Configuración y administración de servidores Web.

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: Ubuntu 22.04 LTS
- Servidor Web: Apache (apache2)
  - Configuración de Red:
  - Servidor Web: 192.168.56.56
  - Cliente de pruebas (desde donde se lanza el navegador): 192.168.56.1

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 **C:\Windows\System32\drivers\etc\hosts**, si necesitamos que la resolución de nombres se realice en el servidor linux debemos editar el fichero **/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? ...

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.

Apache Software Foundation (Apache 2.0)

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 cuánto 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 cuánto 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 servidor Ubuntu 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 Ubuntu

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

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

/var/www/html
└── 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/html/index.html`** referenciada a través del archivo **`/etc/apache2/sites-avaliable/000-default.conf`**, 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 el contenido del directorio **`/var/www/html`** por las páginas que quieres servir incluidas en el siguiente archivo zip **`web-estática.zip`**. Puedes comprobarlo siguiendo el procedimiento:

Desde tu máquina WIndows, copia el archivo web-estatica.zip al servidor ubuntuserver en el directorio raíz del usuario ubuntu:

```
scp <ruta a web-estatica.zip> ubuntu@<dirIP-servidor>:/home/ubuntu
```

Ya en el servidor,.salvamos el fichero original /var/www/html/index.html en el directorio raíz del usuario ubuntu para no machacarlo con los ficheros nuevos y descomprimimos y copiamos el contenido del directorio web-estatica al directorio /var/www/html

```
sudo mv index.html .
unzip web-estatica.zip
sudo cp -r web-estatica/* /var/www/html
```

Comprueba que el despliegue ha funcionado accediendo desde un navegador a la URL, **`http://www.dawdistancia.net`**

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 2.4)

1.2.- Contenido dinámico.

Citas para pensar

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

David Davies (CC BY-SA)

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_php, mod_python.`**

Vamos a sustituir el sitio web estático de la sección anterior por otro sitio creado con un script PHP que permite jugar a adivinar un número secreto. Puedes descargar el sitio desde este enlace, web-dinamica.zip.

Para desplegar esta aplicación seguimos los sigguientes pasos:

Desde tu máquina Windows, copia el archivo web-dinamica.zip al servidor ubuntuserver en el directorio raíz del usuario ubuntu:

```
scp <ruta a web-dinamica.zip> ubuntu@<dirIP-servidor>:/home/ubuntu
```

Ya en el servidor, descomprimios el archivo zip y copiamos el contenido del directorio web-estatica al directorio /var/www/html.

```
sudo rm -R /var/www/html/*
sudo cp -R web-dinamica/* /var/www/html
```

Comprueba que el despliegue ha funcionado accediendo desde un navegador a la URL **`http://www.dawdistancia.net/index.php`**

Debes conocer

Composer es una herramienta de administración de dependencias para PHP que facilita la gestión de bibliotecas y paquetes en tus proyectos utilizado por los desarrolladores PHP. Algunas de sus funcionalidades clave incluyen:

- Instalación de dependencias: Composer te permite definir las bibliotecas y paquetes que tu proyecto necesita y los instala automáticamente a partir de registros de paquetes en línea.
- Autocarga de clases: Composer genera un archivo de autocarga que simplifica la inclusión de clases en tu código, eliminando la necesidad de incluir manualmente archivos de clase.
- Resolución de conflictos: Resuelve automáticamente conflictos entre diferentes versiones de dependencias para garantizar la compatibilidad de las bibliotecas en tu proyecto.
- Actualización de dependencias: Puedes actualizar las dependencias de tu proyecto fácilmente con Composer, lo que te permite mantener tu código actualizado con las últimas versiones de las bibliotecas.
- Creación de paquetes personalizados: Composer te permite crear y distribuir tus propias bibliotecas y paquetes PHP de manera sencilla.

Para instalarlo en un servidor ubuntu 22.04 hay que seguir los pasos descritos en el siguiente enlace.

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 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.

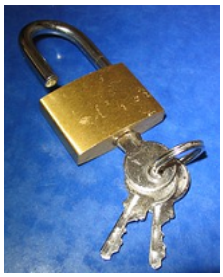
No es correcta. Si la página web muestra acciones distintas según el navegador que solicita la página, la página es dinámica.

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

Solución

1. Incorrecto
2. Incorrecto
3. Incorrecto
4. Opción correcta

1.3.- Protocolo HTTP y HTTPS.



Miala (CC BY-SA)

El Protocolo de Transferencia de Hipertexto (HTTP) es un protocolo de comunicación utilizado en la World Wide Web para la transferencia de datos entre un cliente (como un navegador web) y un servidor web. Los mensajes del protocolo HTTP constan de dos tipos principales:

`Solicitud (Request)`: Este mensaje es enviado por el cliente al servidor para solicitar un recurso web, como una página HTML, una imagen o un archivo. Una solicitud HTTP generalmente incluye métodos como GET (para recuperar datos), POST (para enviar datos al servidor), PUT (para actualizar un recurso), DELETE (para eliminar un recurso), y otros. Además, contiene la URL del recurso solicitado y una serie de cabeceras que proporcionan información adicional.
`Respuesta (Response)`: El servidor responde a una solicitud HTTP con un mensaje de respuesta que incluye un código de estado (como 200 OK para éxito o 404 Not Found para recurso no encontrado), el tipo de contenido, y el contenido real solicitado (por ejemplo, una página web, una imagen, un archivo de datos, etc.). También puede incluir cabeceras adicionales para proporcionar información adicional sobre la respuesta.

Estos mensajes HTTP son la base de la comunicación entre los navegadores web y los servidores web, permitiendo la transferencia de datos y recursos a través de la World Wide Web.

¿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 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 si 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 HTPS 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 si, 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.

Steve Rhode (CC BY-NC-ND)

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.

Scott MacLeod Liddle (CC BY-NC-ND)

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. Esta directiva esta deshabilitada a partir de la versión 2.3.
- **`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 reprodución del vídeo.

Mostrar retroalimentación

Solución

1. Correcto
2. Incorrecto
3. Correcto
4. Incorrecto

2.- Servidores virtuales (VirtualHosts). 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 `www.dawdistancia.net`, entonces, ¿no se puede alojar páginas de distintos dominios en el mismo servidor web? La respuesta es que si, si se puede, ¿cómo?, mediante la configuración de servidores virtuales o virtualhosts. Éstos básicamente lo que hacen es permitir que un mismo servidor web pueda alojar múltiples dominios, así configurando servidores virtuales podemos alojar: **empresa1.com** , **empresa2.com**, ..., **empresaN.com** en el mismo servidor web. Cada empresa tendrá su servidor virtual único e independiente de las demás.

Aunque como se ha comentado anteriormente cada servidor virtual es único e independiente de los demás, todo aquello que no esté incluido en la definición de cada servidor virtual se heredará de la configuración principal: **apache2.conf (`/etc/apache2/apache2.conf`)**, así. si quieres definir una directiva común en todos los servidores virtuales no debes modificar cada uno de los servidores introduciendo esa directiva sino que debes definir esa directiva en la configuración principal del servidor web Apache, de tal forma que todos los servidores virtuales 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 dos tipos de servidores virtuales: basados en el nombre de dominio del servidor y basados en dirección IP. La diferencia radica en el dato que usa el servidor Apache para enviar las peticiones a uno u otro servidor virtual que conviven en la misma instalación.

El proceso de creación de un servidor virtual consiste en crear un fichero de configuración en el directorio `/etc/apache2/sites-available`. Además, para activar o desactivar los servidores virtuales se pueden usar los comandos:

- **a2ensite**: Utilizado para activar o habilitar un servidor virtual. Se le indica como parámetro el nombre del archivo de configuración (sin la extensión .conf) que hemos creado previamente. Si no se le pasa el parámetro preguntará que sitio se desea habilitar. Los ficheros de configuración de los sitios disponibles están en `/etc/apache2/sites-available/` y al habilitarlos se crea un enlace simbólico desde `/etc/apache2/sites-enabled/` .
- **a2dissite**: Utilizado para deshabilitar un servidor virtual. Se le indica como parámetro el nombre del archivo de configuración del servidor virtual (sin la extensión .conf). Sin ningún parámetro preguntará que sitio se desea deshabilitar. Los ficheros de configuración de los módulos disponibles están en `/etc/apache2/sites-available/` y al deshabilitarlos se elimina el enlace simbólico desde `/etc/apache2/sites-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/.

Debes conocer

Para realizar resolución de nombres de dominio a direcciones IP de forma local, es decir, en las misma máquina que usa el nombre de dominio en la petición, se deben definir las correspondencias de nombre de dominio e IP en un fichero llamado hosts.

Si se trata de un PC con SO Linux puedes generar estas entradas modificando el archivo **`/etc/hosts`**, añadiéndolas al final del mismo:

```
# IP nombre-dominio
192.168.56.56 empresa1.com www.empresa1.com
```

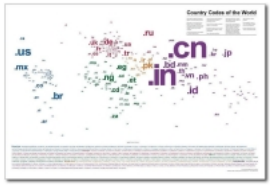
Si se trata de un PC con SO Windows tendrás que modificar el archivo **`C:\Windows\System32\drivers\etc\|hosts`**. Para editar este fichero necesitas abrir un bloc de notas con permiso de Administrador.

Estas entradas solamente serán efectivas en el equipo en el que se modifique el archivo **`hosts`**.

2.1.- Servidores virtuales basados en nombre.

Vamos a crear un servidor virtual basado en nombre de dominio para desplegar el sitio web estático que desplegamos en la sección anterior pero esta vez bajo el nombre de dominio **gatitos.net**. Para ello vamos a seguir los pasos siguientes:

1. En la configuración de Apache2 existe un directorio **/etc/apache2/sites-available** donde se definen los servidores virtuales, cada servidor virtual dispone de un fichero de configuración distinto. De modo que creamos el fichero **gatitos.net.conf** en la ruta **/etc/apache2/sites-available**.
2. Añadimos la configuración al fichero del servidor virtual **gatitos.net.conf**




```
<VirtualHost *:80>
        DocumentRoot "/var/www/gatitos"
        ServerName gatitos.net
        ServerAlias www.gatitos.net
</VirtualHost>
```

A continuación, se explican las directivas utilizadas en la configuración del servidor virtual **gatitos.net.conf**:

- **<VirtualHost IP_Servidor_Web:puerto>** : Inicio etiqueta VirtualHost, define la IP del servidor web donde se aloja el sitio web. En nuestro caso el carácter * indica que este servidor virtual estará escuchando en todos los interfaces de red que se hayan definido en el servidor. El puerto para el protocolo HTTP por defecto es el 80, definido en la configuración principal del servidor, mediante la directiva **Listen**. 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 se puede configurar:

```
Listen 192.168.56.56:80
Listen 192.168.56.57:8080
```

- **DocumentRoot /var/www/gatitos,** Definición de la ruta donde está alojada la página web en el servidor, en este caso: /var/www/gatitos/ mediante la directiva **DocumentRoot.**
- **ServerName gatitos.net,** Indica el nombre del dominio que aparece en el URL de acceso al sitio web desde el navegador. Se utiliza para redireccionar las peticiones que llegan al servidor Apache al servidor virtual correcto, de tal manera que este servidor virtual solo recibirá peticiones dirigidas a este dominio.
- **ServerAlias www.gatitos.net** : permite asociar el servidor virtual a otros nombres de dominio.
- **</VirtualHost>** : Fin de la etiqueta **VirtualHost**: fin de la definición de este virtualhost para la gatitos.net.

Ahora puedo copiar de nuevo los archivos del sitio web que descargamos en la sección anterior (web-estatica.zip) en la ruta **/var/www/gatitos** del servidor web.

```
sudo mkdir /var/www/gatitos
sudo cp -R web-estatica/* /var/www/gatitos
```

No te olvides habilitar el nuevo sitio web para que el servidor Apache pueda servir las peticiones que le llegan.

```
sudo a2ensite gatitos.net
```

Para comprobar que el sitio web se ha desplegado correctamente vamos a acceder a ella desde nuestra máquina Windows. Para ello, debemos añadir una nueva entrada al fichero de resolución de nombres de WIndows (C:\Windows\System32\drivers\etc\hosts) con la correspondencia entre el nombre del servidor www.gatitos.net y la dirección IP donde reside.

Ahora ya podemos abrir un navegador y acceder a la URL: http://www.gatitos.net para acceder a la página principal del sitio.

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
2. Opción correcta
3. Incorrecto
4. Incorrecto

2.2.- Servidores virtuales basados en IP.

bcostin (CC BY-NC-SA)

Ahora vamos a crear un nuevo servidor virtual basado en la IP a la que está asociado en la configuración. La directiva **`VirtualHost`** va a especificar la IP de uno de los interfaces con los que está configurado el servidor, en nuestro caso la dirección que conecta el servidor ubuntuserver a la red NAT es 10.0.2.2.

El sitio web que vamos a desplegar ofrece la funcionalidad del juego del buscaminas programado en PHP y JavaScript y se puede descargar desde este enlace.

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.

Para realizar el despliegue del sitio web seguimos un procedimiento similar al utilizado para el servidor virtual basado en nombre:

1. En la configuración de Apache2 existe un directorio **`/etc/apache2/sites-available`** donde se definen los servidores virtuales, cada servidor virtual en un fichero de texto de configuración distinto.
2. Creamos el fichero de configuración del servidor virtual **`buscaminas.net.conf`** en la carpeta **`/etc/apache2/sites-available.`**

```
<VirtualHost 10.0.2.2:80>
        DocumentRoot "/var/www/buscaminas"
        ServerName www.buscaminas.net
</VirtualHost>
```

En este caso, conviene fijarse en la direcciíon IP que aparece en la definición de la etiqueta **`<VirtualHost 10.0.2.2:80>.`** Dicha configuración permite al servidor Apache redirigir todas las peticiones que llegan hacia esa dirección IP a este servidor virtual.

El resto de las directivas del fichero de configuración ya han sido explicadas en la sección anterior.

A continuación, copiamos el archivo descargado **`buscaminas.zip`** al directorio raíz del usuario ubuntu en el servidr ubuntuserver desde nuestro PC en Windows:

```
scp <ruta a buscaminas.zip> ubuntu@<dirIP-servidor>:/home/ubuntu
```

Ahora, abrimos una sesión con el servidor y realizo las operaciones de descomprimir el archivo, instalar las librerías y paquetes necesarios para ejecutar la aplicación, ubicar los archivos en el directorio indicado por la directiva **`DocumentRoot`** del servidor virtual y cambiar la información del propietario y grupo de los archivos para que puedan manejarse sin problemas de permisos.

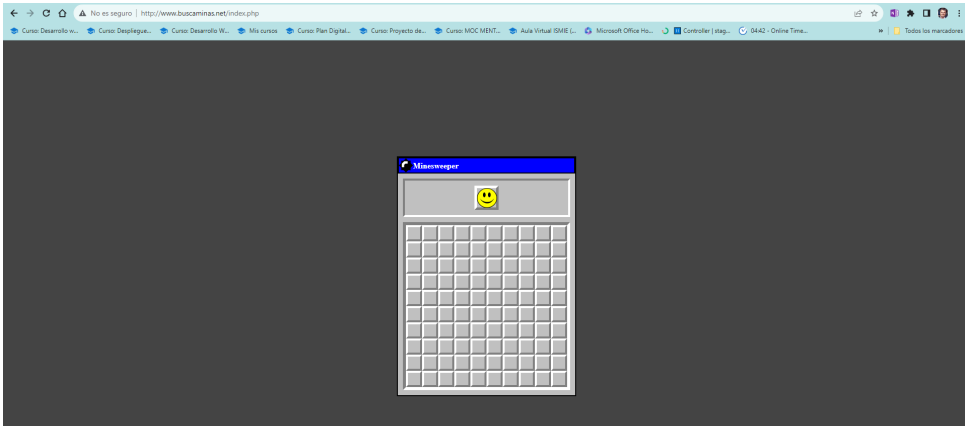
```
unzip buscaminas.zip
cd buscaminas
composer install
sudo cp -R /home/ubuntu/buscaminas /var/www
sudo chown -R www-data:www-data /var/www/buscaminas
```

No te olvides habilitar el nuevo sitio web para que el servidor Apache pueda servir las peticiones que le llegan.

```
sudo a2ensite buscaminas.net
```

Para comprobar que el sitio web se ha desplegado correctamente vamos a acceder a ella desde nuestra máquina virtual ubuntudesktop. Para ello, debemos añadir una nueva entrada al fichero de resolución de nombres de ubuntu (**`\etc\hosts`**) con la correspondencia entre el nombre del servidor **`www.buscaminas.net`** y la dirección IP donde reside.

Ahora ya podemos abrir un navegador y acceder a la URL: **`http://www.buscaminas.net`** para acceder a la página principal del sitio.



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.- Directivas de configuración

En secciones anteriores, se han ido introduciendo algunas de las directivas más útiles para configurar el comportamiento de los servidores web habilitados en una instalación de Apache. A continuación, vamos a presentar algunas directivas de configuración adicionales que se pueden usar en la definición de los servidores web.

- **`Directory`**. se utiliza para aplicar configuraciones específicas a un directorio en el servidor web. Esta directiva permite definir reglas y restricciones para ese directorio, como opciones de autorización, límites de acceso, configuraciones de seguimiento de enlaces simbólicos y más.

```
<Directory ruta_del_directorio>
    ... configuraciones especificas del directorio ...
</Directory>
```

- **`DirectoryIndex`**. Especifica el nombre del archivo que se debe buscar cuando se accede a un directorio. Por lo general, se establece en index.html, index.php, u otros archivos de índice. Por ejemplo podemos cambiar el archivo raíz del sitio de gatitos.net al archivo indice.html de la siguiente forma:

```
<VirtualHost *:80>
        DocumentRoot "/var/www/gatitos"
        ServerName gatitos.net
        ServerAlias www.gatitos.net
       <Directory /var/www/gatitos>
            DirectoryIndex indice.html
       </Directory>
</VirtualHost>
```

- **`Options`**. Establecer varias opciones de configuración para un directorio o un servidor virtual específico. Estas opciones controlan el comportamiento y las características del servidor web para las solicitudes que se manejan en ese contexto. La directiva Options puede tener múltiples valores que se pueden combinar para personalizar la configuración.
  Aquí están algunos de los valores más comunes que puedes especificar en la directiva Options:
  - **`Indexes`**: Permite o prohíbe la generación automática de listados de directorios cuando no se encuentra un archivo de índice en un directorio. Si se habilita, se mostrará una lista de archivos en el directorio si no hay un archivo de índice presente.
  - **`FollowSymLinks`**: Permite la navegación simbólica, lo que permite a los usuarios seguir enlaces simbólicos dentro del sistema de archivos. Esto habilita la capacidad de seguir enlaces simbólicos para acceder a recursos en otros directorios.
  - **`MultiViews`**: Habilita la búsqueda de archivos con nombres similares si no se encuentra un archivo con el nombre exacto en una solicitud. Por ejemplo, si se solicita "archivo" y no se encuentra, Apache buscará "archivo.html" o "archivo.php", si están presentes.

Por ejemplo podemos configurar el servidor virtual de gatitos.net para que liste los archivos de la carpeta img y para que sirva el archivo contacto.html aunque en la URL se solicite el archivo contacto.

```
<VirtualHost *:80>
        DocumentRoot "/var/www/gatitos"
        ServerName gatitos.net
        ServerAlias www.gatitos.net
       <Directory /var/www/gatitos>
            DirectoryIndex indice.html
            Options MultiViews
       </Directory>
       <Directory /var/www/gatitos/img>
           Options Indexes
       </Directory>
</VirtualHost>
```

- **`AllowOverride`**. controla qué configuraciones se pueden anular o sobrescribir mediante archivos **`.htaccess`** en un directorio específico. Los archivos **`.htaccess`** son archivos de configuración que permiten a los administradores de sitios web modificar la configuración de Apache en un directorio determinado, lo que brinda flexibilidad para ajustar la configuración en un nivel más local.
  La directiva `AllowOverride` acepta varios valores para determinar qué configuraciones pueden ser anuladas en un directorio. Algunos de los valores más comunes incluyen:
  - **`None`**: No se permiten anulaciones en ningún nivel. Los archivos **`.htaccess`** se ignoran completamente.
  - **`All`**: Se permiten todas las anulaciones. Los archivos `.htaccess` pueden sobrescribir prácticamente todas las configuraciones de Apache, incluyendo la mayoría de las opciones de configuración.
  - **`AuthConfig`**: Permite la anulación de configuraciones relacionadas con la autenticación y la autorización de usuarios, como **`AuthType`**, **`AuthName`**, **`AuthUserFile`**, etc.
  - **`FileInfo`**: Permite la anulación de configuraciones relacionadas con la información del archivo, como **`AddType`**, **`AddOutputFilter`**, **`DefaultType`**, etc.
  - **`Indexes`**: Permite la anulación de configuraciones relacionadas con la generación de listados de directorios, como **`Options +Indexes`**, **`IndexOptions`**, etc.

- **`Limit`**: Permite la anulación de configuraciones relacionadas con la limitación de acceso a través de directivas como **`Order`**, **`Deny`**, **`Allow`**, etc.
- **`Options`**: Permite la anulación de configuraciones relacionadas con las opciones del servidor web, como **`Options`**, **`DirectoryIndex`**, **`AddHandler`**, etc.

  Por ejemplo, si deseas permitir que los archivos **`.htaccess`** anulen la configuración de autenticación y las opciones del servidor en un directorio específico, puedes configurar **`AllowOverride`** de la siguiente manera:

```
<Directory /var/www/gatitos>
     AllowOverride AuthConfig Options
</Directory>
```

Esta configuración permitirá la anulación de configuraciones relacionadas con la autenticación y las opciones del servidor a través de archivos **`.htaccess`** en el directorio **`/var/www/gatitos`**. Ten en cuenta que debes configurar **`AllowOverride`** de manera segura, ya que las anulaciones pueden tener un impacto en la seguridad y el rendimiento del servidor si se utilizan de manera inadecuada.

3.- Módulos.

Caso práctico

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

pennuja (CC BY)

- 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/.`**

Debes conocer

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

Módulos

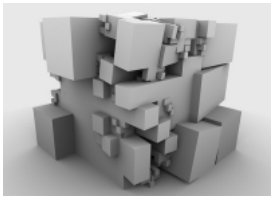
La instalación o desinstalación de un módulo no implica la desinstalación de Apache o la nueva instalación de Apache perdiendo la configuración del servidor en el proceso, simplemente implica la posibilidad de poder trabajar en Apache con un nuevo módulo o no.

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.



syntopia (CC BY)

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 Ubuntu

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 Ubuntu, también puedes ejecutar el comando **`a2enmod`** sin parámetros. La ejecución de esté 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 esté comando ofrecerá una lista de módulos a deshabilitar, escribes el módulo en cuestión y el módulo se deshabilitará.

Es posible listar los módulos estáticos que se han cargado al compilar el servidor ejecutando el comando: **`sudo apachectl -l`**

También podemos consultar los módulos activados dinámicamente consultando el directorio **`/etc/apache2/mods-enabled`**.

Una vez habilitado o deshabilitado los módulos Apache sólo reconocerá estos cambios cuando recargues su configuración, con lo cual debes ejecutar el comando: **`sudo systemctl restart apache2`**

Si la configuración es correcta y no quieres reiniciar Apache puedes recargar la configuración mediante el comando: **`sudo systemctl reload apache2`**.

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: **`systemctl restart apache2`**

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

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.

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

Como 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 l**a 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.

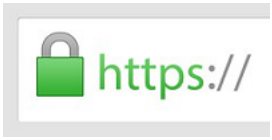
Firefox (MPL)

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.



Sean MacEntee (CC BY)

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.

TLS (Transport Layer Security) es un protocolo de seguridad diseñado para garantizar la confidencialidad, integridad y autenticación de la comunicación en Internet. A continuación, te explico cómo funciona el protocolo TLS:

1. **Inicio de la comunicación:**

   Cuando un cliente (como un navegador web) desea establecer una conexión segura con un servidor (como un sitio web), se inicia el proceso TLS. El cliente y el servidor negocian el uso de TLS y comienzan un apretón de manos (handshake).

2. **Negociación de protocolo y parámetros de cifrado:**

   Durante el apretón de manos, el cliente y el servidor acuerdan qué versión de TLS utilizarán y los algoritmos de cifrado que se emplearán para proteger la comunicación. Esto incluye algoritmos para el cifrado simétrico (como AES), algoritmos de intercambio de claves asimétricas (como RSA o Diffie-Hellman), y algoritmos de resumen (hash) para garantizar la integridad de los datos.

3. **Intercambio de claves:**

   El servidor envía su clave pública al cliente como parte del apretón de manos. El cliente utiliza esta clave para cifrar una clave de sesión que será compartida entre el cliente y el servidor. Esto permite que la comunicación se cifre de manera segura sin necesidad de compartir la clave de sesión a través de la red.

4. **Autenticación del servidor:**

   El servidor presenta su certificado digital al cliente, que contiene su clave pública y está firmado por una entidad de certificación confiable (CA). El cliente verifica la autenticidad del servidor asegurándose de que el certificado sea válido y provenga de una fuente confiable. Esto previene los ataques de intermediarios maliciosos.

5. **Cifrado y autenticación de la comunicación:**

   Una vez que se ha completado el apretón de manos y se ha compartido una clave de sesión segura, todos los datos intercambiados entre el cliente y el servidor se cifran con esa clave de sesión. También se utilizan códigos de autenticación de mensajes (MAC) para garantizar la integridad de los datos.

6. **Comunicación segura:**

   A partir de este punto, la comunicación entre el cliente y el servidor está cifrada y protegida de escuchas no autorizadas y manipulación de datos. Los datos se envían de manera segura a través de Internet.

7. **Cierre de la conexión segura:**

   Al finalizar la comunicación, se cierra la conexión de manera segura, lo que evita cualquier intento de interceptación o modificación de los datos restantes.

TLS es esencial para garantizar la seguridad de la comunicación en Internet, y es ampliamente utilizado en protocolos como HTTPS (HTTP seguro) para proteger la transmisión de datos en la web, así como en otros servicios y aplicaciones que requieren seguridad en la comunicación.

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

Módulo ssl

Ejercicio resuelto

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

Mostrar retroalimentación

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?

Mostrar retroalimentación

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.

En Ubuntu, 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 si 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:

Firefox (MPL)

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:

```
192.168.56.56 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.

Ralf S. Engelschall (BSD)

Para que nuestro host virtual **www.dawdistancia.net** sea considerado seguro por los navegadores que acceden a él, es necesario dotarlo de un certificado digital que certifique su autenticidad. Dicho certificado debería ser firmado a su vez por una autoridad certificadora (CA) de la lista preconfigurada en el navegador. En nuestro caso el certificado va a estar autofirmado por lo que el servidor enviará el certificado pero los navegadores lo considerarán "no seguro".

Para crear el certificado autofirmado haremos lo siguiente:

- Instalación del paquete openssl: (Este paso no es necesario en Ubuntu 22.04 si se ha instalado openssl con el sistema operativo)

```
sudo apt-get install openssl
```

- Genera la clave privada del servidor para generar su certificado y autofirmarlo.

```
sudo mkdir /etc/apache2/tus-ssl/
cd /etc/apache2/tus-ssl/
sudo openssl genrsa -out  seguro.key 4096
```

```
root@servubuntu:/etc/apache2/tus-ssl# openssl genrsa -out  seguro.key 4096
Generating RSA private key, 4096 bit long modulus (2 primes)
.....................................................++++
.....................................................................................
.............................................++++
e is 65537 (0x010001)
```

- Genera la solicitud de certificado digital del servidor.

```
sudo openssl req -new -key seguro.key -out seguro.csr
```

Asegúrate que usas los parámetros de creación tal y como se muestra en la imagen.

```
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]:Madrid
Locality Name (eg, city) []:Alcobendas
Organization Name (eg, company) [Internet Widgits Pty Ltd]:IESVP
Organizational Unit Name (eg, section) []:DINF
Common Name (e.g. server FQDN or YOUR name) []:www.dawdistancia.net
Email Address []:admin@virgendelapaz.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:IES Virgen de la Paz
```

- Firma el certificado con su misma clave privada. Para que el servidor fuera seguro habría que firmarlo con la clave privada de una autoridad certificadora (CA) reconocida.

```
sudo openssl x509 -req -days 365 -in seguro.csr -signkey seguro.key -out seguro.crt
```

- 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/seguro.crt
SSLCertificateKeyFile /etc/apache2/tus-ssl/seguro.key
```

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

Si no lo has hecho previamente, habilita el módulo ssl y la configuración SSL por defecto.

```
a2enmod ssl
a2ensite default-ssl
systemctl restart apache2
```

Si lanzas el navegador desde tu equipo:

1. Indicar **`https://www.dawdistancia.net`** 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:

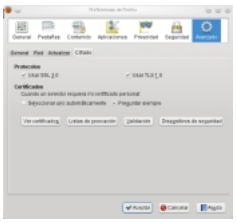
Firefox (MPL)

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:



Firefox (MPL)

- 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 > 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.

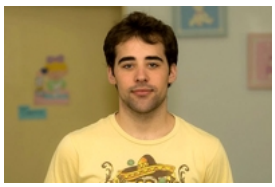
Para saber más

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.- 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/2020:17:19:18 +0200] "GET /index.html HTTP/1.1" 200 20		
Campos (especificadores)	**Definición**	**Ejemplo**
host (%h)	Identifica el equipo cliente que solicita la información en el navegador.	192.168.200.100
ident (%l)	Información del cliente cuando la máquina de éste ejecuta identd y la directiva IdentityCheck está activada.	
authuser (%u)	Nombre de usuario en caso que la URL solicitada requiera autenticación HTTP.	
date (%t)	Fecha y hora en el que se produce la solicitud al servidor. Va encerrado entre corchetes. Este campo tiene su propio formato: [dia/mes/año:hora:minuto:segundo zona]	[05/May/2020:17:19:18 +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.

5.1.- 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.

The people from the Tango! project (Dominio público)

Directivas para archivos de registro.

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

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

Sintaxis y uso de directivas para archivos de registro

Directiva TransferLog	
Sintaxis	`TransferLog nombre_fichero_archivo_registro | tubería_para_enviar_al_programa_la_información de_registro`
Uso	`TransferLog logs/acceso_a_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`

Podemos usar estas directivas para configurar los logs asociados al servidor virtual correspondientes al nombre de dominio gatitos.net, para ello editamos el fichero de configuración en el directorio sites-available:

```
# Definir un formato de registro personalizado a nivel global
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" miFormatoPersonalizado

<VirtualHost *:80>
        DocumentRoot "/var/www/gatitos"
        ServerName gatitos.net
        ServerAlias www.gatitos.net

        # Referenciar el formato de registro personalizado en el Virtual Host
        CustomLog /var/log/apache2/gatitos.net-access.log miFormatoPersonalizado
        ErrorLog /var/log/apache2/gatitos.net-error.log
</VirtualHost>
```

En este ejemplo, el formato miFormatoPersonalizado incluye:

- %h: La dirección IP del cliente.
- %l: El identificador remoto (generalmente '-').
- %u: El userID del cliente si la autenticación HTTP está en uso.
- %t: La fecha y hora de la solicitud.
- \"%r\": La línea de solicitud exacta recibida del cliente.
- %>s: El estado de la respuesta HTTP.
- %b: El tamaño del objeto devuelto al cliente, no incluye encabezados de respuesta HTTP.
- \"%{Referer}i\": La página que refirió al cliente al recurso actual.
- \"%{User-Agent}i\": El agente de usuario del cliente.

5.2.- 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 24horas

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

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

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

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

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

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

Por defecto, si no se define formato mediante ningún modificador % para guardar los archivos de registro, el sufijo nnnnnnnnn (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
```

Alias logformat predefinidos en /etc/apache2/apache2.conf

```
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.`

5.2.1.- 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

```
/usr/sbin/logrotate -d /etc/logrotate.d/apache2
```

Forzar la ejecución de logrotate

```
/usr/sbin/logrotate -f /etc/logrotate.conf
```

`/etc/cron.daily/logrotate`: Fichero tipo para ejecutar logrotate diariamente en el cron

```
#!/bin/sh

test -x /usr/sbin/logrotate || exit 0

/usr/sbin/logrotate /etc/logrotate.conf
```

Ejemplo para añadir al archivo crontab del sistema (crontab -e)

```
# Rotar logs de apache con logrotate a las 3 am

0 03 * * * root /usr/sbin/logrotate /etc/logrotate.conf > /dev/null 2>&1
```

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.`**

6.- 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:

En la aplicación de desarrollo montada por **María** se realizarán las pruebas, siendo los encargados de las mismas **Antonio** y **Carlos**.

1.- El diseño web de la aplicación recaerá en **Ana**: banners, logos ...
2.- **Juan** se dedicará a la programación del panel de control: autenticación por medio de LDAP
3.- 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 autenticacarse en la aplicación y todos empiezan a trabajar en el resto de las cosas.

Desconocida (Copyright 2011 © OpenLDAP Foundation. Privacy Statement)

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 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).

Para configurar la autenticación básica en Apache 2.4, sigue estos pasos:

- Crear el archivo de contraseñas: Primero, necesitas crear un archivo que contenga los nombres de usuario y contraseñas. Esto se hace con la herramienta htpasswd. Por ejemplo, para crear un nuevo archivo con un usuario, usarías un comando como este en tu terminal:

```
sudo htpasswd -c /etc/apache2/.htpasswdgatitos admin
```

La opción -c solo es necesario si se está creando el fichero de usuarios, si el fichero ya existe y solo se quiere añadir nuevos usuarios se debe ocuandomitir. El comando solicita el password del usuario para ser autenticado cuando quiera acceder a la parte de la web protegida.

- Configurar el archivo de configuración del servidor: Necesitas decirle a Apache qué directorio quieres proteger. Esto se puede hacer en el archivo de configuración principal de Apache (por ejemplo, httpd.conf o apache2.conf) o en un archivo .htaccess en el directorio que deseas proteger. A continuación se muestran las dos mecanismos:

Aquí tienes un ejemplo de lo que podrías añadir a tu archivo de configuración del servidor virtual de gatitos, gatitos.net.conf en la carpeta sites-available:

```
<VirtualHost *:80>
        DocumentRoot "/var/www/gatitos"
        ServerName gatitos.net
        ServerAlias www.gatitos.net
        <Directory "/var/www/gatitos">
           AuthType Basic
           AuthName "Mensaje de autenticacion"
           AuthUserFile /etc/apache2/.htpasswdgatitos
           Require valid-user
        </Directory>
</VirtualHost>
```

- **`AuthType Basic`** indica que estás utilizando la autenticación básica.
- **`AuthName`** es el mensaje que se mostrará en el cuadro de diálogo de autenticación.
- **`AuthUserFile`** debe apuntar a la ubicación del archivo .htpasswd que creaste.
- **`Require valid-user`** significa que cualquier usuario válido en el archivo .htpasswd puede acceder.

Para que el cambio de configuración tenga efecto deberás recargar la configuración de Apache

```
sudo systemctl reload apache2
```

En la autenticación HTTP Basic es muy típico utilizar archivos .htaccess en los directorios que queremos controlar el acceso. Los archivos .htaccess en Apache son archivos de configuración distribuida que permiten gestionar la configuración de Apache a nivel de directorio. Se utilizan para modificar la configuración sin necesidad de alterar los archivos de configuración principales del servidor web. Un uso común de los archivos .htaccess es para implementar la autenticación básica, que restringe el acceso a determinados recursos en tu sitio web.

En este caso, es ncesario crear un fichero .htaccess en el directorio raíz del sitio de gatitos, /var/www/gatitos con el siguiente contenido:

```
AuthType Basic
AuthName "Área restringida"
AuthUserFile /etc/apache2/.htpasswdgatitos
Require valid-user
```

Para que Apache lea y respete las configuraciones en .htaccess, la directiva AllowOverride en el archivo de configuración principal del servidor debe estar establecida en All o al menos incluir AuthConfig para el directorio en cuestión. La configuración del archivo de configuración gatitos.net.conf en el directorio /etc/apache2/sites-available sería:

```
<VirtualHost *:80>
        DocumentRoot "/var/www/gatitos"
        ServerName gatitos.net
        ServerAlias www.gatitos.net
        <Directory "/var/www/gatitos">
            AllowOverride AuthConfig
        </Directory>
</VirtualHost>
```

De nuevo para activar los cambios debes ejecutar

```
sudo systemctl reload apache2
```

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:

Acceder a la página de autentificación, autorización y control de acceso de Apache

En Apache 2.4, las directivas de control de acceso se utilizan para restringir el acceso a los recursos del servidor web. Estas directivas se han simplificado y mejorado en comparación con las versiones anteriores de Apache. Aquí te explico algunas de las directivas más importantes y cómo se utilizan:

Directivas Principales de Control de Acceso

`Require`: Esta directiva se usa para especificar qué usuarios, grupos o condiciones deben cumplirse para permitir el acceso. Ejemplos de uso:

`Require all granted`: Permite el acceso a todos.
`Require all denied`: Niega el acceso a todos.
`Require user [nombre_de_usuario]`: Permite el acceso solo al usuario especificado.
`Require group [nombre_del_grupo]`: Permite el acceso solo a los miembros del grupo especificado.**`Require all granted`**: Permite el acceso a todos.
`Require ip [dir_ip]`: Permite el acceso solo a peticiones provenientes de una dirección IP.

`Order, Allow, Deny` (Desaconsejadas en Apache 2.4): Estas directivas eran comunes en Apache 2.2, pero en Apache 2.4 se desaconseja su uso en favor de la nueva sintaxis Require. Si aún las encuentras, funcionan así:

`Order`: Define el orden de procesamiento de las directivas Allow y Deny.
`Allow`: Especifica una condición bajo la cual se permite el acceso.
`Deny`: Especifica una condición bajo la cual se deniega el acceso.

`<RequireAll>`: Esta directiva se utiliza para agrupar varias condiciones que deben cumplirse todas para permitir el acceso.

```
<RequireAll>
    Require ip 192.168.1.0/24
    Require not ip 192.168.1.5
</RequireAll>
```

Este ejemplo permite el acceso a todos en la subred 192.168.1.0/24, excepto a la dirección 192.168.1.5.

`<RequireAny>`: Si se cumplen alguna de las condiciones dentro de este bloque, se permite el acceso.

```
<RequireAny>
    Require user user1
    Require ip 192.168.1.100
</RequireAny>
```

Aquí, el acceso se permite si el solicitante es el usuario user1 o si la solicitud proviene de la IP 192.168.1.100.

<RequireNone>: Se deniega el acceso si se cumplen alguna de las condiciones dentro de este bloque.

```
<RequireNone>
    Require user user1
    Require ip 192.168.1.100
</RequireNone>
```

En este caso, se deniega el acceso si el solicitante es user1 o si la solicitud proviene de la IP 192.168.1.100.

Implementación en la Configuración de Apache
Estas directivas se pueden utilizar dentro de los archivos de configuración de los servidores o dentro de archivos .htaccess. Por ejemplo, para proteger el fichero contacto.html en el servidor de gatitos , puedes usar:

```
<VirtualHost *:80>
        DocumentRoot "/var/www/gatitos"
        ServerName gatitos.net
        ServerAlias www.gatitos.net
        <Files "contacto.html">
           Require ip 192.168.56.1
        </Files>
</VirtualHost>
```

6.1.- 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 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`** o **`www.autenticacion-ldap.empresa-proyecto.com`**

Firefox (MPL)

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.

En la siguiente sección se puede ver un ejemplo de despliegue de la aplicación OpenCart. (**Anexo I.- Despliegue aplicación OpenCart.**)

Se deberá tener instalado un servidor LAMP, en la unidad didáctica "Implantación de arquitecturas web", se vio la instalación del entorno Apache, MySQL y PHP, en Ubuntu.

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

Página instalación WordPress

Debes conocer

En el siguiente enlace podemos ver una descripción detallada de cómo instalar OpenCart sobre Ubuntu 18.04: