

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:

- ✓ Sistema Operativo: Debian GNU/Linux 6.0
- ✓ 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**.

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? ...

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**, **dominios de Internet** o solamente uno, esto es, permite **hosts 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 visione 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

```
/etc/apache2/  
├─ apache2.conf  
├─ conf.d  
├─ envvars  
├─ httpd.conf  
├─ magic  
├─ mods-available  
├─ mods-enabled  
├─ ports.conf  
├─ sites-available  
└─ sites-enabled
```

```
/etc/apache2/sites-available/  
├─ default  
└─ default-ssl
```

```
/var/www/  
└─ index.html
```

```
/etc/apache2/mods-available/mime.conf
```

```
/etc/apache2/apache2.conf
```

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.

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 **Favoritos** la [dirección URL](#) 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 más [módulos](#) tengamos que soportar, más tendremos que configurar y actualizar. Esto también tendrá una gran repercusión en la seguridad del servidor web: cuántos más módulos más posibilidades de problemas de seguridad, así si la página web dinámica necesita, para ser ofrecida, de ejecución en el servidor 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.



1.3.- 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 [DNI](#) 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 sí, 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 puedes encontrar un vídeo muy ameno sobre el funcionamiento de Internet.

Resumen textual alternativo

1.4.- 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.

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](#)

1.4.1.- 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.](#)



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.

Mostrar Información

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 heredará 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.

Si no tienes configurado un servidor DNS con las entradas de dominio necesarias, puedes generar estas entradas modificando el archivo **/etc/hosts**, añadiéndolas al final del mismo:

```
#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.

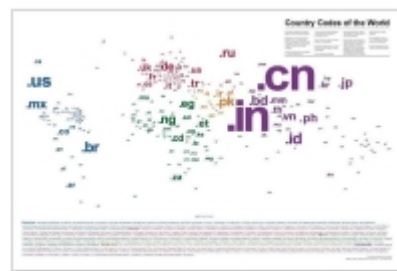
Estas entradas solamente serán efectivas en el equipo en el que se modifique el archivo **/etc/hosts**. Así debes modificar el archivo **/etc/hosts** en cada equipo que quieres que se resuelvan esas entradas.

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

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

- ### 3. Fichero configuración virtualhost: empresa2

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

Explicación fichero virtualhost:

<VirtualHost IP_Servidor_Web:80> : Inicio etiqueta virtualhost, define la IP del servidor web donde se aloja la página de la empresa, en este caso empresa1. 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.

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](#)

2.3.- Virtualhosts basados en varios servidores principales

Citas para pensar

Richard Stallman: "Cuando me preguntan cuándo estará listo un programa, contesto: depende de cuánto trabaje usted en ello."

Este método es el más complejo de todos, solo tiene sentido cuando quieras tener varios archivos de configuración **apache2.conf** independientes organizando cada uno sus propios hosts virtuales, en otro caso, mejor emplear alguno de los dos métodos anteriores.

Para saber más



XAMPP es una forma fácil de instalar y usar el servidor web Apache con MySQL, PHP y Perl. XAMPP, basta con descargarlo, extraerlo y comenzar. En este momento hay cuatro versiones de XAMPP, para: Linux, Windows, Mac OS X y Solaris.

[XAMPP](#)

Te proponemos los siguientes enlaces con interesantísimos vídeos sobre la instalación y uso de XAMPP en Windows.

[Vídeo sobre XAMPP en Windows](#)

[Resumen textual alternativo](#)

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/`.

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/`

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
<code>apt-get install nombre-modulo</code>	<code>apt-get install libapache2-mod-gnutls</code>
Desinstalar un módulo	Ejemplo: Desinstalar el módulo ssl
<code>apt-get remove nombre-modulo</code>	<code>apt-get remove libapache2-mod-gnutls</code>
Habilitar un módulo	Ejemplo: Habilitar el módulo ssl
<code>a2enmod nombre-modulo-apache</code>	<code>a2enmod ssl</code>
Deshabilitar un módulo	Ejemplo: Deshabilitar el módulo ssl
<code>a2dismod nombre-modulo-apache</code>	<code>a2dismod ssl</code>

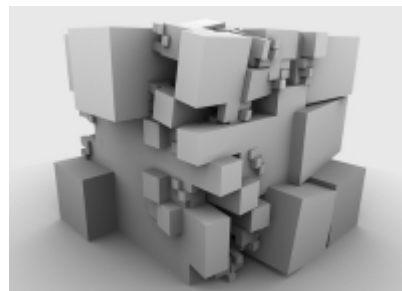
Para habilitar un módulo Apache, en Debian, también puedes ejecutar el comando **a2enmod** sin parámetros. La ejecución de este comando ofrecerá una lista de módulos a habilitar, escribes el módulo en cuestión y el módulo se habilitará. Del mismo modo para deshabilitar un módulo Apache, en Debian, puedes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

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.

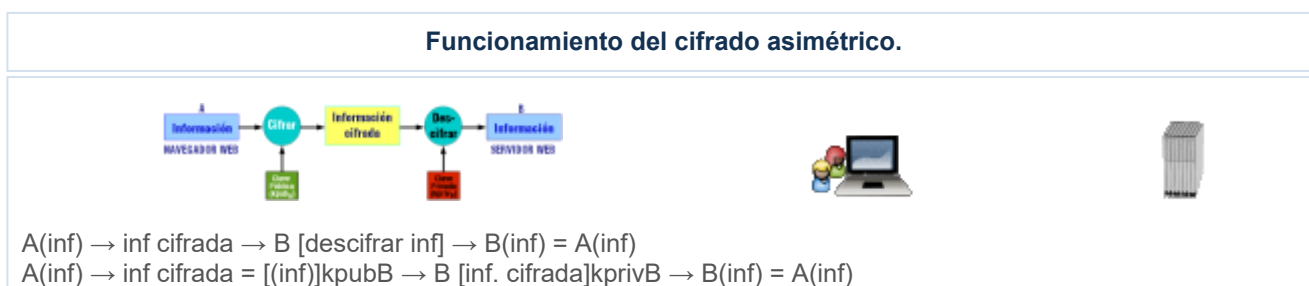
El cifrado al que nos referimos es el cifrado de clave pública o asimétrico: **clave pública (kpub)** y **clave privada (kpriv)**. La **kpub** interesa publicarla para que llegue a ser conocida por cualquiera, la **kpriv** no interesa que nadie la posea, solo el propietario de la misma. Ambas son necesarias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



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

Como 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 un 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. Éstas certifican el documento de asociación entre clave pública e identidad de un individuo o máquina (servidor web) firmando dicho documento con su clave privada, esto es, mediante firma digital.

La idea consiste en que los **dos extremos de una comunicación, por ejemplo cliente (navegador web) y servidor (servidor web Apache)** puedan confiar directamente entre sí, si ambos tienen relación con una tercera parte, que da fe de la fiabilidad de los dos, aunque en la práctica 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, periodo 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

¿Como harías, en Debian 6, para deshabilitar el modulo ssl de Apache?
Y ¿cómo lo harías si no dispones del comando para Debian?

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

En Debian, Apache posee por defecto en su instalación el fichero `/etc/apache2/sites-available/default-ssl` (0.03 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 sí mismo -ya que Apache actúa como entidad certificadora-.

Cuando activaste el módulo ssl, mediante el comando `a2enmod ssl` permitiste que Apache atienda el protocolo SSL. 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:

192.168.200.250 usa un certificado de seguridad no válido.

No se confía en el certificado porque está autofirmado.

El certificado sólo es válido para debian-servidor-fp.

(Código de error: `sec_error_untrusted_issuer`)

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.

4.3.1.- Crear un servidor virtual seguro en Apache (II).

En una distribución Debian 6 el procedimiento para generar un certificado digital sería el siguiente:

1. Instalación del paquete openssl

```
apt-get install openssl
```

2. Crear un certificado autofirmado para el servidor web.

```
mkdir /etc/apache2/tus-ssl/  
make-ssl-cert /usr/share/ssl-cert/ssleay.cnf /etc/apache2/tus-ssl/apache.pem
```

Cuando se solicite el nombre del servidor HTTP indicar el nombre DNS que corresponda a la IP del certificado, por ejemplo: **autofirmado.ssl.empresa-proyecto.com**

El nombre de dominio **autofirmado.ssl.empresa-proyecto.com** debe resolverse a una IP mediante un servidor DNS o en su defecto mediante el fichero **/etc/hosts**.

El fichero generado **/etc/apache2/tus-ssl/apache.pem** contiene tanto el certificado del servidor como la clave privada asociada al mismo.

El comando, de Debian, **make-ssl-cert** permite generar certificados autofirmados para pruebas. Los datos de configuración del certificado a generar se indican en **/usr/share/ssl-cert/ssleay.cnf**. Internamente hace uso de las utilidades de la librería **openssl**.

3. Editar la configuración SSL por defecto en el archivo **/etc/apache2/sites-available/default-ssl** para indicar el certificado del servidor y su respectiva clave privada asignando los siguientes valores a los parámetros :

```
SSLEngine on  
SSLCertificateFile /etc/apache2/tus-ssl/apache.pem  
SSLCertificateKeyFile /etc/apache2/tus-ssl/apache.pem
```

4. Asegúrate que el fichero **/etc/apache2/ports.conf** incluya el valor **Listen 443**

5. Habilita el soporte SSL en Apache y habilita la configuración SSL por defecto:

```
a2enmod ssl  
a2ensite default-ssl  
/etc/init.d/apache2 restart
```

En el equipo cliente, **192.168.200.100**, lanzas el navegador:

1. Indicar **https://autofirmado.ssl.empresa-proyecto.com** en la barra de direcciones.
2. Dará un aviso de que la AC que firma el certificado del servidor no está reconocida. Añadir la correspondiente excepción de seguridad y permitir la descarga y aceptación del certificado. Antes de aceptarlo puedes ver el contenido del certificado:



4.3.2.- Crear un servidor virtual seguro en Apache (III).

De forma genérica, por si no posees el comando **make-ssl-cert**, puedes emplear el comando **openssl** para generar los certificados. Por ejemplo:



1. Instalación del paquete openssl:

```
apt-get install openssl
```

2. Genera el certificado y la clave privada de tu autoridad de certificación (AC)

```
mkdir /etc/apache2/tus-ssl/  
cd /etc/apache2/tus-ssl/
```

Puedes ver la ejecución de los dos comandos que se utilizan a continuación en el archivo [openssl_autofirmado.txt](#) (0.83 KB)

```
openssl req -new -nodes -keyout tupaginaweb.key -out tupaginaweb.csr
```

Este comando genera dos archivos:

- ✔ La clave privada con el que firmarás tus futuros certificados: `tupaginaweb.key`
- ✔ El certificado con la clave pública de la AC: `tupaginaweb.csr`

Este comando pedirá algunos datos: nombre de empresa, país, contraseña... La contraseña, puedes omitirla, pero por seguridad es conveniente crearla para utilizarla cuando firmes un certificado SSL.

3. Autofirma el certificado. Puedes hacerlo porque eres una AC, de tal forma que el primer certificado que firmas es el de tu propia AC.

```
openssl x509 -in tupaginaweb.csr -out tupaginaweb.crt -req -signkey tupaginaweb.key -days 3650
```

El campo **days 3650** significa que el certificado de tu AC tardará 10 años en caducar.

4. Editar la configuración SSL por defecto en el archivo **/etc/apache2/sites-available/default-ssl** para indicar el certificado del servidor y su respectiva clave privada asignando los siguientes valores a los parámetros :

```
SSLEngine on  
SSLCertificateFile /etc/apache2/tus-ssl/tupaginaweb.crt  
SSLCertificateKeyFile /etc/apache2/tus-ssl/tupaginaweb.key
```

5. Asegúrate que el fichero **/etc/apache2/ports.conf** incluya el valor **Listen 443**

6. Habilita el módulo ssl y la configuración SSL por defecto.

```
a2enmod ssl  
a2ensite default-ssl  
/etc/init.d/apache2 restart
```

Reflexiona

En el archivo `/usr/share/doc/apache2.2-common/README.Debian.gz` encontrarás información sobre como configurar SSL y crear certificados autofirmados.

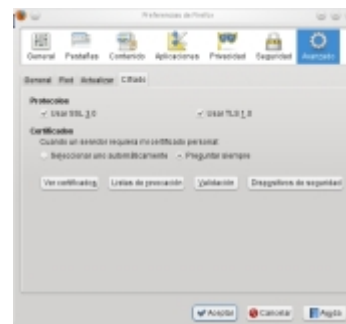
4.4.- Comprobar el acceso seguro al servidor.

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 una 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 > 3.x , donde x es el número de revisión de la versión 3, puedes verlas dirigiéndote por las pestañas a:

Editar → Preferencias → Avanzado → Cifrado → Ver certificados

- ✓ Revisar la lista de revocaciones que posee tu navegador. En **Firefox**, versión > 3.x , donde x es el número de revisión de la versión 3, puedes verlas dirigiéndote por las pestañas a:
- ✓ **Editar → Preferencias → Avanzado → Cifrado → Listas de revocación**

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.

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.

Para saber más

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

[AAA](#)

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](#)

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

5.1.- Autenticar usuarios en apache mediante LDAP.

Se ha comentado en el apartado anterior que el servidor web Apache permite la autenticación de usuarios mediante LDAP. Esto es posible mediante los módulos **ldap** y **authnz_ldap**.

Debes conocer

Es conveniente que visites el siguiente enlace a un archivo que contiene como instalar y configurar un servidor OpenLDAP en un GNU/Linux basado en Debian, con el que Apache puede realizar la autenticación.

[Instalación y configuración del servidor OpenLDAP](#)

Para una instalación de OpenLDAP en Debian 6 visita el enlace al siguiente documento:

[Instalación y configuración del servidor OpenLDAP en Debian 6](#)

Para saber más

En el siguiente enlace encontrarás más información sobre la autenticación LDAP para el servidor web Apache.

[Autenticación LDAP en Apache mediante el módulo **authnz_ldap**.](#)

Para el buen funcionamiento de lo expuesto a continuación se asume que tanto Apache2 como OpenLDAP están instalados y configurados:

1. Habilita el soporte LDAP para Apache2:

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

2. Configura el virtualhost **autenticacion-ldap-apache** como sigue:

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

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

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

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

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

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



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

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: **rotatelog** 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 rotatelog

Uso de rotatelog
CustomLog " ruta_rotatelog ruta_log_a_rotar numero_segundos tamaño_máximoMB" alias_logformat
Ejemplos
Rotar el archivo de registro access.log cada 24horas CustomLog " /usr/sbin/rotatelog /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/rotatelog /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/rotatelog /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 tras pasados 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 **rotatelog**: `man rotatelog`.

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	
<code>/usr/sbin/logrotate -d /etc/logrotate.d/apache2</code>	
Forzar la ejecución de logrotate	
<code>/usr/sbin/logrotate -f /etc/logrotate.conf</code>	
<code>/etc/cron.daily/logrotate</code> : 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 (c	
<pre># Rotar logs de apache con logrotate a las 3 am 0 03 * * * root /usr/sbin/logrotate /etc/logrotate.conf > /dev/nu</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**.

Reflexiona

En el siguiente enlace podrás comprobar tus conocimientos adquiridos hasta el momento:

Resumen textual alternativo



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, en el siguiente documento puedes ver un [ejemplo basado en la aplicación Opencart](#).

En este documento se supone que tienes funcionando el siguiente entorno básico: [Apache](#), [MySQL](#) y [PHP](#). En Debian 6 ,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
```

Otra buena opción sería instalar el paquete [XAMMP para GNU/Linux](#):

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 Sínope: "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 http://opencart.googlecode.com/files/opencart_v1.4.9.5.zip (6 MB)
- ✓ Descomprime el paquete

```
cd /tmp/pruebas apt-get install unzip unzip opencart_v1.4.9.5.zip
```

2. Lee el fichero de instalación install.txt.

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
CustomLog var/log/apache2/access_tienda-virtual.log "%h %l %u %t \"%r\" %>s %b \"
</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 0755 ó 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 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,]
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;"`












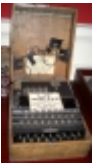
Alternativamente puedes usar, si lo posees, tu panel de control Web o bien phpMyAdmin para crear la base de datos 'db_opencart' y el usuario 'db_user_opencart'

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/>



Anexo.- Licencias de contenidos.

Licencias de recursos utilizados

Recurso (1)	Datos del recurso (1)	
	<p>Autoría: msarturlr Licencia: CC BY-NC-SA 2.0 Procedencia: http://www.flickr.com/photos/31899509@N02/3782931291/sizes/l/in/photostream/</p>	
	<p>Autoría: Apache Software Foundation. Licencia: Apache License Version 2.0 Procedencia: http://www.apache.org/licenses/LICENSE-2.0</p>	
	<p>Autoría: David Davies Licencia: CC BY-SA 2.0 Procedencia: http://www.flickr.com/photos/davies/3231308527/sizes/z/in/photostream/</p>	
	<p>Autoría: Steve Rhode Licencia: CC BY-NC-ND 2.0 Procedencia: http://www.flickr.com/photos/steverhode/3183290111/in/photostream</p>	
	<p>Autoría: ivanpw Licencia: CC BY 2.0 Procedencia: http://www.flickr.com/photos/28288673@N07/4848301878/sizes/m/in/photostream/</p>	
	<p>Autoría: bcostin Licencia: CC BY-NC-SA 2.0 Procedencia: http://www.flickr.com/photos/bcostin/2556940166/sizes/t/in/photostream/</p>	
	<p>Autoría: AJC1 Licencia: CC BY-NC-SA 2.0 Procedencia: http://www.flickr.com/photos/ajc1/4663140532/sizes/m/in/photostream/</p>	
	<p>Autoría: PolandMFA Licencia: CC BY-ND 2.0 Procedencia: http://www.flickr.com/photos/polandmfa/3986390339/sizes/sq/in/photostream/</p>	
	<p>Autoría: DaveBleasdale Licencia: CC BY 2.0</p>	



Procedencia:
<http://www.flickr.com/photos/sidelong/3878741556/sizes/z/in/photostream/>



Autoría: Ministerio de Educación (Ricardo Feijoo Costa)
Licencia: Apache License, Version 2.0
Procedencia: Elaboración propia sobre captura de pantalla de Apache.



Autoría: Mozilla
Licencia: Creative Commons Attribution Share-Alike License v3.0
Procedencia: Captura de pantalla del navegador Firefox.