escuchar en otro puerto solo podremos acceder a las páginas web añadiendo dicho puerto detrás de la dirección. Como vimos al instalar Tomcat, éste escucha por defecto en el puerto 8080 y por ello accedíamos a él con `http://localhost:8080`

LoadModule

Cuando instalamos Apache, habilitamos la opción de Dynamic Shared Object (DSO) que permite añadir módulos dinámicamente sin necesidad de recompilar el servidor. La directiva LoadModule indica qué módulos dinámicos cargar. No es necesario incluir los módulos que se compilaron con el servidor.

En el archivo aparece comentada porque no añadimos ningún módulo. Más adelante hablaremos de los módulos.

De todas formas en nuestra distribución se ha optado por otro método que veremos más adelante.

User y Group

Estas dos directivas indican con qué usuario y grupo se lanzarán los procesos hijos que genere Apache. Estos procesos hijo no deben lanzarse con el usuario root por razones de seguridad ya que crearían una brecha perfecta para los hackers.

Si no arrancamos el servidor con el usuario root, los procesos hijo que se lanzarán con ese mismo usuario ya que solo root puede cambiar el usuario y el grupo de un proceso por lo que estas directivas se ignorarán.

En Linux por defecto el usuario nobody y el grupo nogroup tienen muy pocos privilegios por lo que son buenos candidatos para lanzar los procesos hijo.

Si queremos usar el número del grupo o del usuario en lugar del nombre debemos añadir # justo antes del número.

User nobody

Group #-1

En nuestra distribución se utilizan dos variables globales.

ServerAdmin

Esta directiva y las siguientes aparecen en `/etc/apache2/sites-available/000-default.conf`

Esta opción permite configurar la dirección del administrador del servidor web que se mostrará si el servidor genera una página de error. Evidentemente debe ser una dirección real y generalmente es del mismo dominio que el propio servidor web. Nosotros no disponemos de un nombre de dominio, pero podemos configurar una dirección de correo real.

ServerAdmin profesor.scv@gmail.com

ServerName

Indica el nombre del servidor. Debe cumplir con las especificaciones DNS y estar en nuestro poder.

ServerName www.ejemplo.es:80

DocumentRoot

Con esta opción indicamos el directorio raíz donde colocaremos las páginas web. Podemos crear subdirectorios dentro de éste y accederemos a los documentos que pongamos en el subdirectorio con una ruta relativa.

DocumentRoot "/www/docweb"

El problema es que no basta con cambiar esta ruta, es necesario cambiar también otra de la directiva Directory que veremos luego.

<Directory "/www/docweb">

...

Si queremos comprobar que funciona lo mejor es modificar el archivo `index.html` del nuevo directorio o añadir un archivo nuevo y cargar ese.

En el caso que nos ocupa ahora, esta directiva aparece en el archivo `default` del directorio `/etc/apache2/sites-available`. Esto es así porque la instalación que usamos nosotros viene prepeorada por defecto para funcionar con sitios virtuales.

Directory

Esta directiva había sido derivada al archivo de configuración del sitio por defecto, para que no se aplicara a todos los sitios del servidor ya que no permite sobrescritura de las directivas de seguridad. En nuestro caso está en el servidor por lo que afectaría a todos los sitios que utilizaran el directorio especificado.

Esta opción se usa para configurar cómo se comportará y qué se permitirá en cada directorio al que tiene acceso el servidor Apache. Esta configuración se aplica a un directorio y los subdirectorios que contiene si no se sobrescribe en otra definición sobre un directorio más concreto.

Nos encontramos dos veces esta etiqueta. La primera hace referencia al directorio raíz y se configura siempre con opciones muy restrictivas.

<Directory />

Options FollowSymLinks

AllowOverride None

Require all denied

</Directory>

que impide que nadie que accede al servidor pueda acceder a ningún directorio de la estructura de nuestro servidor ya que se aplica a los subdirectorios también y estamos partiendo del directorio raíz.

Esto es una de las cosas que ha cambiado desde la versión 2.2 ya que antes se utilizaba una sintaxis diferente a la hora de establecer los permisos.

Notación 2.2

<Directory />

Options FollowSymLinks

AllowOverride None

Order deny,allow

Deny from all

</Directory>

El contenido lo vemos a continuación. Primero vamos a mostrar también la otra vez que aparece, haciendo referencia al directorio DocumentRoot y al compartido del usuario.

<Directory /usr/share>

AllowOverride None

Require all granted

</Directory>

<Directory /var/www/>

Options Indexes FollowSymLinks

AllowOverride None

Require all granted

</Directory>

Aunque ambos casos estén configurados igual, podrían diferir.

Pueden añadirse directivas para otros directorios según lo vayamos necesitando.

La primera línea permite a Apache seguir enlaces simbólicos y la segunda que las opciones de acceso de cada directorio (archivo `.htaccess` del que hablaremos en la directiva `AccessFileName`) no priman sobre éstas. La tercera directiva indica los permisos. Por ahora nos basta saber que se permite el acceso a todos (y en la anterior se denegaba a todos).

DirectoryIndex

Esta directiva en la actualidad se encuentra en `/etc/apache2/mods-enabled/dir.conf`

Especifica la página por defecto que se buscará al acceder a un directorio de la jerarquía de nuestro sitio. Acceder a un directorio es usar una dirección web que acaba con una barra "/". El directorio raíz está incluido en esa jerarquía por lo que cuando accedemos a nuestro sitio web (en nuestros casos http://localhost) carga el archivo index.html que es el valor por defecto de DirectoryIndex.

DirectoryIndex index.html

Puede establecerse una sucesión de archivos y el servidor mostrará la primera que encuentre del orden establecido en la directiva.

DirectoryIndex index.html, index.htm, inicio.html, inicio.htm

Si accedemos a un directorio que no contiene ninguno de los archivos especificados, Apache crea dinámicamente un archivo que lista los contenidos.

AccessFileName

Indica el nombre del archivo en el que se deben buscar las directivas de acceso determinadas en cada directorio. Por defecto es .htaccess y no se recomienda cambiarlo en absoluto.

AccessFileName .htaccess

Files

Es un contenedor que sirve para establecer directivas para tipos de archivo. Por lo menos es necesario añadir un grupo que impida el acceso a los archivos que empiezan por .ht por motivos de seguridad.

```
<FilesMatch "\.ht">
Require all denied
</FilesMatch>
```

Se puede usar para otros tipos de archivo según creamos conveniente.

HostnameLookups

Indica al servidor si debe hacer una consulta DNS para cada petición. Esto consume mucho tiempo por lo que por defecto está deshabilitada.

HostnameLookups off

ServerSignature

Esta directiva se encuentra en el archivo /etc/apache2/conf-available/security.conf

Indica si al mostrarse una página generada automáticamente por el servidor (no las generadas mediante lenguajes a partir de algo establecido por el usuario sino las páginas de error, listado de directorios FTP, etc) debe mostrarse el nombre y la versión del servidor. Esto puede ser usado maliciosamente así que si no estás conforme establécelo a off.

ServerSignature On

Hay otra opción (EMail) que añade la dirección del administrador de la web a la información mostrada.

Alias

Esta directiva se encuentra en el archivo /etc/apache2/mods-enabled/alias.conf

Permite crear alias para archivos o directorios. Se pueden añadir los que queramos pero el más habitual es el de la carpeta de iconos que usa Apache.

```
Alias /icons/ "/usr/share/apache2/icons/"
<Directory "/usr/share/apache2/icons">
Options FollowSymlinks
AllowOverride None
Require all granted
</Directory>
```

La directiva es solo la primera línea. Lo demás es la directiva de configuración del directorio.

ScriptAlias

Se encuentra en /etc/apache2/conf-enabled/serve-cgi-bin.conf

Indica dónde se ubicará la carpeta de scripts CGI. El funcionamiento es similar al anterior pero solo debemos incluirlos si vamos a usar scripts CGI.

```
ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
<Directory "/usr/lib/cgi-bin">
AllowOverride None
Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
Require all granted
</Directory>
```

IndexOptions, AddIconByEncoding, AddIconByType, AddIcon, DefaultIcon, ReadmeName, HeaderName e IndexIgnore

Son directivas asociadas a la creación de índices de manera automática por el servidor. Cuando nos muestra el servidor web un listado de contenidos de un directorio por ejemplo. Están en el archivo /etc/apache2/mods-enabled/autoindex.conf

No vamos a profundizar en ellas.

LanguagePriority

Tanto esta directiva como la siguiente tienen mucho que ver son el módulo MIME que veremos más adelante aunque no forman parte de él y con los conocimientos que traemos del primer curso podemos entender cómo funcionan.

Permite establecer una prioridad de los idiomas en caso de que no se especifique uno o haya un empate en la negociación por diferentes motivos. Por defecto viene en inglés, pero en la mayoría de los casos nosotros queremos establecerlo en español.

LanguagePriority es en fr de

Para poder probar este tipo de configuración es necesario tener varios archivos en diferentes idiomas con el mismo nombre. En ésta página podemos ver cómo hacerlo, pero aunque es bastante completa es antigua. En el W3c nos proporcionan una información más reciente y en la documentación oficial de Apache aparece todo especificado.

Los códigos de idiomas están mantenidos por la IANA. Como son los mismos que se utilizan en HTML, XML, etc existen páginas muy completas sobre ellos.

AddDefaultCharset

Debería establecerse al juego de caracteres que mejor se ajuste a la zona en la que se sitúa el servidor y al idioma del contenido. En caso de no estar seguros es mejor dejarlo como está.

AddDefaultCharset ISO-8859-1

BrowserMatch

Se encuentran (entre otros) en /etc/apache2/mods-enabled/setenvif.conf

Sirve para modificar la respuesta dependiendo de la configuración del cliente en cuanto a navegador y plugins. Esto suele usarse para evitar problemas con navegadores que no siguen algún estándar o evitar agujeros de seguridad. Las siguientes líneas son muy comunes porque incluyen configuraciones con problemas de sobra conocidos.

```
BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4.0b2:" nokeepalive downgrade-1.0 force-response-1.0
BrowserMatch "RealPlayer 4.0" force-response-1.0
```


BrowserMatch "Java/1\0" force-response-1.0
BrowserMatch "JDK/1\0" force-response-1.0

Por ejemplo podemos configurar el servidor Apache para que la carpeta raíz sea /pagweb y prueba a consultarlo desde la máquina host. Solo admitirá consultar por el puerto 9090. Además la página por defecto que cargará en cada directorio debe ser inicio.html. Por último añade el idioma español, catalán, gallego y vasco en este orden. El juego de caracteres por defecto debe ser el europeo occidental. Prueba el correcto funcionamiento de todo.

Para saber más

[Guía rápida de directivas Apache](#)

[Índice de directivas de Apache](#)

Registros de error

Los archivos de son fundamentales en la gestión de servidores ya que nos permiten comprobar qué ha sucedido en cada momento. Aunque hay otros tipos, los que veremos ahora son los que guardan errores de configuración o funcionamiento. Estas directivas de registro de errores pueden ir tanto en la configuración principal como en la de algún sitio (host) específico. Aunque no nos extenderemos por falta de tiempo sí vamos a echar un vistazo general al registro, ya que es tremendamente útil cuando tenemos algún problema de funcionamiento en el servidor o alguno de los sitios web enlazados. Lo primero es echar un vistazo general a la documentación de Apache para hacernos una idea.

ErrorLog

Esta directiva es muy importante ya que indica dónde ubicar el archivo de registro de los errores que se produzcan en el servidor. El lugar por defecto es %ServerRoot%/logs/error_log. Muchos administradores crean una partición exclusivamente para situar este tipo de archivos y así tener más probabilidades de poder consultarlos en caso de un error fatal.

ErrorLog logs/error_log

LogLevel

Establece cuánta información se guardará en el archivo de registro de errores. El nivel por defecto es suficiente para empezar pero cuanto mayor sea el servidor y más importante su función más información necesitaremos. Los valores posibles incluyen: debug, info, notice, warn, error, crit, alert, emerg.

LogLevel warn

LogFormat

Establece qué y en qué formato se registrará.

CustomLog

Establece la ruta al archivo de que registra las visitas a nuestro sitio web.

CustomLog logs/access_log combined

1.4.- Protocolo HTTP y HTTPS.



¿Quieres conservar la información de forma confidencial? ¿Quieres transferir información de forma segura? Si estás pensando en este tipo de preguntas necesariamente estás pensando en el protocolo HTTPS y no en el protocolo HTTP.

El protocolo HTTPS permite que la información viaje de forma segura entre el cliente y el servidor, por la contra el protocolo HTTP envía la información en texto claro, esto es, 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 y siempre y cuando sean validados la información será transferida cifrada. Pero cifrar la información requiere un tiempo de computación, por lo que será perjudicado el rendimiento del servidor web. Así, ¿es necesario que toda, absolutamente 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.

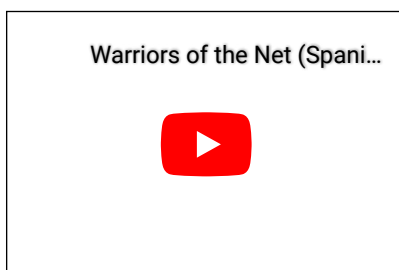
Un servidor web, como Apache, puede emitir certificados, pero puede que en algún navegador sea interpretado como peligroso, esto suele ser debido a que los navegadores poseen en su configuración una lista de Entidades Certificadoras que verifican, autentican y dan validez a los certificados. ¿Tú, confiarías en un DN que no fuese certificado por una entidad de confianza como el Ministerio del Interior? Pues, lo mismo le pasa a los navegadores, solamente confían en quien confían. Eso no quiere decir que no puedes crear tus certificados en un servidor web, de hecho muchas empresas lo hacen, sobre todo para sitios internos o externos en los que solamente puede acceder personal autorizado por la propia empresa. Ahora si, si utilizas certificados mediante Apache en un sitio visible a través de Internet y accesible por cualquier usuario, o bien eres una empresa o entidad en la que de por sí confía el usuario o la imagen de la empresa o entidad quedará muy mal parada, ya que lo más probable es que el usuario no aceptará la comunicación, por visionar en el navegador un aviso de problema de seguridad.

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

Bien, pero, ¿cómo funcionan? En el protocolo HTTP cuando escribes una dirección URL en el navegador, por ejemplo <http://www.debian.org/index.es.html>, antes de ver la página en el navegador existe todo un juego de protocolos, sin profundizar en todos ellos básicamente 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 a tu navegador. Sin embargo cuando escribes en el navegador una dirección URL con llamada al protocolo HTTPS, el procedimiento es similar al anterior pero un poco 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 -el navegador aunque es poco habitual también puede enviar el suyo-. 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.

Puedes 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 visites la página web a mayores en la dirección URL debes 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 4333. Como ves, puedes configurar los puertos, pero ten en cuenta que cualquiera que quisiera acceder a esas páginas debería saber el puerto TCP de la solicitud. Entonces, quiere decir que aunque no escribas el puerto TCP en las direcciones URL estas se interpretan en el puerto 80 y 443 para el protocolo HTTP y HTTPS respectivamente? Pues sí, así es. Es lo mismo escribir <http://www.tupagina.local:80> que <http://www.tupagina.local> y es lo mismo escribir <https://www.tupagina.local:443> que <https://www.tupagina.local>

En la página oficial de [warriorsoftthenet](http://warriorsoftthenet.com) puedes encontrar un vídeo muy ameno sobre el funcionamiento de Internet.



[Resumen textual alternativo](#)

1.5.- Tipos MIME.

Reflexiona

¿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 como 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) :

RFC2045

RFC 2046

RFC 2047

RFC 4288

RFC4289

RFC2077

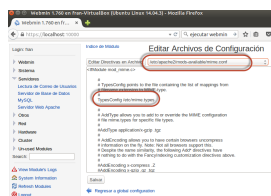
¿Cómo funciona? Imagínate el siguiente ejemplo: Transferencia de una página web.

Cuando un navegador intenta abrir un archivo el estándar MIME le permite saber con que 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.

Bien, pero ¿cómo lo hace?

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, oculto al usuario, es parte de las **cabeceras HTTP**, protocolo que se sigue en la web.

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 cuando visitas <http://www.debian.org/index.es.html> puedes 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.



La directiva que indica donde está el archivo que describe los tipos MIME (mime.types)

TypesConfig etc/mime.types

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

Para saber más

Complementos del navegador Firefox para ver cabeceras HTTP/HTTPS:

[Tamper Data](#)

[Live HTTP Headers](#)

1.5.1.- Configurar el servidor para enviar los tipos MIME correctos.

En un servidor web podemos especificar el tipo MIME por defecto para aquellos archivos que el servidor no pueda identificar automáticamente como pertenecientes a un tipo concreto, esto es, para aquellos los cuales no se resuelven según su extensión.

Para el servidor web Apache se utilizan dos directivas: `DefaultType` y `ForceType`.

- ✓ `DefaultType` asigna la cabecera `Content-Type` a cualquier archivo cuya MIME no pueda determinarse desde la extensión del archivo.
- ✓ `ForceType` hace que todos los ficheros cuyos nombres tengan una equivalencia con lo que se especifique sean servidos como contenido del tipo MIME que se establezca.



Ejemplos:

- ✓ `DefaultType text/plain` : Esto significa que cuando el navegador web solicita y recibe ese archivo como respuesta, desplegará el contenido como un archivo de texto.
- ✓ `DefaultType text/html` : Desplegará el contenido como un archivo HTML.
- ✓ `ForceType image/gif` : Desplegará el contenido como un archivo de imagen `gif`.
- ✓ `ForceType video/mp4` : Desplegará el contenido como un archivo de vídeo `mp4`.

En el siguiente enlace puedes encontrar más información sobre la directiva `DefaultType`.

[Directiva DefaultType](#)

Para saber más

Puedes consultar más información en la documentación de Apache sobre directivas.

[Directivas](#)

[Guía rápida de referencia de directivas](#) .

En el servidor web Apache existe el archivo `/etc/apache2/mods-available/mime.conf` donde encontrarás una referencia al archivo `/etc/mime.types`, el cual contiene la lista de tipos MIME reconocidos por el servidor.

Para saber más

En el siguiente enlace encontrarás la lista oficial de los tipos MIME.

[Lista oficial de los tipos MIME.](#)

[Tipos MIME \(Wikipedia\)](#)

Autoevaluación

Abres el navegador y solicitas una página web que contiene un vídeo con la extensión `.flv` a un servidor web Apache: ¿cuáles de las siguientes afirmaciones son correctas teniendo en cuenta que el vídeo puede reproducirse y visualizarse sin problemas?

- ☐ El servidor web no identifica el tipo MIME pero la extensión `.flv` es reconocida por el navegador, es por esto que el navegador asocia el programa correspondiente al vídeo y se reproduce sin problemas.

.....

- ☐ El archivo no es reconocido por el servidor web, por lo que el servidor web envía al navegador otro tipo MIME, compatible con el esperado y el vídeo se reproduce sin problemas.

.....

- ☐ Si la extensión `.flv` no es reconocida por el navegador ni por el servidor web es debido a que el tipo MIME es reconocido por cómo está programada la página web.

.....

- ☐ El servidor web no identifica el tipo MIME pero como el servidor web reconoce la extensión `.flv` modifica la programación de la página web incorporando el código necesario para la reproducción del vídeo.

.....

Solución

1. [Correcto](#)
2. [Incorrecto](#)
3. [Correcto](#)
4. [Incorrecto](#)

Vídeo acceso a archivos de configuración de Apache y tipos MIME con Webmin:

2.- Hosts virtuales. Creación, configuración y utilización.

Caso práctico

A la empresa BK Programación le ha surgido el siguiente proyecto: una empresa con varias sucursales quiere montar una aplicación web por sucursal. La empresa en cuestión consta de 7 sucursales. Todas ellas dedicadas a la misma línea de negocio. Así, las aplicaciones tendrán un frontal similar, pero estarán personalizadas dependiendo de la situación de la sucursal, de tal forma que los banners, logos e imágenes de cada aplicación serán monumentos locales a la zona de la sucursal.

El equipo de trabajo del proyecto está coordinado por María, ella es la encargada del montaje, creación y configuración del servidor web donde irán alojadas las aplicaciones web.

La empresa quiere que las sucursales puedan ser localizadas en Internet mediante URLs tipo:

`www.sucursal-zonaX.empresa-proyecto.com`, donde X puede variar de 1 a 7. Además quiere que si las páginas se buscan sin `www` éstas sigan viéndose, es decir, que `sucursal-zonaX.empresa-proyecto.com` se dirija a la misma página que `www.sucursal-zonaX.empresa-proyecto.com`

La empresa también desea que exista un único panel de control de usuarios, en la URL `www.empresa-proyecto.panel-de-control.com`, de tal forma que según el perfil que posea el usuario podrá ver un contenido u otro. Así, desea que los comerciales tengan la posibilidad de saber que productos y cantidades de los mismos existen en stock. Al panel de control se accede a través de un enlace configurado en cada aplicación.

María se reúne con Juan, el encargado del desarrollo de las aplicaciones web, y con Antonio, que ejerce el rol del usuario destinado a comprobar el buen funcionamiento de las aplicaciones haciendo pruebas con distintos navegadores:

—Pienso —dijo María— que la mejor forma de llevar a buen puerto el proyecto se realiza configurando hosts virtuales en el servidor web Apache y no solamente colgando las aplicaciones web en un directorio raíz común para luego, cada una, disponer de su espacio en una carpeta independiente.

—Sí, —dijo Juan—, además tenemos que tener en cuenta la seguridad del panel de control, deberíamos pensar en el protocolo HTTPS, para asegurarnos que la información vaya cifrada.

—Estoy de acuerdo —afirmó María—. Entonces, Antonio, deberás hacer las pruebas mediante HTTP y HTTPS.

—Vale, de acuerdo —dijo Antonio—.



Anteriormente hemos visto como poder alojar múltiples páginas web en el servidor web Apache, pero todas pertenecientes al mismo sitio/dominio, es decir, todas pertenecientes a **empresa.com**, entonces, ¿no se puede alojar páginas de distintos dominios en el mismo servidor web? La respuesta es que sí, si se puede, ¿cómo?, mediante la configuración de hosts virtuales o **virtualhosts**. Éstos básicamente lo que hacen es permitir que un mismo servidor web pueda alojar múltiples dominios, así configurando hosts virtuales podemos alojar: **empresa1.com**, **empresa2.com**, ..., **empresaN.com** en el mismo servidor web. Cada empresa tendrá su **virtualhost** único e independiente de las demás.

Aunque como se ha comentado anteriormente cada **virtualhost** es único e independiente de los demás, todo aquello que no esté incluido en la definición de cada **virtualhost** se hereda de la configuración principal: **apache2.conf** (`/etc/apache2/apache2.conf`), así, si quieres definir una directiva común en todos los **virtualhost** no debes modificar cada uno de los **virtualhost** introduciendo esa directiva sino que debes definir esa directiva en la configuración principal del servidor web Apache, de tal forma que todos los **virtualhost** heredarán esa directiva, por ejemplo en `apache2.conf` puedes encontrar la directiva **Timeout 300**, que establece la directiva **Timeout** igual a 300 segundos, esto es, indica el número de segundos antes de que se cancele un conexión por falta de respuesta.

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

Para probar el uso de los Host Virtuales en la máquina virtual de Ubuntu en el archivo `/etc/hosts` tenemos que la IP de nuestro servidor es 127.0.0.1. Podemos añadir a nuestro archivo para probar nuestro nuevo servidor con nuestra nueva web:

```
127.0.0.1 ejemplo.com www.ejemplo.com
```

Podemos configurar un servidor DNS con las entradas de dominio necesarias, puedes generar estas entradas modificando el archivo `/etc/hosts`, añadiéndolas al final del mismo. Esto lo veremos o adelante, pero el ejemplo quedaría así:

```
#IP nombre-dominio
192.168.200.250 empresa1.com www.empresa1.com
192.168.200.250 empresa2.com www.empresa2.com
```

Cada campo de cada entrada puede ir separado por espacios o por tabulados.

Estas entradas solamente serán efectivas en el equipo en el que se modifique el archivo `/etc/hosts`. Así debes modificar el archivo `/etc/hosts` en cada equipo que quieres que se resuelven esas entradas.

Caso práctico

Configuración de Virtual Host de Apache 2.2 a Apache 2.4

Paso a paso:

1. Creamos los directorios donde van a estar los archivos de nuestros sitios web dentro de `/var/www`

```
/var/www/uno.com/public_html  
/var/www/dos.com/public_html
```

2. Creamos una página de inicio para cada servidor virtual, por ejemplo:

```
<html>  
<head>  
<title>Bienvenido a uno.com!</title>  
</head>  
<body>  
<h1>Éxito! El Virtual Host uno.com esta funcionando!</h1>  
</body>  
</html>
```

3. Crear nuevos archivos Virtual Host (VH). Apache tiene un archivo por cada VH. El archivo por defecto es `/etc/apache2/sites-available/000-default.conf` podemos hacer una copia y crear dos archivos para cada VH `/etc/apache2/sites-available/uno.com.conf` y `/etc/apache2/sites-available/dos.com.conf`. Por ejemplo:

```
<VirtualHost *:80>  
    ServerAdmin admin@uno.com  
    ServerName uno.com  
    ServerAlias www.uno.com  
    DocumentRoot /var/www/uno.com/public_html  
</VirtualHost>
```

4. Habilitamos los nuevos archivos Virtual Host. Usamos la herramienta de Apache:

```
sudo a2ensite uno.com.conf  
sudo a2ensite dos.com.conf
```

5. Modificamos el fichero `/etc/host`. `sudo nano /etc/hosts`

y añadimos: 192.168.10.2 www.uno.com uno.com

6. Reiniciamos Apache:

```
sudo service apache2 restart
```

7. Probamos resultados: <http://uno.com> y <http://dos.com>

2.1.- Virtualhosts basados en nombre

La IP que debemos poner siempre en la definición de la directiva Virtualhost es la IP del servidor web, en nuestro escenario: IP_Servidor_Web=192.168.200.250 o también, si no tenemos el servidor DNS configurado podemos usar *.

¿Cómo lo haces? Sigues el procedimiento:

1. En la configuración de Apache2 existe un directorio `/etc/apache2/sites-available` donde se definen los virtualhosts, cada virtualhost en un fichero de texto de configuración distinto, así crea los dos ficheros siguientes en la ruta `/etc/apache2/sites-available`.
2. Fichero configuración virtualhost: `empresa1`

```
<VirtualHost *:80>
DocumentRoot /var/www/empresa1/
ServerName www.empresa1.com.
ServerAlias empresa1.com empresa1.es www.empresa1.es
</VirtualHost>
```

3. Fichero configuración virtualhost: `empresa2`

```
<VirtualHost *:80>
DocumentRoot /var/www/empresa2/
ServerName www.empresa2.com.
ServerAlias empresa2.com empresa2.es www.empresa2.es
</VirtualHost>
```

Explicación fichero virtualhost:

`<VirtualHost *:80>` : Inicio etiqueta virtualhost, define la IP del servidor web donde se aloja la página de la empresa, en este caso `empresa1`. Si ponemos * podrá ser una IP que no conocemos o es dinámica. El puerto TCP para el protocolo HTTP por defecto es el 80, definido en la configuración principal del servidor, mediante la directiva `Listen`, por lo cual no es necesario ponerlo. Se pueden usar varias directivas `Listen` para especificar varias direcciones y puertos de escucha. El servidor responderá a peticiones de cualquiera de esas direcciones y puertos. Por ejemplo, para hacer que el servidor acepte conexiones en los puertos 80 y 8080, usa:

```
Listen 80
Listen 8080
```

Para hacer que el servidor acepte conexiones en dos direcciones IP y puertos diferentes, usa:

```
Listen 192.168.200.250:80
Listen 192.168.200.251:8080
```

- ✔ **DocumentRoot /var/www/empresa1/** : Definición de la ruta donde está alojada la página web en el servidor, en este caso: `/var/www/empresa1/` mediante la directiva `DocumentRoot`.
- ✔ **ServerName www.empresa1.com** : Definición del nombre DNS que buscará la página alojada en la ruta anterior del servidor mediante la directiva `ServerName`. Es el nombre que escribes en el navegador para visitar la página.
- ✔ **ServerAlias empresa1.com** : La directiva `ServerAlias` permite definir otros nombres DNS para la misma página.
- ✔ **</VirtualHost>** : Fin de la etiqueta `VirtualHost`: fin de la definición de este virtualhost para la empresa1.

Autoevaluación

Si deseas que tu servidor web ofrezca en la misma IP las URL:

`www.sucursal-zona2.empresa-proyecto.com`, `sucursal-zona2.empresa-proyecto.com`

`www.empresa-proyecto.panel-de-control.com`.

donde las 2 primeras identifican el mismo sitio web y la última otro totalmente distinto. Entonces, ¿podrías utilizar para definir los virtualhosts?

- ☐ Un solo fichero.
- ☐ Dos ficheros.
- ☐ No se pueden utilizar virtualhosts, debido a que los dominios son distintos.
- ☐ Incorrecta la pregunta ya que las URL están mal definidas, no pueden contener el carácter guión.

No, ya que las 3 URL definen 2 sitios totalmente distintos.

Si, ya que las 3 URL definen 2 sitios totalmente distintos.

No, porque ya que los dominios son distintos y queremos que apunten a la misma IP del servidor web debemos utilizar virtualhosts.

Falso, si permiten el carácter guión.

Solución

1. [Incorrecto](#) ([Retroalimentación](#))
2. [Opción correcta](#) ([Retroalimentación](#))
3. [Incorrecto](#) ([Retroalimentación](#))
4. [Incorrecto](#) ([Retroalimentación](#))

2.2.- Virtualhosts basados en IP

La IP que debemos poner ahora en la definición de la directiva Virtualhost cambia, cada IP corresponde a una interfaz de red del servidor web, en nuestro escenario: 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.

¿Cómo lo haces? Sigues el mismo procedimiento usado para los virtualhost basado en nombre, únicamente se diferencia en los ficheros a crear para los virtualhost, así:

1. En la configuración de Apache2 existe un directorio `/etc/apache2/sites-available` donde se definen los virtualhost, cada virtualhost en un fichero de texto de configuración distinto, así crea los dos ficheros siguientes en la ruta `/etc/apache2/sites-available`.
2. Fichero configuración virtualhost: empresa3

```
<VirtualHost IP1_Servidor_Web:80>
DocumentRoot /var/www/empresa3/
ServerName www.empresa3.com.
ServerAlias empresa3.com empresa3.es www.empresa3.es
</VirtualHost>
```

3. Fichero configuración virtualhost: empresa4

```
<VirtualHost IP2_Servidor_Web:80>
DocumentRoot /var/www/empresa4/
ServerName www.empresa4.com.
ServerAlias empresa4.com empresa4.es www.empresa4.es
</VirtualHost>
```

Explicación fichero virtualhost:

<VirtualHost>: Inicio etiqueta virtualhost, define la IP1 del servidor web donde se aloja la página de la empresa, en este caso empresa3. El puerto TCP por defecto es el 80, definido en la configuración principal del servidor, mediante la directiva Listen, por lo cual no es necesario ponerlo. Se pueden usar varias directivas Listen para especificar varias direcciones y puertos de escucha. El servidor responderá a peticiones de cualquiera de esas direcciones y puertos. Por ejemplo, para hacer que el servidor acepte conexiones en los puertos 80 y 8080, usa:

```
<VirtualHost IP1_Servidor_Web:80>
DocumentRoot /var/www/empresa3/
ServerName www.empresa3.com.
ServerAlias empresa3.com empresa3.es www.empresa3.es
</VirtualHost>
```

Para hacer que el servidor acepte conexiones en dos direcciones IP y puertos diferentes, usa:

```
<VirtualHost IP2_Servidor_Web:80>
DocumentRoot /var/www/empresa4/
ServerName www.empresa4.com.
ServerAlias empresa4.com empresa4.es www.empresa4.es
</VirtualHost>
```

- ✔ **DocumentRoot /var/www/empresa3/**: Definición de la ruta donde está alojada la página web en el servidor, en este caso: `/var/www/empresa3/` mediante la directiva **DocumentRoot**.
- ✔ **ServerName www.empresa3.com** : Definición del nombre DNS que buscará la página alojada en la ruta anterior del servidor mediante la directiva **ServerName**. Es el nombre que escribes en el navegador para visitar la página.
- ✔ **ServerAlias empresa3.com** : La directiva **ServerAlias** permite definir otros nombres DNS para la misma página.
- ✔ **</VirtualHost>**: Fin de la etiqueta VirtualHost: fin de la definición de este virtualhost para la empresa3.

Para saber más

En el siguiente enlace encontrarás información sobre la directiva **RewriteRule**, la cual te puede evitar tener que utilizar la directiva **ServerAlias**, pues te permite reescribir las direcciones URL.

[Directiva RewriteRule](#)



3.- Módulos.

Caso práctico

Está bien -pensó María-. Manos a la obra. Debo montar un nuevo servidor web Apache y necesito... veamos:

- ✓ Que varias aplicaciones web atiendan en el mismo dominio, tal que:
`sucursal-zonaX.empresa-proyecto.com`, `www.sucursal-zonaX.empresa-proyecto.com`,
- ✓ Un único panel de control de usuarios, en la URL `www.empresa-proyecto.panel-de-control.com`,
- ✓ También soporte SSL para cifrado.
- ✓ Soporte para páginas dinámicas mediante PHP.
- ✓ Y soporte para control de usuarios LDAP.



Uhm..., ya lo tengo claro. Tengo que montar Apache con varios módulos, así primero instalaré Apache, luego verificaré que módulos vienen instalados por defecto, si me conviene dejarlos instalados o no, igual tengo que desinstalar alguno y tendré que investigar cuales son los módulos nuevos a instalar.

Muy bien, pues lo dicho: ¡Manos a la obra!

La importancia de un servidor web radica en su: estabilidad, disponibilidad y escalabilidad. Es muy 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 nos otorga el servidor web Apache mediante sus módulos sea uno de los servidores web más manejables y potentes que existen: que necesito soporte SSL pues módulo SSL, que necesito soporte PHP pues módulo PHP, que necesito soporte LDAP pues módulo LDAP, que necesito...

En Debian, y derivados, existen dos comandos fundamentales para el funcionamiento de los módulos en el servidor web Apache: `a2enmod` y `a2dismod`.

- ✓ `a2enmod`: Utilizado para habilitar un módulo de apache. Sin ningún parámetro preguntará que módulo se desea habilitar. Los ficheros de configuración de los módulos disponibles están en `/etc/apache2/mods-available/` y al habilitarlos se crea un enlace simbólico desde `/etc/apache2/mods-enabled/`.
- ✓ `a2dismod`: Utilizado para deshabilitar un módulo de Apache. Sin ningún parámetro preguntará que módulo se desea deshabilitar. Los ficheros de configuración de los módulos disponibles están en `/etc/apache2/mods-available/` y al deshabilitarlos se elimina el enlace simbólico desde `/etc/apache2/mods-enabled/`.
- ✓ Si no dispones de esos comandos para poder habilitar y deshabilitar módulos Apache simplemente haces lo que ellos: crear los enlaces simbólicos correspondientes desde `/etc/apache2/mods-enabled/` hasta `/etc/apache2/mods-available/`.

Para consultar los módulos instalados:

```
sudo apachectl -l
```

`a2ensite` es un comando (en Debian y derivados) para habilitar configuraciones de "sitios web" en Apache2. Los ficheros de configuración de los "sitios web" disponibles (normalmente son configuraciones de hosts virtuales) están en `/etc/apache2/sites-available/` y al habilitarlos se crea un enlace simbólico desde `/etc/apache2/sites-enabled/`

En el archivo `apache2.conf` nos encontramos:

```
# Include module configuration:  
IncludeOptional mods-enabled/*.load  
IncludeOptional mods-enabled/*.conf
```

En la carpeta `mods-enabled` solo hay enlaces simbólicos a los archivos en `mods-available`. En el directorio `/etc/apache2/mods-enabled` podemos ver que hay dos archivos asociados a cada módulo activo. Los archivos `.load` incluyen la directiva para que se cargue el módulo en cuestión mientras que los archivos `.conf` incluyen directivas de configuración que únicamente se aplicarán si se carga un módulo determinado. Esto se consigue con la directiva `IfModule` que funciona como un `if` de programación

Debes conocer

Puedes consultar más información en la documentación de Apache sobre módulos.

[Módulos](#)

La instalación o desinstalación de un módulo no implica la desinstalación de Apache o la nueva instalación de Apache perdiendo la configuración del servidor en el proceso, simplemente implica la posibilidad de poder trabajar en Apache con un nuevo módulo o no.

3.1.- Operaciones sobre módulos.

Citas para pensar

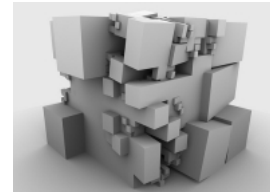
Confucio: "Me lo contaron y lo olvidé. Lo vi y lo entendí. Lo hice y lo aprendí."

Los módulos de Apache puedes instalarlos, desinstalarlos, habilitarlos o deshabilitarlos, así, puedes tener un módulo instalado pero no habilitado. Esto quiere decir que aunque instales módulos hasta que los habilites no funcionarán.

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

Operaciones sobre módulos Apache en un GNU/Linux Debian

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 puedes ejecutar el comando **a2enmod** sin parámetros. La ejecución de este comando ofrecerá una lista de módulos a habilitar, escribes el módulo en cuestión y el módulo se habilitará. Del mismo modo para deshabilitar un módulo Apache, en Debian, puedes ejecutar el comando **a2dismod** sin parámetros. La ejecución de este comando ofrecerá una lista de módulos a deshabilitar, escribes el módulo en cuestión y el módulo se deshabilitará.

Una vez habilitado o deshabilitado los módulos Apache sólo reconocerá estos cambios cuando recargues su configuración, con lo cual debes ejecutar el comando: **/etc/init.d/apache2 restart**

Si la configuración es correcta y no quieres reiniciar Apache puedes recargar la configuración mediante el comando: **/etc/init.d/apache2 reload**.

Si no dispones de los comandos **a2enmod** y **a2dismod** puedes habilitar y deshabilitar módulos Apache creando los enlaces simbólicos correspondientes desde **/etc/apache2/mods-enabled/** hasta **/etc/apache2/mods-available/**, por ejemplo si quisieras habilitar el módulo ssl:

1. Te sitúas en el directorio **/etc/apache2/mods-available**.

```
cd /etc/apache2/mods-available
```

2. Verificas que el módulo aparece en esta ruta y por lo tanto está instalado
ls ssl.*

Este comando debe listar dos ficheros: **ssl.conf** (la configuración genérica del módulo) y **ssl.load** (la librería que contiene el módulo a cargar)

3. Creas el enlace simbólico para habilitar el módulo:

```
ln -s ../mods-enabled/ssl.conf /ssl.conf
```

```
ln -s ../mods-enabled/ssl.load /ssl.load
```

Estos comandos crean los enlaces **/etc/apache2/mods-enabled/ssl.conf** y **/etc/apache2/mods-enabled/ssl.load** que apuntan a los ficheros **/etc/apache2/mods-available/ssl.conf** y **/etc/apache2/mods-available/ssl.load** respectivamente.

4. Recargas la configuración de Apache:

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

5. El módulo ssl ya está habilitado.

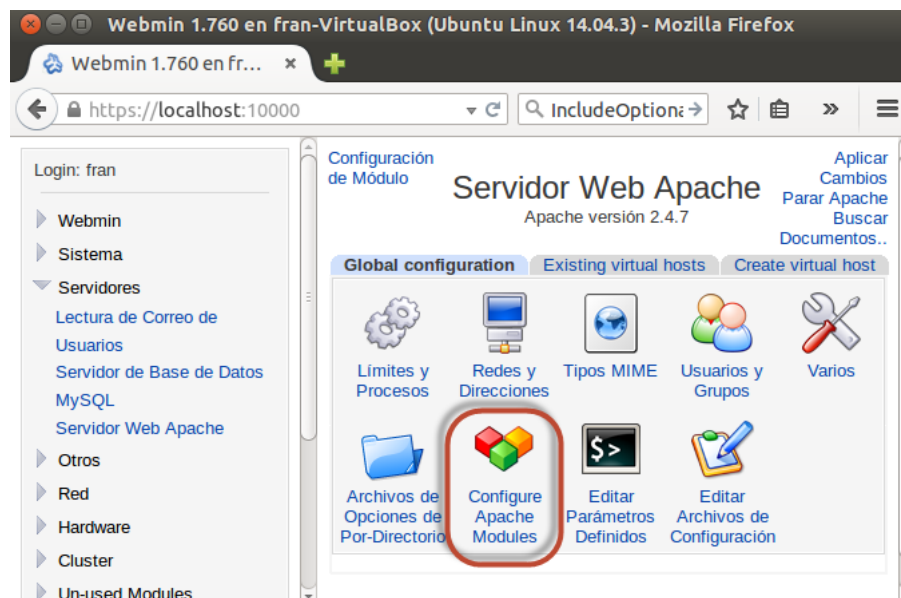
Y si quisieras deshabilitarlo, simplemente eliminas en **/etc/apache2/mods-enabled** los enlaces simbólicos creados, así si quisieras deshabilitar el módulo ssl ejecutarías el siguiente comando:

```
rm -f /etc/apache2/mods-enabled/ssl.*
```

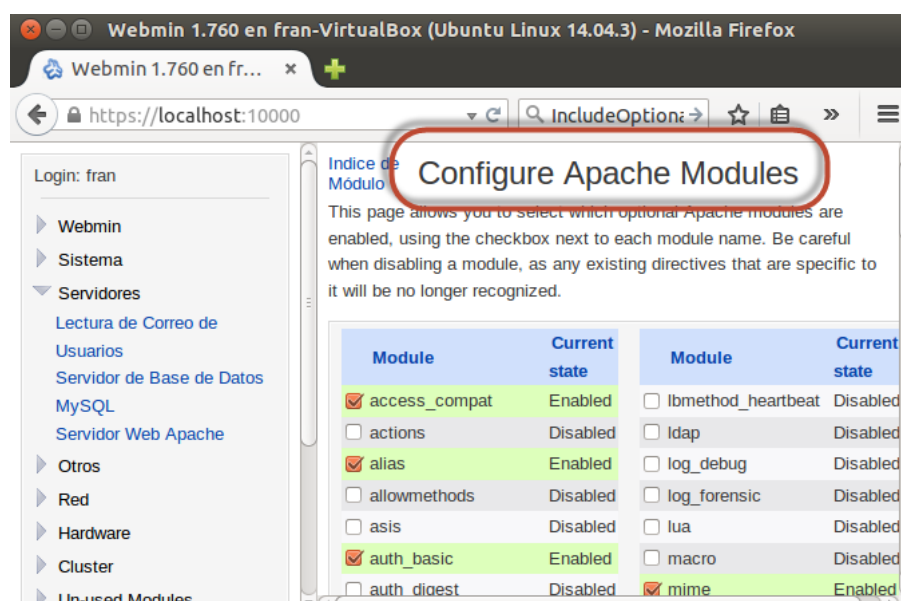
Por último, no te olvides recargar la configuración de Apache: **/etc/init.d/apache2 restart**

Caso práctico

Habilitar módulos con herramienta gráfica Webmin:



Activamos o desactivamos y guardamos la configuración:



4.- Acceso a carpetas seguras.

Caso práctico

En el transcurso del proyecto sobre aplicaciones web de varias sucursales para una empresa en las oficinas de la empresa BK Programación tuvo lugar la siguiente charla:

Bien, -le dijo Ana a Ada-, ya tenemos casi configurado el servidor web Apache.

- ¿Entonces?- preguntó Ada-

-Nos falta la configuración de la navegación de forma segura, para que la comunicación viaje cifrada.

-¿Os llevará mucho tiempo?

-Bueno...



Citas para pensar

Miguel de Icaza: "Depende qué tan hombre eres."

¿Todas las páginas web que están alojadas en un sitio deben ser accesibles por cualquier usuario? ¿Todas las accesibles deben enviar la información sin cifrar, en texto claro? ¿Es necesario que todo el trasiego de información navegador-servidor viaje cifrado?

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.

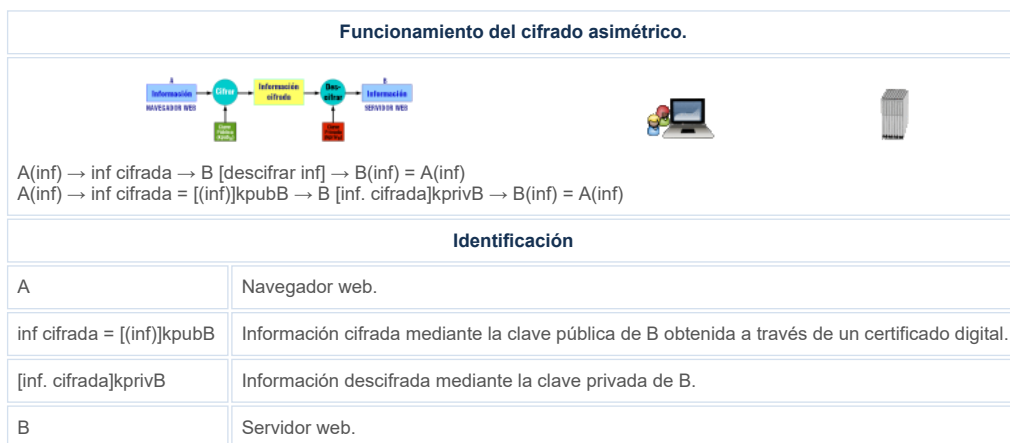
El cifrado al que nos referimos es el cifrado de clave pública o asimétrico: **clave pública (kpub)** y **clave privada (kpriv)**. La **kpub** interesa publicarla para que llegue a ser conocida por cualquiera, la **kpriv** no interesa que nadie la posea, solo el propietario de la misma. Ambas son necesarias 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**.

Cifrado de clave pública o asimétrico

[Resumen textual alternativo](#)

En el cifrado asimétrico podemos 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:

Funcionamiento del cifrado asimétrico.



Como ves, **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 garantizas la autenticidad de B? Pues ya que supones que B es quien dice ser mediante un certificado digital, debes confiar en ese certificado, así ¿quién emite certificados digitales de confianza? Igual que el DNI es emitido por una entidad certificadora de confianza, el Ministerio del Interior, en Internet existen autoridades de certificación(CA ó AC) que aseguran la autenticidad del certificado digital, y así la autenticidad de B, como: [Verisign](#) y [Thawte](#). Pero, como ya hemos comentado el Servidor Web Apache permite ser CA, por lo que tienes la posibilidad de crear tus propios certificados digitales, ahora bien, ¿el navegador web(A) confiará en estos certificados? Pues, en principio no, por lo que los navegadores avisarán que la página a la cuál intentas acceder en el servidor web representa un peligro de seguridad, ya que no existe en su lista de autoridades certificadoras de confianza. En determinados casos, por imagen, puede ser un problema, pero si la empresa posee una entidad de importancia reconocida o el sitio es privado y no público en Internet o sabes el riesgo que corres puedes aceptar la comunicación y el flujo de información viajará cifrado.

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, individuo o máquina, por ejemplo un servidor web, y es emitido por autoridades en las que pueden confiar los usuarios. Estas 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 te suele interesar solamente la fiabilidad del servidor, para saber que te conectas con el servidor que quieres y no con otro servidor -supuestamente cuando tú te conectas con el navegador al servidor eres tú y no otra persona la que establece la conexión-.** 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.

Existen varios formatos para certificados digitales, pero los más comúnmente empleados se rigen por el estándar [UIT-T X.509](#). El certificado X.509 contiene los siguientes campos: versión, nº de serie del certificado, identificador del algoritmo de firmado, nombre del emisor, período de validez, nombre del sujeto, información de clave pública del sujeto, identificador único del emisor, identificador único del sujeto y extensiones.



4.2.- Módulo ssl para apache.

Reflexiona

Todos los días los bancos efectúan transferencias bancarias, así como también aceptan conexiones a sus páginas web para ofrecer su servicio online. ¿Qué pasaría si cualquiera pudiese interceptar una comunicación bancaria de ese tipo? ¿Sería interesante cifrar la información efectuada antes y durante la conexión bancaria?

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

El **módulo ssl** es quien permite cifrar la información entre navegador y servidor web. En la instalación por defecto éste módulo no viene activado, así que debes ejecutar el siguiente comando para poder activarlo: `a2enmod ssl`

Este módulo proporciona SSL v2/v3 y TLS v1 para el Servidor Apache HTTP; y se basa en OpenSSL para proporcionar el motor de la criptografía.



En el siguiente enlace puedes encontrar más información sobre el módulo ssl

[Módulo ssl](#)

Ejercicio resuelto

¿Cómo harías, en Ubuntu 14.04, para deshabilitar el módulo ssl de Apache?

Mediante el comando `a2dismod ssl` y recargando el servicio Apache: `/etc/init.d/apache2 reload` ó `/etc/init.d/apache2 restart`.

Y ¿cómo lo harías si no dispones del comando para Ubuntu?

Quitando los enlaces existentes en `mods-enabled` que apunten a los archivos ssl correspondientes de la carpeta `mods-available`, por ejemplo en un Debian 6 eliminarías los archivos: `ssl.conf` y `ssl.load` situados en `/etc/apache2/mods-enabled/` y recargando el servicio Apache: `/etc/init.d/apache2 reload` ó `/etc/init.d/apache2 restart`

4.3.- Crear un servidor virtual seguro en Apache (I).

Apache utiliza un módulo específico basado en un proyecto que se llama OpenSSL que además implementa también TLS. Ahora el servidor debería estar escuchando tanto el puerto 80 (http) como en el 443 (https). Si miramos el archivo de configuración de puertos:

```
gedit /etc/apache2/ports.conf
```

En Ubuntu, Apache posee por defecto en su instalación el fichero </etc/apache2/sites-available/default-ssl.conf> (2.62 KB), que contiene la configuración por defecto de SSL. En su contenido podemos ver las siguientes líneas:

```
SSLEngine on
SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem
SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
```

donde,

SSLEngine on : Activa o desactiva SSL

SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem : Certificado digital del propio servidor Apache

SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key : Clave privada del servidor Apache.

Esas líneas lo que quieren decir es que Apache permite conexiones SSL y posee un certificado digital autofirmado por si mismo -ya que Apache actúa como entidad certificadora-.

Cuando activaste el módulo ssl, mediante el comando `a2enmod ssl` permitiste que Apache atiende el protocolo SSL. Tenemos que activar el sitio por defecto `default-ssl`:

```
sudo a2ensite default-ssl
sudo service apache2 restart
```

Así, si ahora lanzas el navegador **Firefox** con la dirección de tu servidor web Apache mediante el protocolo HTTPS, verás una imagen similar a la siguiente:



Lo que indica que el certificado digital del servidor no viene firmado por una **AC** contenida en la lista que posee el navegador, sino por el mismo Apache. Si lo compruebas haciendo clic en **Detalles Técnicos** verás algo similar a:



Ahora tienes dos opciones: Confiar en el certificado o no.

- ✓ Si confías haces clic en **Entiendo los riesgos y Añadir excepción...**

Una vez que confías puedes, antes de **Confirmar excepción de seguridad**, ver el contenido del certificado. Si estás de acuerdo la comunicación se establece y la información viaja cifrada.

- ✓ Si no confías haces clic en **¡Sácame de aquí!**

¿Pero...? Como eres AC puedes firmar certificados e incluso puedes generar también tu propio certificado autofirmado similar al que viene por defecto en Apache.

Hay que tener en cuenta que la negociación SSL es dependiente totalmente de la IP, no del nombre del sitio web, así no puedes servir distintos certificados en una misma IP.

Debes conocer

No es una buena opción dejar abierta la posibilidad de acceder al mismo sitio mediante una conexión segura y una insegura. Por ello si habilitamos un sitio con HTTPS no deberíamos tener activo el equivalente con HTTP como ahora mismo tenemos con `'localhost'` (los dos sitios por defecto). Lo correcto sería deshabilitar el sitio por defecto.

4.3.1. Pasos para crear el servidor virtual seguro

Recomendación

Debemos tener un sitio virtual creado y habilitado, con su directorio correspondiente, el servidor web debe 'escuchar' por el puerto 443, y si queremos lo podemos asociar a través del DNS una dirección IP.

En este ejemplo tenemos el sitio 'tato.com' asociado a la dirección 192.168.10.2

Podemos deshabilitar el sitio web seguro por defecto:

```
sudo a2dissite default-ssl
```

1. Debemos tener un certificado antes de poder usar SSL. No vamos a solicitar uno a una entidad CA, crearemos uno autofirmado aprovechando OpenSSL que tendremos instalado en nuestro sistema.

`sudo apt-get install openssl`

2. El nombre de dominio con el que vamos a trabajar es www.tato.com que tendremos configurado como un host virtual con nombre. Ahora necesitamos una clave privada, lo hacemos con el comando 'genrsa' lo haremos sin contraseña para que no sea necesario introducirla cada vez que el certificado se use.

```
root@fran-VirtualBox: /home/fran# openssl genrsa -out clavepru.key 2048
Generating RSA private key, 2048 bit long modulus
.....+++
....+++
e is 65537 (0x10001)
root@fran-VirtualBox: /home/fran#
```

3. Ahora generamos una petición para nuestro certificado. Esta petición es la que deberíamos enviar a la CA para obtener un certificado firmado por ellos. Luego esperaríamos a que lo firmaran y nos enviaran el certificado para instalarlo. Nosotros usaremos uno autofirmado. Ahora nos irá solicitando una serie de datos que vamos rellenando.

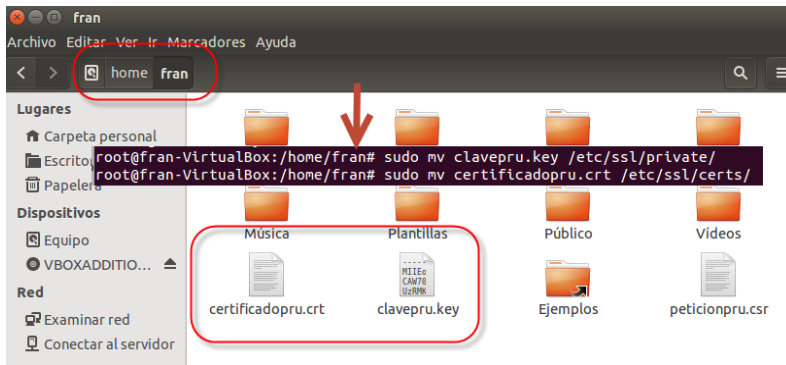
```
root@fran-VirtualBox: /home/fran# openssl req -new -key clavepru.key -out peticionpru.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Cartagena
Locality Name (eg, city) []:Cartagena
Organization Name (eg, company) [Internet Widgits Pty Ltd]:CarlosIII
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:admin@admin.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
root@fran-VirtualBox: /home/fran#
```

4. Ahora obtenemos el certificado autofirmado. Usará el formato X.509 y tendrá una validez de un año.

```
root@fran-VirtualBox: /home/fran# openssl x509 -req -days 365 -in peticionpru.csr
-signkey clavepru.key -out certificadoprpu.crt
Signature ok
subject=/C=ES/ST=Cartagena/L=Cartagena/O=CarlosIII/emailAddress=admin@admin.com
Getting Private key
root@fran-VirtualBox: /home/fran#
```

5. Los archivos generados están en el directorio donde hemos trabajado, hay que colocarlos en los directorios adecuados.



6. El sitio virtual por nombre y su fichero de configuración en `/etc/apache2/sites-available/`:

```

tato.com.conf x
<IfModule mod_ssl.c>
NameVirtualHost 192.168.10.2:443
<VirtualHost 192.168.10.2:443>
    ServerAdmin admin@tato.com
    ServerName tato.com
    ServerAlias www.tato.com
    DocumentRoot /var/www/tato.com/public_html
    ErrorLog ${APACHE_LOG_DIR}/error_tato.com.log
    CustomLog ${APACHE_LOG_DIR}/tato.com_access.log combined

    <Directory /var/www/tato.com/public_html>
        DirectoryIndex index.html
        Options -Indexes
        AllowOverride None
        Require all granted
    </Directory>
    LogLevel warn

    # Esta es la parte de SSL
    SSLEngine on
    SSLCertificateFile /etc/ssl/certs/certificadopru.crt
    SSLCertificateKeyFile /etc/ssl/private/clavepru.key
    # Recuerda que lo siguiente es para mantener la compatibilidad con ciertas
    # versiones de Microsoft Internet Explorer
    BrowserMatch "MSIE [2-6]" \
        nokeepalive ssl-unclean-shutdown \
        downgrade-1.0 force-response-1.0
    BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown
</VirtualHost>
</IfModule>

```

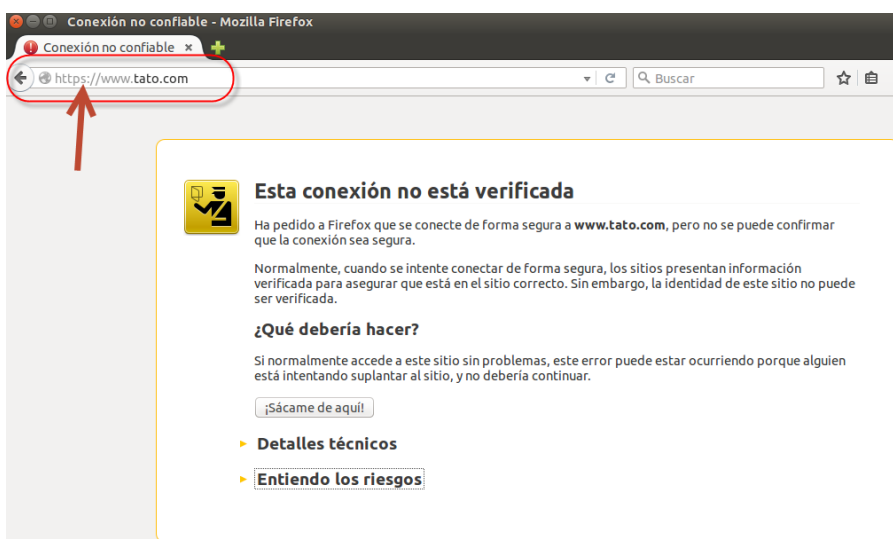
7. Habilitamos el sitio y reiniciamos Apache:

```

root@fran-VirtualBox:/home/fran# sudo a2ensite tato.com
Site tato.com already enabled
root@fran-VirtualBox:/home/fran# service apache2 reload
* Reloading web server apache2
*
root@fran-VirtualBox:/home/fran#

```

8. Comprobamos que funciona:



4.4.- Comprobar el acceso seguro al servidor.

Citas para pensar

Proverbio español: "La manera de estar seguro es no sentirse nunca seguro."

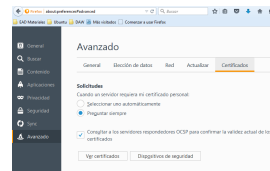
A continuación una serie de actuaciones que te servirán para comprobar que el acceso seguro que estableces con el servidor seguro es el esperado:

- ✓ Siempre que te conectes mediante SSL a un página web y el certificado no sea admitido, debes ver los campos descriptivos del certificado antes de generar la excepción que te permita visitar la página.
- ✓ Debes comprobar en el certificado si la página a la que intentas acceder es la misma que dice el certificado.
- ✓ Típicamente en los navegadores, si no está configurado lo contrario, cuando accedes mediante cifrado SSL a una página web puedes ver en algún lugar del mismo un icono: un candado, por lo cual debes verificar su existencia para asegurarte que estás accediendo por https.

Incluso si el certificado pertenece a alguna AC que el navegador posee en su lista de AC puedes ver en la barra de direcciones indicaciones del tipo de certificado con el que se cifra la comunicación.

- ✓ Revisar la lista de certificados admitidos que posee tu navegador. En **Firefox**, versión > 41.x , donde x es el número de revisión de la versión 41, puedes verlas dirigiéndote por las pestañas a:

Editar → Preferencias → Avanzado → Cifrado → Ver certificados



Puedes **Importar/Exportar** certificados en los navegadores, con lo cual los puedes llevar a cualquier máquina. Esto es muy útil cuando necesitas un certificado personal en máquinas distintas.

Podemos ver en este vídeo como comprobar en el navegador Firefox (versión 41.0) que el certificado de seguridad autofirmado que hemos creado por nosotros está instalado.

Para saber más

En el siguiente enlace encontrarás información muy interesante, amena y explicativa sobre la seguridad de la información y el cifrado.

[Enciclopedia de la seguridad de la información.](#)

5.- Autenticación y control de acceso.

Caso práctico

La reunión tuvo lugar.

El equipo de BK Programación destinado al proyecto de aplicaciones web para varias sucursales de una empresa llegó a un acuerdo para la autenticación y el control de acceso sobre la aplicación de panel de control. Se barajaron varias alternativas: usuarios del sistema, ficheros de usuarios, base de datos SQL y LDAP. Al final se decantaron por dos opciones: ficheros de usuarios para el estado de pruebas y LDAP para la aplicación definitiva, con lo cual establecieron el siguiente protocolo de actuación:

1. En la aplicación de desarrollo montada por María se realizarán las pruebas, siendo los encargados de las mismas Antonio y Carlos.
2. El diseño web de la aplicación recaerá en Ana: banners, logos ...
3. Juan se dedicará a la programación del panel de control: autenticación por medio de LDAP
4. La encargada de montar el servicio LDAP, integrarlo en Apache y conseguir el control de acceso fue María.

Ante la espera que María instale y configure Apache con LDAP, y con ello imposibilidad de probar la autenticación por LDAP, María crea un fichero de usuarios para autenticarse en la aplicación y todos empiezan a trabajar en el resto de las cosas.



Puede que interese impedir el acceso a determinadas páginas ofrecidas por el servidor web, así: ¿crees que a una empresa le interesaría que cualquiera tuviera acceso a determinada información confidencial?, o puede que interese controlar el acceso hacia un servicio a través de la web, como el correo electrónico. Para este tipo de casos tenemos que pensar en la autenticación y el control de acceso.

Cuando nos autenticamos en una web suele transferirse 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](http://www.OpenLDAP.org) una de las alternativas más empleadas. Esto lo trabajaremos en unidades posteriores.

En este apartado nos vamos a centrar en las grandes posibilidades que nos ofrece el servidor web Apache respecto al control de acceso y la autenticación.

Para saber más

Puedes visitar el enlace de wikipedia [AAA](#) donde encontrarás más información referente a la autenticación:

[AAA](#)

A modo introductorio:

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 empleas este método que lo hagas combinado con conexión SSL (HTTPS).

En la autenticación HTTP Basic es muy típico utilizar archivos `.htaccess` en los directorios que queremos controlar el acceso. Puedes encontrar un ejemplo sobre basic con https en el archivo [virtualhost-ssl-basic](#) (0.56 KB) y un ejemplo sobre `.htaccess` en el archivo [htaccess](#) (0.26 KB).

Para usar archivos `.htaccess`, necesitas tener una configuración en el servidor que permita poner directivas de autenticación en estos archivos, mediante la directiva `AllowOverride`, así: `AllowOverride AuthConfig`

Para saber más

Puedes visitar el siguiente enlace donde encontrarás más información referente a la autenticación http basic:

[Autenticación, autorización y control de acceso \(Apache 2.0\)](#)

[Autenticación y Control de Acceso \(Apache 2.4\)](#)

También se puede controlar el acceso mediante IP. Puedes encontrar un ejemplo en el archivo [virtualhost-control-por-IP](#) (0.37 KB).

5.1. Control de Acceso

Según la [documentación de Apache 2.4](#): "El control de acceso se refiere a cualquier medio de controlar el acceso a cualquier recurso". En el control de acceso están implicados distintos módulos de Apache, los más importantes son [mod_auth_core](#) y [mod_auth_host](#). Además también se usan [mod_setenvif](#) y [mod_rewrite](#) que veremos más adelante.

El control de acceso está ligado a la autenticación y autorización de los accesos a los recursos por parte de los usuarios.

La directiva [Require](#) ofrece la variedad de formas para permitir o denegar el acceso a los recursos. La unión y mezcla de las directivas [RequireAll](#) [RequireAny](#) y [RequireNone](#) permitirán crear cualquier política de acceso. Las veremos en el siguiente apartado.

Las directivas [Allow](#) [Deny](#) y [Order](#) proporcionadas por el módulo [mod_access_compat](#) están en desuso y desaparecerán en versiones futuras.

El uso de la directiva es:

```
Require host address
Require ip ip.address
```

En la primera *forma*, la dirección es un nombre de dominio completo (o un nombre de dominio parcial); podemos proporcionar múltiples direcciones o nombres de dominio.

En la segunda forma, *ip.address* es una dirección IP, una dirección IP parcial, un par de red/máscara de red. Se pueden utilizar direcciones IPv4 o IPv6.

Podemos utilizar *not* para negar a un host, dirección IP o nombre de dominio en particular el acceso a un recurso. El uso de las directivas [RequireAll](#) [RequireAny](#) y [RequireNone](#) puede ser utilizado para hacer cumplir conjuntos más complejos de requisitos.

```
<RequireAll>
Require all granted
Require not ip 10.252.46.165
Require not host host.example.com
Require not host gov
</RequireAll>
```

5.1.2. Módulo rewrite

Mediante el uso del módulo `mod_rewrite` podemos controlar el acceso según criterios arbitrarios. Por ejemplo si queremos denegar el acceso durante el periodo de las ocho de la tarde a las seis de la mañana, escribiremos:

```
RewriteEngine On
RewriteCond %{TIME_HOUR} >20 [OR]
RewriteCond %{TIME_HOUR} <07
RewriteRule ^/fridge - [F]
```

El uso de este método se basa en las directivas `RewriteCond` y `RewriteRule` pero no profundizamos más en ellos.

Para saber más

Directivas [rewritecond](#) y [rewriterule](#)

5.2. Autenticación y Autorización

Reflexiona

La autenticación es cualquier proceso por el cual se comprueba que alguien es quien dice ser. La autorización es cualquier proceso por el cual se permite a alguien estar donde quieren ir, o para tener información que quieren tener.

Recomendación

Módulos y Directivas que participan en el proceso de autenticación y autorización. Por lo general, hay que elegir al menos un módulo de cada grupo para su aplicación.

- Tipo de autenticación (ver directiva [AuthType](#))
 - [mod_auth_basic](#)
 - [mod_auth_digest](#)
- Proveedor de autenticación (ver directivas [AuthBasicProvider](#) y [AuthDigestProvider](#))
 - [mod_authn_anon](#)
 - [mod_authn_dbd](#)
 - [mod_authn_dbm](#)
 - [mod_authn_file](#)
 - [mod_authnz_ldap](#)
 - [mod_authn_socache](#)
- Autorización (ver directiva [Require](#))
 - [mod_authnz_ldap](#)
 - [mod_authz_dbd](#)
 - [mod_authz_dbm](#)
 - [mod_authz_groupfile](#)
 - [mod_authz_host](#)
 - [mod_authz_owner](#)
 - [mod_authz_user](#)

Además de estos módulos, también tenemos ***mod_authn_core*** y ***mod_authz_core***. Estos módulos implementan directivas básicas que son fundamentales para todos los módulos de autenticación. El módulo ***mod_authnz_ldap*** es tanto un proveedor de autenticación y autorización. El módulo ***mod_authz_host*** proporciona la autorización y control de acceso basado en el nombre de host, la dirección IP o las características de la solicitud, pero no es parte del sistema de proveedor de autenticación.

Si en nuestro sitio web tenemos información sensible o destinada a ser sólo para un pequeño grupo de personas, lo que vamos a ver a continuación nos ayudará a asegurarnos de que las personas que ven esas páginas son las personas adecuadas.

Si queremos que los datos realmente estén en un sitio seguro, tendremos que usar [mod_ssl](#) además de cualquier autenticación.

Requisitos previos

Las directivas para la autenticación tendrán que ir ya sea en el archivo de configuración del servidor (por lo general en una sección **<Directory>**), o en archivos de configuración **.htaccess**.

Si vamos a utilizar archivos **.htaccess**, tendremos que tener una configuración de servidor que permite poner las directivas de autenticación en estos archivos. Esto se hace con la directiva **AllowOverride**, que especifica que las directivas, en su caso, puede ser puestas en archivos de configuración por directorio.

Necesitaremos una instrucción como esta:

```
AllowOverride AuthConfig
```

Si ponemos las directivas directamente en el archivo de configuración del servidor principal, necesitaremos tener permiso de escritura en ese archivo.

Tendremos que asegurarnos de que los módulos ***mod_authn_core*** y ***mod_authz_core*** están activos en nuestro servidor. Como ya hemos dicho ambos módulos proporcionan directrices básicas y funcionalidad que son fundamentales para la configuración y el uso de la autenticación y autorización en el servidor web.

5.2.1. Proteger acceso a un directorio con contraseña

En primer lugar, es necesario crear un archivo de contraseñas. Como hacerlo variará dependiendo de qué proveedor de autenticación se ha elegido. Para empezar, vamos a utilizar un archivo de contraseñas de texto.

Este archivo debe estar en algún lugar no accesible desde la web. Esto es para que la gente no puede descargar el archivo de contraseñas. Por ejemplo, si se sirven los documentos desde `/var/www/` una idea puede ser poner el archivo de contraseñas en `/var/passwd`

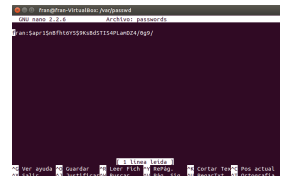
Creamos el archivo usando la utilidad `htpasswd` que viene con Apache. Se encuentra en el directorio donde se ha instalado Apache. Si no está instalada podemos hacerlo:

```
sudo apt-get install apache2-utils
```

Uso de htpasswd

Para crear el archivo, escribimos la siguiente instrucción para crear un usuario llamado *fran* con la contraseña que introducimos al crearlo:

```
sudo htpasswd -c /var/passwd/passwords fran
```



Ahora por medio de las dos posibilidades que hablamos: editando el archivo `.conf` o usando un archivo `.htaccess`. Por ejemplo, si queremos proteger el directorio `/var/www/html/prueba` usamos las siguientes directivas en la sección Directory dentro `/sites-available/000-default.conf`:

```
<Directory "/var/www/html/prueba">
    AuthType Basic
    AuthName "Restricted Files"
    # (Following line optional)
    AuthBasicProvider file
    AuthUserFile "/var/passwd/passwords"
    Require user fran
</Directory>
```



- `AuthName` le indica al usuario qué hacer. Es un mensaje para el usuario.
- `AuthType` es el tipo de autenticación que usaremos; `http` solo admite `Basic`. La otra opción que existe es `Digest` que a diferencia de la opción `Basic` no transmite los nombres de usuario y contraseña como texto plano (y por lo tanto es una opción de seguridad mejor) pero que no está disponible para todos los navegadores Web como opción "out-of-the-box". El cifrado que usa la opción `Digest` es bastante débil por y aunque su uso no es idéntico al de la opción `Basic` es bastante similar, por lo que no lo veremos.
- `AuthUserFile` es el archivo que se utilizará para guardar las contraseñas.
- `Require` especifica que será necesario acceder con un usuario válido.

Grupo de usuarios

Requisitos previos: tener habilitado el módulo `mod_authz_groupfile`

Si queremos autorizar a un grupo de usuarios a utilizar un recurso de nuestro sitio, tenemos que crear un archivo donde definamos el grupo de usuarios. El archivo lo podemos poner en el mismo sitio que el archivo de contraseñas `/var/passwd/groupousers`. En el archivo pondremos:

```
GroupPrueba: fran eva
```

Creamos una un usuario nuevo con su contraseña:

```
sudo htpasswd /var/passwd/passwords eva
```

Modificamos la sección `<Directory>`:

```
<Directory "/var/www/html/prueba">
    AuthType Basic
    AuthName "Con invitación solamente"
    # Optional line:
    AuthBasicProvider file
    AuthUserFile "/var/passwd/passwords"
    AuthGroupFile "/var/passwd/groupousers"
    Require group GroupPrueba
</Directory>
```

Ahora solo entrarán al directorio prueba los usuarios que se encuentran en el grupo GroupPrueba defnido en el archivo grupousers con la contraseña que tienen en el fichero passwords.

Hay otra manera menos específica de autorizar a varios usuarios. En lugar de crear un archivo de grupo, utilizamos la siguiente directiva:

Require valid-user

Permitirá a cualquier usuario que aparece en el archivo de contraseñas, y que entra correctamente con su contraseña.

Debes conocer

Directiva [Require](#)

Para saber más

[Autenticación en Apache](#)

5.2.2. Archivos .htaccess

La solución vista hasta ahora no es muy adecuada si queremos poder delegar la creación y control de zonas privadas para miembros determinados. Esto puede ser muy útil si por ejemplo hemos montado una web para una empresa que tiene su propio administrador de sistemas ya que evitará que tengamos que gestionar todo nosotros. También nos facilitará el trabajo si hay muchos cambios en las zonas privadas o miembros que se conecten a ellas.

Para permitir el uso de ficheros .htaccess en nuestro servidor o sitio virtual (la directiva se puede usar en ambos entornos) lo primero que debemos hacer es modificar el archivo de configuración. Vamos a hacerlo en el sitio virtual por defecto. Debemos modificar la directiva

```
AllowOverride None
```

Y cambiarla a:

```
AllowOverride AuthConfig
```

El lugar en el que modificar la directiva depende de lo que necesitemos. Ten en cuenta que las directivas se heredan si no se encuentra otra más específica, por ello, en nuestro archivo de configuración del sitio virtual por defecto debería ir en el directorio /var/www por lo menos ya que si lo ponemos en el raíz pero no en el primero, se mantendría la configuración anterior.

```
<Directory /var/www/>
Options Indexes FollowSymLinks MultiViews
AllowOverride AuthConfig
Require all granted
</Directory>
```

Lo que permite que modifiquemos las directivas de autorización mediante un fichero .htaccess. En muchos sitios indican que hay que permitir la sobre escritura de todas las directivas mediante AllowOverride All pero es evidente que es peor opción. Luego reiniciamos el servidor Apache

```
service apache2 reload
```

Ahora podríamos crear ficheros directorios y configurar su control de acceso en cada uno de ellos.

```
mkdir /var/www/ficheros/
cd /var/www/ficheros/
```

En cada directorio que queremos gestionar así debemos crear un fichero **.htaccess** y darle un contenido similar al siguiente.

```
AuthName "Sección Privada: Prueba de .htaccess"
AuthType Basic
AuthUserFile /var/secreto/.miembros
Require valid-user
```

Ten en cuenta que:

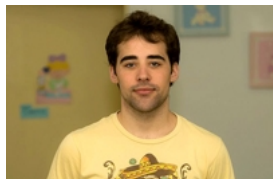
- Lo que hemos hecho ha sido sacar la configuración del control de acceso del archivo de configuración del servidor/sitio virtual a un archivo independiente.
- He usado el mismo archivo de usuarios y contraseñas que en el punto anterior para darle coherencia a los ejemplos, pero esto no es necesario.
- Los usuarios y sus contraseñas se crearían igual que en el apartado anterior.
- La creación o modificación de un fichero de este tipo no implica reiniciar el servidor.
- Se debe mejorar la seguridad cambiando los permisos de acceso al fichero .htaccess. Por lo menos el usuario de Apache debe tener acceso.

Para saber más

- [Trucos y ejemplos de configuración del archivo htaccess de Apache](#)
- [.htaccess ¿Para qué sirve?](#)

6.- Monitorización del acceso: Archivos de registro (logs).

Caso práctico



¿Qué, quién, dónde, cuándo, por qué ha pasado? Eso es lo que queremos saber en todo momento -comentó María-. Recordad que es necesario guardar los archivos de registro al menos durante 1 año según la LSSI/CE. Tenemos que estar preparados ante cualquier petición de los logs (requerimiento judicial) por parte de las administraciones. Es por esto que tú, Antonio, vas a realizar una batería de pruebas: accesos a páginas existentes y no existentes, búsqueda de listado de ficheros y no solamente el index.html, accesos no permitidos a bases de datos, accesos controlados por IP, por usuario, etc.

Muy bien, eso está hecho -dijo Antonio-.

Tan importante como es configurar un servidor web lo es mantener y comprobar su correcto funcionamiento, y para ello debes ayudarte de los logs o archivos de registro que te 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 podamos 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.

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: [día/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 para archivos de registro.

Directivas	Definición
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_empresa1.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_empresa1.log
Directiva ErrorLog	
Sintaxis	ErrorLog nombre_fichero_archivo_registro
Uso	ErrorLog logs/acceso_a_empresa1.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_empresa1.log
Directiva CookieLog	
Sintaxis	CookieLog nombre_fichero_archivo_registro
Uso	CookieLog logs/acceso_a_empresa1.log

En **GNU/Linux** puedes **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 accedes 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 tienes dos opciones para rotar tus registros: **rotatelogs** un programa proporcionado por Apache, o **logrotate**, una utilidad presente en la mayoría de los sistemas GNU/Linux.

No debes 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

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 24 horas 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).

El alias **logformat** es muy interesante, porque permite definir un grupo de modificadores en una palabra, de tal forma que incorporando esa palabra en la directiva log correspondiente estás activando todo un grupo de modificadores. En Apache existen predefinidos en el archivo `/etc/apache2/apache2.conf` los alias **logformat**: **vhost_combined**, **combined**, **common**, **referer** y **agent**, que puedes ver a continuación

Alias predefinidos

Alias logformat predefinidos en /etc/apache2/apache2.conf
LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" vhost_combined
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

Para saber más

Es conveniente que le des una visita al manual de **rotatelogs**: [man rotatelogs](#).

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. Suele emplearse en una tarea diaria del cron.

En **Debian** puedes encontrar los siguientes archivos de configuración para **logrotate**:

- ✓ **/etc/logrotate.conf** : Define los parámetros globales, esto es, los parámetros por defecto de logrotate. Puedes 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**. Puedes encontrar un archivo tipo en el siguiente enlace: [logrotate.d/apache2](#)

Uso de logrotate

Uso de logrotate	
Comprobar la correcta configuración de la rotación de un log	
<code>/usr/sbin/logrotate -d /etc/logrotate.d/apache2</code>	
Forzar la ejecución de logrotate	
<code>/usr/sbin/logrotate -f /etc/logrotate.conf</code>	
/etc/cron.daily/logrotate: Fichero tipo para ejecutar logrotate diariamente en el cron	
<pre>#!/bin/sh test -x /usr/sbin/logrotate exit 0 /usr/sbin/logrotate /etc/logrotate.conf</pre>	
Ejemplo para añadir al archivo crontab del sistema (crontab -e)	
<pre># Rotar logs de apache con logrotate a las 3 am 0 03 * * * root /usr/sbin/logrotate /etc/logrotate.conf > /dev/null 2>&1</pre>	

Para saber más

El **rotado de logs** descrito anteriormente lo podemos aplicar a cualquier otra herramienta del sistema. Es conveniente que le des una visita al manual de **logrotate**: `man logrotate`.

Para saber más

[Programar tareas con crontab](#)

7.- Despliegue de aplicaciones sobre servidores Web.

Caso práctico



La empresa ha quedado muy contenta con el proyecto realizado por BK Programación, con lo cual ha considerado la posibilidad de contratarlos para un nuevo proyecto: la creación de una tienda virtual para la venta del material de la empresa a través de Internet. Para ello mantuvieron una reunión con los siguientes integrantes de BK Programación: Ada, la directora de la empresa y Juan el encargado de desarrollo de aplicaciones web.

-Juan -comentó-, pienso que se podría aprovechar para este proyecto varias aplicaciones de software libre, así el costo se abarataría y la comunidad de programadores es una garantía para la estabilidad del proyecto.

-Entonces -preguntó el representante de la empresa-, el desarrollo del proyecto mediante software libre y no la creación de una tienda virtual propia ¿reduciría el costo y el tiempo de desarrollo del proyecto?

-Sí, -dijo Juan-, existen varias aplicaciones de software libre en el mercado para tiendas virtuales, como: OpenCart, Magento, osCommerce.

-¿Cuál nos recomiendas?

-Pues, hoy en día, OpenCart, pero cualquiera de las tres son una buena elección.

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.

A continuación, puedes ver un ejemplo basado en la aplicación Opencart en el Anexo I.

En este documento se supone que tienes funcionando el siguiente entorno básico: [Apache](#), [MySQL](#) y [PHP](#). Nosotros ya realizamos la instalación pero puedes instalar de diferentes maneras, modo comando o con un gestor de paquetes o con el paquete XAMMP para Linux.

En Ubuntu, instalado Apache, puedes lograrlo con el comando:

```
apt-get install libapache2-mod-auth-mysql mysql-server-5.1 php5-mysql curl php5-curl php5-gd libgd-tools
```

Para saber más

En los siguientes enlaces encontrarás demos de las aplicaciones para tienda virtual: OpenCart, Magento, osCommerce.

[Página demo OpenCart](#)

[Página demo Magento](#)

[Página demo osCommerce](#)

Anexo I.- Despliegue aplicación OpenCart.

Citas para pensar

Diógenes de Sinope: "El movimiento se demuestra andando."

Procede con el siguiente ejemplo: **Instalación de OpenCart**

1. Descarga y descomprime la aplicación:

- ✓ En la página de descarga de OpenCart(<http://www.opencart.com/index.php?route=download/download>) puedes ver los requisitos para la instalación de OpenCart: Web Server (preferably Apache) , PHP (at least 5.2) , MySQL , Curl , Fsock
- ✓ Descarga el último paquete estable de OpenCart de la página web de descarga en `/tmp/pruebas mkdir /tmp/pruebas wget -c`
- ✓ Descomprime el paquete

```
cd /tmp/pruebas apt-get install unzip unzip opencart_v.X.X.zip
```

2. Lee y sigue las instrucciones del fichero de instalación [install.txt](#) (por ejemplo).

3. Crea el virtualhost para OpenCart:

- ✓ Copia la carpeta upload en el servidor web. Para ello genera en `/etc/apache2/sites-available/` un virtualhost de nombre tienda-virtual como el siguiente:

```
<VirtualHost 192.168.200.250:80>
DocumentRoot /var/www/tienda-virtual
ServerName ww.tienda-virtual.empresa-proyecto.com ErrorLog /var/log/apache2/error_tienda-virtual.log
CustomLog var/log/apache2/access_tienda-virtual.log "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" %l %O"
</VirtualHost>
```

- ✓ Ahora mueve la carpeta upload con el nombre tienda-virtual en `/var/www/tienda-virtual`
- ✓ Activa el sitio nuevo tienda-virtual: `a2ensite tienda-virtual`
- ✓ Recarga la configuración de Apache: `/etc/init.d/apache2 reload`
- ✓ Verifica que los siguientes ficheros y carpetas tengan permisos de escritura en `/var/www/tienda-virtual/`: `chmod 0777` para: `image/`, `image/cache/`, `image/data/`, `system/cache/`, `system/logs/`, `download/`, `config.php`, `admin/config.php`

4. Crea la base de datos para OpenCart y el usuario (Que no se el root) con permisos en la misma:

Asegúrate que posees una base de datos mysql para OpenCart y un usuario distinto de root con permisos en la misma:

- ✓ Primero, debes crear una nueva base de datos para tu sitio OpenCart: `/usr/bin/mysql -h127.0.0.1 -uroot -p -e "CREATE DATABASE db_opencart;"` donde:
 - root es el usuario administrador de MySQL y por lo tanto tiene los privilegios para crear una base de datos.
 - db_opencart es el nombre de la base de datos de opencart que acabas de crear.

MySQL te pide la contraseña del usuario root y luego crea los archivos iniciales de la base de datos.

- ✓ Segundo, creas el usuario con privilegios en la base de datos de nuevo se requiere la contraseña de root-.

```
/usr/bin/mysql -h127.0.0.1 -uroot -p -e "GRANT SELECT,UPDATE,INSERT,DELETE,DROP,INDEX,ALTER,CREATE ON "db_opencart" .*
TO "db_user_opencart"@localhost IDENTIFIED BY 'opencart';"
```

donde:

- 'db_opencart' es el nombre de tu base de datos
- 'db_user_opencart@localhost' es el nombre de usuario de MySQL que posee los privilegios en la base de datos 'db_opencart'.
- 'opencart' es la contraseña requerida para iniciar sesión como el usuario 'db_user_opencart' en MySQL
- ✓ Tercero, para activar los nuevos cambios ejecuta: `/usr/bin/mysql -h127.0.0.1 -uroot -p -e "flush privileges;"`

Alternativamente puedes usar, si lo posees, tu panel de control Web o bien phpMyAdmin para crear la base de datos 'db_opencart' y el usuario 'db_user_opencart'

5. Visita la página principal de tu OpenCart, por ejemplo: <http://www.tienda-virtual.empresa-proyecto.com/>

6. Sigue las instrucciones que aparecen en pantalla.

7. Una vez acabada la instalación borra la carpeta install.














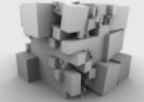









8. Puedes ya visitar tu tienda online en: <http://www.tienda-virtual.empresa-proyecto.com/> y tu panel de administración en: <http://www.tienda-virtual.empresa-proyecto.com/admin/>

Orientaciones para el alumnado

Posibles problemas al instalar el [módulo mCrypt de PHP](#) en Ubuntu 14.04

Anexo.- Licencias de contenidos.

Licencias de recursos utilizados en la Unidad de Trabajo.

Recurso (1)	Datos del recurso (1)	Recurso (2)	Datos del recurso (2)
	Autoría: msarturlr Licencia: CC BY-NC-SA 2.0 Procedencia: http://www.flickr.com/photos/31899509@N02/3782931291/sizes/l/in/photostream/		Autoría: mayhem Licencia: CC BY-NC-SA 2.0 Procedencia: http://www.flickr.com/photos/mayhem/
	Autoría: Apache Software Foundation. Licencia: Apache License Version 2.0 Procedencia: http://www.apache.org/licenses/LICENSE-2.0		Autoría: Debian (http://www.debian.org) Licencia: Copyright (c) 1999 Scott O. Rothstein Procedencia: http://www.debian.org
	Autoría: David Davies Licencia: CC BY-SA 2.0 Procedencia: http://www.flickr.com/photos/davies/3231308527/sizes/z/in/photostream/		Autoría: Svein M. Licencia: CC BY-NC-SA 2.0 Procedencia: http://www.flickr.com/photos/sveinm/
	Autoría: Steve Rhode Licencia: CC BY-NC-ND 2.0 Procedencia: http://www.flickr.com/photos/steverhode/3183290111/in/photostream		Autoría: Scott MacLeod Liddle Licencia: CC BY-NC-ND 2.0 Procedencia: http://www.flickr.com/photos/scottmacleodliddle/
	Autoría: ivanpw Licencia: CC BY 2.0 Procedencia: http://www.flickr.com/photos/28288673@N07/4848301878/sizes/m/in/photostream/		Autoría: Jason Whittaker Licencia: CC BY-NC-SA 2.0 Procedencia: http://www.flickr.com/photos/jasonwhittaker/
	Autoría: bcostin Licencia: CC BY-NC-SA 2.0 Procedencia: http://www.flickr.com/photos/bcostin/2556940166/sizes/t/in/photostream/		Autoría: Apache Friends Licencia: GPL2 Procedencia: http://commons.wikimedia.org/wiki/File:XAMPP.png
	Autoría: AJC1 Licencia: CC BY-NC-SA 2.0 Procedencia: http://www.flickr.com/photos/ajc1/4663140532/sizes/m/in/photostream/		Autoría: syntopia Licencia: CC BY 2.0 Procedencia: http://www.flickr.com/photos/syntopia/
	Autoría: PolandMFA Licencia: CC BY-ND 2.0 Procedencia: http://www.flickr.com/photos/polandmfa/3986390339/sizes/sq/in/photostream/		Autoría: Ministerio de Educación Licencia: Apache License, Versión 2.0 Procedencia: Elaboración propia
	Autoría: DaveBleasdale Licencia: CC BY 2.0 Procedencia: http://www.flickr.com/photos/sidelong/3878741556/sizes/z/in/photostream/		Autoría: Mozilla Licencia: Creative Commons Attribution-ShareAlike License Procedencia: Captura de pantalla
	Autoría: Ministerio de Educación (Ricardo Feijoo Costa) Licencia: Apache License, Version 2.0 Procedencia: Elaboración propia sobre captura de pantalla de Apache.		Autoría: Ralf S. Engelschall Licencia: BSD Procedencia: http://www.modssl.org/
	Autoría: Mozilla Licencia: Creative Commons Attribution Share-Alike License v3.0 Procedencia: Captura de pantalla del navegador Firefox.		Autoría: Desconocida Licencia: Copyright 2011 © OpenLDAP Procedencia: http://www.openldap.org/
Página profesor Sergio Cuesta	Autoría: Sergio Cuesta Licencia: Reconocimiento-NoComercial-CompartirIgual (CC BY-NC-SA 3.0 ES) 3.0 España		Autoría: DCG Licencia: Creative Commons Attribution-ShareAlike License Procedencia: obra derivada de