UNIVERSIDAD DE GRANADA

Tecnologías Web

3º Grado en Ingeniería Informática

Guión de prácticas
Instalación de un servidor web

© Prof. Javier Martínez Baena
Dpto. Ciencias de la Computación e I. A.
Universidad de Granada

Departamento de
Ciencias de la Computación
e Inteligencia Artificial

1. Objetivo

El objetivo de la práctica es instalar un servidor Web en una máquina con sistema operativo GNU/Linux. Para esta práctica el alumno puede llevar su propio ordenador o bien utilizar un PC del aula de prácticas. El alumno también podrá instalar el software en otros sistemas operativos aunque en ese caso deberá buscar los pasos equivalentes a los explicados en el guión. En cualquier caso se recomienda que el alumno se familiarice con el S.O. GNU/Linux ya que será el que utiliza el *host* en el que se van a hacer las entregas de todas las prácticas de la asignatura.

Concretamente se instalará un servidor web Apache junto con un intérprete de PHP. Ya existen paquetes que instalan de forma más o menos directa e integrada todos estos servicios que suelen ser, en muchos casos, la configuración estándar de los servidores de aplicaciones web. Estos paquetes (también llamados *stacks*) son conocidos por los acrónimos LAMP, MAMP, WAMP o XAMP:

- L=Linux, M=MacOS, W=Windows, X=cualquier SO.
- A = Apache.
- M = MySQL.
- P = PHP (o Perl o Python).

Observe que además de Apache y PHP también incluyen algún sistema gestor de BBDD como, por ejemplo, MySQL. En una sesión futura haremos la instalación y configuración de un servidor MySQL. También es habitual que incluyan algunas herramientas adicionales (por ejemplo `PHPMyAdmin`) o incluso otros gestores de BBDD (`MariaDB`) o de documentos (`MongoDB`), etc. Pueden verse algunos ejemplos en los siguientes enlaces:

- `http://www.ampps.com/` (Windows, Linux, MacOS)
- `https://www.apachefriends.org/es/index.html` (Windows, Linux, MacOS)
- `https://www.mamp.info/en/` (Windows, MacOS)
- `http://www.wampserver.com/` (Windows)
- ...

En esta práctica **no debe usar ninguno de esos *stacks*** de aplicaciones y hará la instalación de cada servicio de forma manual e independiente. De esta forma tendrá una idea más clara del software que está utilizando y de cómo interactúa entre sí. Se recomienda también que siga este mismo procedimiento en su ordenador personal.

2. Entorno de trabajo

Puede realizar esta práctica en su ordenador personal o en el aula de prácticas. Si lo hace en su ordenador tiene dos opciones:

- Hacer la instalación en su S.O. habitual o en una máquina virtual.
- Descargar la imagen de máquina virtual que hay en las aulas de prácticas desde un enlace que encontrará en PRADO.

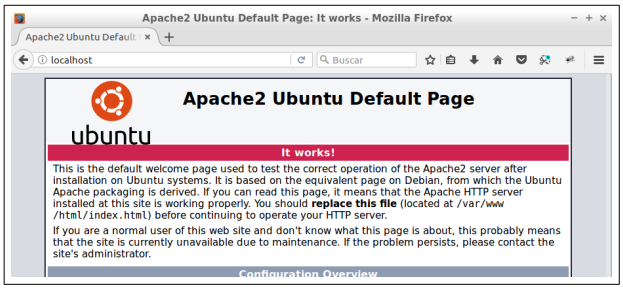
Si hace la instalación en el aula de prácticas deberá replicarlo más adelante en su entorno de trabajo habitual para poder hacer las siguientes prácticas con comodidad.

En PRADO tiene un documento de instrucciones generales de prácticas con más información.

3. Instalación del servidor web Apache

Dependiendo del S.O. y de la distribución que esté usando, podría variar el nombre de este paquete. En la página web `https://httpd.apache.org` dispone de los fuentes del servidor y de enlaces de descarga para distintas plataformas. En el caso de Ubuntu, el paquete que contiene el servidor se llama **`apache2`** y se puede instalar desde los repositorios oficiales:

```
apt-get install apache2
```



Una vez instalado se puede probar que funciona

accediendo a la dirección **http://localhost**. Si la instalación ha sido correcta verá una página como la de la figura.

3.1. Iniciar y detener el servidor web

Un S.O. de tipo Unix puede encontrarse en diferentes niveles de ejecución (*runlevel*):

- Nivel de ejecución 0: Apagado.
- Nivel de ejecución 1: Monousuario (sólo usuario *root*; no es necesaria la contraseña). Se suele usar para analizar y reparar problemas.
- Nivel de ejecución 2: Multiusuario sin soporte de red.
- Nivel de ejecución 3: Multiusuario con soporte de red.
- Nivel de ejecución 4: Como el runlevel 3, pero no se suele usar.
- Nivel de ejecución 5: Multiusuario en modo gráfico (X Windows).
- Nivel de ejecución 6: Reinicio.

Dependiendo del *runlevel*, se ejecutan unos servicios u otros durante el arranque de la máquina. Habitualmente un PC de escritorio levanta un servidor gráfico por lo que su *runlevel* se establece en 5. Un servidor no necesita levantar entorno gráfico y su *runlevel* puede ser 3.

En la carpeta **/etc** verá que existen varias carpetas nombradas **rcX.d** en las que se incluyen los *scripts* que permiten iniciar los distintos servicios de la máquina dependiendo del nivel de ejecución en el que se encuentre. Normalmente, puesto que el mismo servicio puede ser levantado en distintos niveles de ejecución, y para evitar duplicar dicho *script* en varios directorios, el contenido de las carpetas es un enlace al *script*, que se encuentra realmente en **/etc/init.d**.

En esta práctica no debe hacer nada relacionado con este subsistema dado que la instalación del software del servidor Apache ya lo ha configurado adecuadamente, sin embargo, es posible que cuando tenga que hacer una instalación de un servidor real en producción sí tenga que tener en cuenta aspectos relativos a la forma en que se inicia el servidor durante el arranque de la máquina.

Además, cada vez que modifique los ficheros de configuración del servidor Apache, deberá reiniciarlo para que sean cargados por él.

Puede usar los siguientes comandos para iniciar, detener o ver el estado de un servicio desde un terminal. En nuestro caso, el servicio se llama **apache2**:

service apache2 status	*Muestra el estado del servicio*
service apache2 stop	*Detiene el servicio*
service apache2 start	*Inicia el servicio*
service apache2 restart	*Reinicia el servicio (lo detiene y lo inicia)*

3.1.1. El sistema systemd

"**service**" en realidad es un *wrapper* para facilitar a los administradores las tareas habituales para iniciar y detener servicios. Este sistema permite trabajar de forma transparente con distintos entornos para la gestión de servicios. La mayor parte de los sistemas GNU/Linux actuales usan **systemd**, que es un conjunto de demonios, bibliotecas y herramientas para la gestión de servicios.

La forma de trabajar con **systemd**, en lugar de con **service**, para las tareas indicadas antes es la siguiente:

systemctl status apache2	*Muestra el estado del servicio*
systemctl stop apache2	*Detiene el servicio*
systemctl start apache2	*Inicia el servicio*
systemctl restart apache2	*Reinicia el servicio (lo detiene y lo inicia)*

3.2. Abrir puertos en el servidor

Es muy habitual que el ordenador que ejecuta el servidor web esté protegido frente a intrusiones con un *firewall*. Recuerde que debe abrir los puertos en los que está escuchando el servidor Apache. Por defecto esos puertos son el 80 (HTTP) y el 443 (HTTPS).

En Ubuntu puede usar el comando **ufw** (que es un *front-end* para **iptables**) para abrir o cerrar puertos así como habilitar o deshabilitar el *firewall*:

ufw disable	*Deshabilita el cortafuegos por completo*
ufw enable	*Habilita el cortafuegos*
ufw status	*Muestra el estado del cortafuegos*
ufw status numbered	*Muestra el estado del cortafuegos numerando las reglas*
ufw allow http	*Habilita las conexiones al puerto del servicio http*
ufw allow 80	*Habilita las conexiones al puerto 80*
ufw deny http	*Deniega conexiones al servicio http (silencioso)*
ufw reject http	*Deniega conexiones al servicio http (devuelve error)*
ufw delete <nºregla>	*Borra una regla del firewall*

Pruebe a abrir y cerrar los puertos y acceda desde otras máquinas para ver el comportamiento. Recuerde que en las máquinas virtuales del aula de prácticas los puertos están redirigidos.

3.3. Administración de Apache con apachectl

Apache dispone de una herramienta en línea de órdenes para su administración que permite:

- Iniciar y detener el servicio.
- Comprobar el estado del servicio.
- Comprobar si los ficheros de configuración son correctos sin iniciar o reiniciar el servicio.

Para ello, puede ejecutar:

apachectl start	*Muestra el estado del servicio*
apachectl stop	*Detiene el servicio*
apachectl restart	*Reinicia el servicio*

Para comprobar el estado del servicio debe tener instalado algún navegador en modo texto ya que apachectl lo usa para hacer una petición HTTP al servidor. Puede instalar, por ejemplo, **w3m**:

```
apt-get install w3m
```

Una vez hecho esto, puede usar los siguientes comandos para comprobar el estado del servidor:

apachectl status	*Muestra el estado del servicio*
apachectl fullstatus	*Muestra el estado del servicio*

Durante esta práctica deberá modificar la configuración del servidor múltiples veces y, en ocasiones, puede que se equivoque con la sintaxis. Si es el caso, al reiniciar el servidor obtendrá un mensaje de error. El mensaje de error puede variar dependiendo de la forma de reiniciar. Además, dependiendo del *front-end* que use para el reinicio, el servicio puede interrumpirse o no.

Suponga que tiene un error sintáctico en **apache2.conf** y que ejecuta:

```
service apache restart
```

Obtendría un mensaje de error similar a este:

```
Job for apache2.service failed because the control process exited with error code.
See "systemctl status apache2.service" and "journalctl -xe" for details.
```

Además de aportar poca información, como efecto secundario el servidor Apache ha quedado deshabilitado ya que se ha hecho un “stop” seguido de un “start” y este último ha fallado por no poder cargar la configuración. En cambio, si ejecuta:

```
apachectl restart
```

el mensaje aportará información más adecuada y el servicio no es detenido. Por ejemplo:

```
apache2: Syntax error on line 174 of /etc/apache2/apache2.conf: Expected </Directory> but saw
</Divrectory>
Action 'restart' failed.
The Apache error log may have more information.
```

Independientemente del *front-end* que utilice para reiniciar el servicio, con **apachectl** puede

verificar la sintaxis de los ficheros de configuración sin necesidad de detener e iniciar el servicio:

```
apachectl configtest
```

4. Configuración de Apache

El servidor hace uso de dos directorios en el sistema: uno para alojar las páginas web y otro para mantener su configuración global. Las páginas se alojan, por defecto, en **`/var/www/html`** y la configuración está en **`/etc/apache2`**.

Dentro de la carpeta de configuración también pueden variar los nombres de los ficheros o los subdirectorios en función de la distribución. En Ubuntu los principales archivos se organizan así:

```
/etc/apache2/
├── apache2.conf          Fichero principal de configuración
├── envvars               Variables de configuración de entorno para Apache
├── ports.conf            Configuración de los puertos que escucha el servidor
│
├── mods-available        Configuraciones de módulos disponibles
├── mods-enabled          Configuración de los módulos de apache activos
│   ├── *.load
│   └── *.conf
│
├── conf-available        Ficheros de configuración disponibles
├── conf-enabled          Ficheros de configuración activos
│   └── *.conf
│
├── sites-available       Configuraciones de hosts virtuales disponibles
└── sites-enabled         Configuraciones de hosts virtuales activos
    └── *.conf
```

Los sistemas Debian (Ubuntu está basado en Debian) fragmentan la configuración del servidor el múltiples ficheros para facilitar las tareas de administración. En otros sistemas podría encontrar un único fichero con todos los parámetros de configuración (se suele llamar **`httpd.conf`**). A continuación se comentan algunos aspectos de la configuración.

Los tres ficheros principales de la configuración son estos:

- **`apache2.conf`**. Este fichero es el que carga el servidor web cuando se inicia, en él se encuentran algunas directivas globales y otras para cargar el resto de ficheros de configuración.
- **`envvars`**. Este fichero contiene variables de entorno para la ejecución de Apache. Cuando el S.O. ejecuta el servidor carga previamente este *script* para configurar la *shell* en la que ejecuta Apache.
- **`ports.conf`**. Este fichero contiene algunas directivas para indicarle al servidor qué puertos debe escuchar. Normalmente escuchará el puerto 80 para las peticiones HTTP sin cifrado y el 443 si está habilitado el esquema HTTPS.

También hay tres grupos de directorios con diversos ficheros de configuración organizados como:

- **`conf-XXX`** (configuración general). Ficheros con parámetros globales de la configuración. Se pueden mantener distintos aspectos de la configuración en distintos ficheros para facilitar la administración del servidor.
- **`mods-XXX`** (configuración de módulos). Ficheros de configuración de módulos del servidor web instalados en el sistema.
- **`sites-XXX`** (configuración de *hosts* virtuales). Le permite al servidor comportarse como si hubiese instalados varios servidores web en una misma máquina. Esto puede ocurrir cuando una misma IP tiene asignados varios nombres de dominio y queremos que las páginas servidas sean diferentes según el nombre de dominio. Otro caso habitual de uso es disponer de distintos servicios web dependiendo del puerto por el que accedemos al servidor (por defecto 80).

Cada uno de esos grupos de directorios está formado por dos directorios:

- **XXX-available**. Ficheros de configuración disponibles en el sistema pero que no tienen porqué estar en uso. Estos ficheros no se usan al iniciar el servidor web.
- **XXX-enabled**. Enlaces simbólicos a los ficheros en **XXX-available**. Estos ficheros son cargados al iniciar el servidor web.

De esta forma, en los directorios **XXX-available** tenemos configuraciones disponibles y en los **XXX-enabled** tenemos las configuraciones que efectivamente son cargadas cuando se inicia el servidor.

4.1. Configuración principal

El fichero de configuración principal es **/etc/apache2/apache2.conf**. En él se definen algunos parámetros y se hace la inclusión del resto de ficheros de configuración (en particular, los que hay en **XXX-enabled**). El propio fichero incluye instrucciones y documentación detallada. Se describen a continuación algunos parámetros a modo de ejemplo:

Parámetro	Descripción
ServerRoot	Directorio en el que se almacenan los ficheros de configuración. Por defecto vale **/etc/apache2**
Timeout	Número de segundos para esperar antes de cancelar una petición. Se usa, por ejemplo, al recibir datos desde un cliente (tiempo de espera hasta que llegan paquetes TCP) o al enviar datos a un cliente (tiempo de espera hasta recibir ACK del envío).
KeepAlive	Permitir o no conexiones persistentes (varias peticiones en una misma conexión TCP).
KeepAliveTimeout	Tiempo de espera para cerrar la conexión en el caso de que estas sean persistentes.
AccessFileName	Nombre del fichero de configuración por directorios por si hay directrices específicas para esos directorios. Por defecto vale **.htaccess**
ErrorLog	Nombre del fichero de *log* donde se almacenan los errores.
LogLevel	Tipo de mensajes de error que se almacenan en el *log*.
LogFormat	Definición del formato de los mensajes de error para el *log*.
Hostnamelookup	Indica si para registrar entradas en el fichero de *log* hay que resolver el nombre de dominio o se registra la dirección IP.

Además, en este fichero podrá ver directivas de tipo **Include** o **IncludeOptional** que sirven para cargar el resto de ficheros de configuración.

Por último, podrá ver también directivas de la forma:

```
<Directory ...>
    ...
</Directory>
```

Estas se utilizan cuando queremos definir alguna configuración específica solo para algún directorio concreto. Por ejemplo, podemos ver que se incluye esto en la configuración:

```
<Directory />
        Options FollowSymLinks
        AllowOverride None
        Require all denied
</Directory>
```

Con esto se indica que, para el directorio raíz del sistema de ficheros:

- Se permite seguir enlaces simbólicos (**Options FollowSymLinks**).
- No se permite la configuración por directorios (ficheros **.htaccess**) (**AllowOverride None**).

- Se deniega a Apache el acceso al sistema raíz (**`Require all denied`**).

También se incluye esto otro:

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

Con esto se indica que, para el directorio **`/var/www`** (este es el directorio por defecto para almacenar los documentos HTML, PHP, etc.):

- Se permite la configuración por directorios (ficheros **`.htaccess`**) (**`AllowOverride All`**).
- Se permite a Apache acceder al directorio (**`Require all granted`**).

4.2. Configuración de módulos

Una de las características de Apache es que dispone de una amplia gama de módulos que se pueden o no cargar según las necesidades que tengamos. Los ficheros de configuración de estos módulos se encuentran en los directorios **`mods-enabled`** y **`mods-available`**. Se recomienda siempre hacer las modificaciones sobre los ficheros disponibles en **`mods-available`** para, a continuación, activar los cambios si es necesario. Podemos mirar en la carpeta **`mods-available`** para ver qué módulos tenemos disponibles y en la carpeta **`mods-enabled`** para ver los que tenemos activos en este momento. A modo de ejemplo, algunos módulos disponibles son estos:

`alias.conf`	Permite la definición de alias en los ficheros de configuración.
`deflate.conf`	Permite que el servidor envíe las páginas comprimidas al cliente.
`dir.conf`	Secuencia de búsqueda de ficheros cuando se solicita una carpeta.
`file_cache.conf`	Gestiona la caché para ficheros que se piden muchas veces.
`http2.conf`	Soporte para la versión 2 de HTTP.
`mime.conf`	Gestiona los tipos MIME.
`proxy.conf`	Configuración de Apache como proxy.
`userdir.conf`	Configuración de directorios **`public_html`** de los usuarios del sistema.

Activando y desactivando ficheros de configuración

Dispone de un comando **`a2enmod`** y de otro **`a2dismod`** para activar o desactivar algún módulo concreto de entre los disponibles. Estos comandos lo que hacen es, básicamente, crear o quitar el enlace entre las carpetas **`mods-available`** y **`mods-enabled`** (aunque podrían hacer alguna tarea adicional). Más adelante veremos algún ejemplo de activación de módulos.

·▷ Ejercicio 1: Activación y desactivación de un módulo

Si accede a la URL **`https://void.ugr.es/tweb`** verá que en el navegador se muestra una lista de ficheros en lugar de una página web. Esto es así porque al no haber ningún fichero concreto en la URL Apache entiende que lo que estamos pidiendo es el listado del directorio solicitado. Hay un módulo encargado de gestionar esto, es decir, de lo que debe hacer Apache cuando la URL es una ruta que no contiene ningún fichero concreto. El módulo se llama **`autoindex`**.

Desactive el módulo con el comando:

```
a2dismod autoindex
```

Ahora vuelva a solicitar la URL para ver el comportamiento de Apache. Observe que para que tenga efecto el cambio en la configuración del Apache es necesario reiniciar el servidor (ver sección 3.1).

Antes de continuar vuelva a activar el módulo ya que este es considerado esencial por Apache y, además, será necesario para hacer algunos ejercicios de esta práctica:

```
a2enmod autoindex
```

4.3. Ficheros de configuración adicionales

Estos ficheros se encuentran en los directorios **conf-enabled** y **conf-available**. Se recomienda siempre hacer las modificaciones sobre los ficheros disponibles en **conf-available** para, a continuación, activar los cambios si es necesario.

Este esquema de configuración (fraccionada en múltiples ficheros) permite, por una parte, mantener una separación entre la configuración por defecto del servidor y una posible configuración particular y, por otra, facilitar el mantenimiento de la configuración modularizándola. Por ejemplo, en esta práctica deberá modificar algunos aspectos de la configuración. En lugar de modificar los ficheros que se han instalado con Apache, es mejor opción crear un fichero de configuración propia y activarlo o desactivarlo según si queremos que tenga o no efecto. De esa forma siempre será posible restaurar la configuración inicial si hiciese falta.

Activando y desactivando ficheros de configuración

Dispone de un comando **a2enconf** y de otro **a2disconf** para activar o desactivar algún fichero de configuración concreto de entre los disponibles. Estos comandos lo que hacen es, básicamente, crear o quitar el enlace entre las carpetas **conf-available** y **conf-enabled** (aunque podrían hacer alguna tarea adicional).

·▷ Ejercicio 2: Activación y desactivación de un fichero de configuración

En sucesivos ejercicios de esta práctica deberá modificar la configuración de Apache. Para ello deberá crear su propio fichero de configuración llamado **/etc/apache2/conf-available/miconfig.conf**. Cree dicho fichero, aunque por ahora permanezca vacío. Una vez creado pruebe a activarlo con:

```
a2enconf miconfig
```

Compruebe que se ha creado el enlace en **/etc/apache2/conf-enabled**. A continuación desactivelo con:

```
a2disconf miconfig
```

Siempre que modifiquemos un fichero de configuración global de Apache es necesario reiniciar el servidor para que tengan efecto dichos cambios (ver sección 3.1).

4.4. Configuración de hosts virtuales

Estos ficheros se encuentran en los directorios **sites-enabled** y **sites-available**. Se recomienda siempre hacer las modificaciones sobre los ficheros disponibles en **sites-available** para, a continuación, activar los cambios si es necesario.

Aquí se definen parámetros para configurar distintos *hosts* virtuales en un mismo servidor web. Los casos habituales de uso son:

- Disponemos de varios nombres de dominio asociados a una misma dirección IP y queremos que nuestro servidor web se comporte de forma distinta en función del nombre de dominio por el que el usuario ha llegado a él. Se definiría un *host* virtual para cada nombre de dominio.
- Deseamos que, dependiendo del puerto al que se conecta el usuario, el servidor web se comporte de forma diferente. El puerto por defecto para escuchar conexiones HTTP es el 80 pero podríamos hacer que nuestro servidor web respondiese también ante conexiones a otros puertos (ver fichero **ports.conf**). Al disponer de diferentes puertos de entrada al servidor web, podemos hacer que se comporte como si hubiese un servidor web distinto en cada uno de ellos.

El fichero **000-default.conf** contiene la configuración del *host* por defecto. La configuración inicial define un *host* virtual para cualquier nombre de dominio asociado a la máquina y para el puerto 80.

Por defecto también se instala otro fichero (**default-ssl.conf**) para configurar un *host* virtual que atiende al puerto 443 (conexiones HTTPS). Esto lo veremos más adelante.

Activando y desactivando ficheros de configuración

Dispone de los comandos **a2ensite** y **a2dissite** para habilitar o deshabilitar un *host* virtual y funcionan de forma análoga a **a2enconf** y **a2disconf**.

·▷ Ejercicio 3: Creación de un *host* virtual

En este ejercicio debe modificar la configuración para crear un nuevo *host* virtual en el puerto 555. Dicho *host* exportará un sistema de archivos diferente al que está configurado por defecto (**/var/www/html**).

Cree una nueva carpeta **/var/www/host555** y cree en ella un documento llamado **index.html** con el siguiente contenido:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>Nuevo host</title>
</head>
<body>
  <h1>Bienvenido al nuevo host virtual</h1>
</body>
</html>
```

A continuación deberá:

- Copiar el fichero **/etc/apache2/sites-available/000-default.conf** en otro llamado **/etc/apache2/sites-available/nuevohost.conf**.
- Edite el nuevo fichero para:
  - Cambiar el puerto de escucha al 555
  - Cambiar el **DocumentRoot** a **/var/www/host555**
- Modifique el fichero **/etc/apache2/ports.conf** e incluya una línea para que Apache escuche las peticiones al puerto 555:
  - **Listen 555**
- Active el nuevo *host*:
  - **a2ensites nuevohost**
- En caso de que desee acceder al nuevo *host* desde un ordenador diferente no olvide abrir el puerto 555 en su *firewall*.

Si ha hecho bien el ejercicio podrá acceder al nuevo *host* virtual con la URL **http://localhost:555**.

Finalmente, deshabilite el *host* con:

```
a2dissites nuevohost
```

4.5. Ficheros de log

En **/var/log/apache2** se encuentran los ficheros de *log* del servidor Apache. Son ficheros de texto en los que el servidor Apache va anotando todo lo que hace (esencialmente las peticiones de páginas que recibe el servidor y la respuesta que este da).

·▷ Ejercicio 4: Ver el fichero de *log*

Ejecute la siguiente orden en un terminal y a continuación acceda al servidor desde un navegador para ver los datos que se registran:

```
tail -f /var/log/apache2/access.log
```

5. Ejercicio: configuración básica de Apache

Aunque Apache ya tiene una configuración por defecto, es muy frecuente personalizar algunos aspectos tras su instalación. Para ilustrar mejor este proceso realizaremos varios ejercicios mostrando algunas de las opciones frecuentes.

Para comenzar, crearemos en la carpeta **`/var/www/html`** una estructura de carpetas y ficheros como la que sigue:

```
├── biografias
│   ├── ada.html
│   ├── ada.jpg
│   ├── alan.html
│   ├── alan.jpg
│   ├── bjarne.html
│   └── bjarne.jpg
├── hardware
│   ├── alternativo.html
│   ├── cpu
│   │   ├── amd
│   │   │   ├── ryzen-5-5600x
│   │   │   └── ryzen-7-5800x
│   │   └── intel
│   │       ├── i7-11700k
│   │       ├── i9-10900k
│   │       └── i9-11900k
│   │           └── index.html
│   ├── cpu.html
│   ├── disco.html
│   ├── index.html
│   ├── memoria.html
│   ├── placa.html
│   └── teclado.html
├── index.html
├── proveedores
│   ├── amazon
│   │   └── productos.html
│   ├── elcorteingles
│   │   └── productos.html
│   ├── mediamarkt
│   │   └── productos.html
│   └── milistado.html
└── tiendas
```

Puede descargar un fichero *zip* con la estructura ya creada para abreviar el ejercicio desde la web de la asignatura.

Observe que, una vez copiados, es posible acceder a estos recursos desde un navegador a partir de la URL **`http://localhost/…`**, por ejemplo, son válidas las siguientes URL:

- **`http://localhost/`**
- **`http://localhost/hardware`**
- **`http://localhost/proveedores/amazon/productos.html`**
- …

·▷ Ejercicio 5: Diferencia de comportamiento

Acceda a las URL que se listan a continuación:

1. **`http://localhost`**
2. **`http://localhost/biografias`**
3. **`http://localhost/hardware`**

Puede comprobar que en algún caso se muestra una página web y en otro un listado de ficheros y directorios. ¿A qué se debe la diferencia de comportamiento?

·▷ Ejercicio 6: Desactivar listado de directorios

A partir de este ejercicio deberá modificar la configuración de su servidor Apache. En el ejercicio 2 creó su propio fichero de configuración (**/etc/apache2/conf-available/miconfig.conf**). Asegúrese de que lo hizo correctamente y edítelo cada vez que deba actuar sobre la configuración en el resto de ejercicios de este guión. Recuerde que cada vez que modifique ese fichero debe reiniciar el servidor Apache.

Si accede a **http://localhost/biografias** podrá ver un listado con los ficheros incluidos en el directorio. En la mayoría de las ocasiones se considera una mala práctica permitir que los usuarios del sitio web vean listados como ese ya que le pueden confundir y se pueden mostrar detalles de implementación o datos sensibles de nuestro sitio web.

Para activar o desactivar listados de este tipo se utiliza la opción **Indexes** con la siguiente sintaxis:

```
Options Indexes
Options +Indexes
Options -Indexes
```

Si ha funcionado bien debería obtener un error *403 / Forbidden* al acceder a la URL en lugar del listado del directorio.

Si no ha funcionado puede deberse a que:

- No ha incluido **Options** dentro de la directiva **Directory** adecuada.
- La sintaxis es errónea. Solución: escriba bien.
- No ha reiniciado el servidor web. Solución: consulte la sección 3.1 y reinicie el servidor web.
- No está activo el fichero de configuración. Solución: consulte la sección 4.3 y active su fichero de configuración personalizada.

·▷ Ejercicio 7: Cambiar el documento mostrado por defecto

Si accede a **http://localhost/hardware**, el navegador le mostrará el documento **index.html**. Modifique la configuración para que, en lugar de ese documento, se muestre el documento **alternativo.html**.

Para hacer esto debe usar la siguiente directiva:

```
DirectoryIndex <F1 F2 ...>
```

Con ella, le estamos indicando a Apache que en el caso de que la URL acceda a un directorio (y no a un fichero concreto) busque en él alguno de los ficheros F1, F2, ... y que muestre el primero que exista en dicho directorio.

Incluya en esa directiva de búsqueda únicamente el nuevo fichero **alternativo.html**.

Observe que es posible seguir accediendo al anterior listado de hardware si escribimos la URL completa del fichero que lo contiene: **http://localhost/hardware/index.html**.

·▷ Ejercicio 8: Crear un alias de un directorio

En ocasiones puede ser de interés definir alias de algunas rutas de nuestra aplicación. Para hacerlo usaremos la siguiente directiva:

```
Alias “alias” “ruta”
```

Con ella, le estamos indicando a Apache que si recibe una petición a **http://localhost/alias**, sirva el fichero o carpeta indicado en **ruta**. Observe que **ruta** se refiere al sistema de ficheros local.

Algunos ejemplos de ocasiones en las que esto puede resultar útil:

- Para simplificar el acceso a URL muy largas o poco inteligibles o memorizables por los usuarios. Supongamos que en un sitio web tenemos la siguiente URL:
  http://miservidor.com/listados/gestion/profesores/infraestructuras/index.html

  Si incluimos la directiva:
  ```
  Alias "/prof" "/var/www/html/listados/gestion/profesores/infraestructuras"
  ```
  ahora usaríamos la URL:
  http://miservidor.com/prof
- Para acceder a directorios que están fuera del sistema de ficheros accesible por el servidor

web. Supongamos que tenemos un directorio con imágenes fuera de **/var/www/html**. Para permitir el acceso habría que incluir:

```
Alias "/image" "/ftp/pub/image"
<Directory "/ftp/pub/image">
    Require all granted
</Directory>
```

En este ejercicio puede comprobar que en la carpeta **hardware/cpu** hay una pequeña jerarquía de carpetas en la que se incluiría una descripción de diferentes CPU. Cree un alias para que cuando el usuario acceda a la URL **http://localhost/hardware/bestcpu** Apache se comporte como si le hubiese pedido la URL **http://localhost/hardware/cpu/intel/i9-11900k**.

Si hizo correctamente el ejercicio anterior (**DirectoryIndex**) verá un listado con un único fichero: **index.html**. Explique porqué no se muestra su contenido directamente.

·▷ Ejercicio 9: Restaurar la configuración inicial temporalmente

Anule la configuración que ha estado realizando y restaure la configuración inicial. Use los comandos vistos en la sección 4.3. Vuelva a activarla a continuación. Compruebe el efecto de activar y desactivar la configuración visitando las distintas páginas del sitio web.

5.1. Protección de directorios

Los directorios que se están sirviendo pueden protegerse para permitir el acceso desde determinadas direcciones IP o nombres de dominio o para que puedan acceder sólo determinados usuarios. Para ello se utiliza la directiva **Require**:

Require all granted	Permite el acceso desde cualquier lugar y a cualquier usuario
Require all denied	Deniega el acceso
Require ip <dirección>	Permite el acceso solo desde determinadas direcciones IP (separadas por espacios en blanco)
Require not ip <dirección>	Deniega el acceso desde determinadas direcciones IP
Require host <nombre>	Permite el acceso solo desde determinados nombres de dominio (separados por espacios en blanco)
Require not host <nombre>	Deniega el acceso desde determinados nombres de dominio

·▷ Ejercicio 10: Impedir el acceso a un directorio

Continúe con la configuración del servidor e impida que los usuarios tengan acceso al directorio **proveedores/** evitando así que la información contenida en él quede expuesta. Para ello use la directiva "**Require all denied**".

Para comprobar el resultado cargue de nuevo la URL **http://localhost/proveedores** y compruebe si ha funcionado. Si lo ha hecho bien debería obtener un error 403-Forbidden al acceder a la URL.

Al igual que ocurre con el resto de directivas, el efecto se aplica también a subdirectorios. Compruebe que es así accediendo a la URL **http://localhost/proveedores/amazon.**

5.1.1. Protección con usuario y clave

Para proteger un directorio con usuario y clave, primero debe crear un fichero de usuarios y claves. Este fichero se debería almacenar (siempre que sea posible) fuera de la estructura de directorios que exporta el servidor web por motivos de seguridad. Para ello use el comando **htpasswd**:

```
htpasswd -c /var/www/claves.ht usuario
```

Sea cuidadoso porque si el fichero **claves.ht** ya existía, con la opción **-c** lo borra y crea uno nuevo. Si lo que desea es añadir nuevos usuarios a ese fichero ejecute:

```
htpasswd /var/www/claves.ht otrousuario
```

Para borrar un usuario de ese fichero use la opción **-D**:

```
htpasswd -D /var/www/claves.ht usuario
```

La configuración de acceso por clave también puede hacerse a nivel global en el servidor. Para proteger una carpeta, añada a su directiva **Directory** el siguiente contenido:

```
AuthType Basic
AuthName "Acceso restringido"
AuthUserFile "/var/www/claves.ht"
Require user usuario
```

Con ello se da permiso únicamente al usuario indicado. Puede incluir múltiples usuarios, grupos de usuarios, etc. También se puede restringir el acceso por IP o por dominio. Si desea dar acceso a cualquiera de los usuarios que estén almacenados en el fichero de claves puede usar:

```
Require valid-user
```

La opción **AuthName** se conoce como "*Realm*" y es un texto que tiene dos funciones:

- Mostrarlo al usuario en la ventana que solicita las credenciales.
- Si el navegador accede a diferentes carpetas protegidas en las que coincide su *realm*, la clave solo la pedirá la primera vez. Por tanto, es conveniente que use un texto distinto para aplicaciones diferentes evitando así que usuarios autenticados en otra aplicación tengan acceso a la suya.[1]

Debe tener en cuenta que con este método de autenticación, el usuario y la clave se transmiten en texto plano (sin cifrado), por lo que en caso de que se esté usando un esquema http (y no https) supone un riesgo de seguridad importante.

·▷ Ejercicio 11: Acceso selectivo a directorios

Proteja los directorios de los tres proveedores del ejercicio. Para ello debe crear tres usuarios diferentes (uno por cada proveedor) y configurar Apache para que cada proveedor pueda acceder solo a su directorio en base a la siguiente tabla:

Usuario	Clave	Realm	Directorio
amazon	**clave1**	**Acceso restringido**	**/proveedores/amazon**
elcorteingles	**clave2**	**Acceso restringido**	**/proveedores/elcorteingles**
mediamarkt	**clave3**	**Acceso restringido**	**/proveedores/mediamarkt**

Observe que una vez autenticado el usuario, no se vuelve a pedir la clave. Para comprobarlo realice esta secuencia de visitas a la web:

1. **http://localhost/biografias**
2. **http://localhost/proveedores/amazon** (Solicitará usuario y clave)
3. **http://localhost/biografias**
4. **http://localhost/proveedores/amazon** (No solicitará de nuevo datos de acceso)

Uno de los inconvenientes de este método de autenticación es que no permite “desautenticarse”. La única forma de conseguir que vuelva a solicitar identificación es cerrar el navegador y abrirlo de nuevo.

Otra forma de acceder a estos recursos es incluir el usuario y clave en la propia URL de la forma:

```
http://amazon:clave1@localhost/proveedores/amazon
```

Observe que esta forma podría ser, de nuevo, una brecha de seguridad al tener que escribir (de forma visible) la clave y también porque dependiendo del servidor, cabe la posibilidad de que se estén almacenando en algún fichero de log las credenciales.

1 Las últimas versiones de algunos navegadores tienen un comportamiento distinto e intentan soslayar este problema de seguridad por lo que es posible que las credenciales se pregunten cada vez que se accede a un recurso distinto.

5.2. Configuración a nivel de usuario

Con bastante frecuencia, los desarrolladores de aplicaciones web no tienen acceso a la configuración global del sistema (ficheros en **`/etc/apache2`**). Apache posibilita que estos usuarios sin privilegios de administrador puedan definir configuraciones particulares para determinados directorios (generalmente los directorios de las aplicaciones que están desarrollando). Y todo esto sin necesidad de modificar los ficheros de configuración global. Como esto puede suponer un riesgo de seguridad, es necesario que Apache permita este modo de configuración por lo que, si deseamos permitir este comportamiento para determinados directorios (aquellos en los que los usuarios van a desplegar sus aplicaciones) debemos incluir la opción **`AllowOverride`** en la correspondiente directiva **`Directory`**. Esta opción puede tomar diferentes valores dependiendo del tipo de opciones que deseamos permitir que se apliquen a dicho directorio (y subdirectorios):

- **`AllowOverride None`**. No permite la configuración por directorios en absoluto. Por motivos de seguridad, esta es la configuración de Apache por defecto. Podrá comprobarlo si mira en el fichero **`/etc/apache2/apache2.conf`**.
- **`AllowOverride All`**. Permite configurar todo aquello que sea posible mediante este sistema. Tenga en cuenta que la personalización de opciones por parte de los usuario no debe ser ilimitada ya que hay algunas que podrían suponer riesgos de seguridad para el servidor. Por ejemplo, opciones como **`ServerRoot`** o **`Timeout`** no pueden modificarse con este modelo de configuración.
- **`AllowOverride AuthConfig`**. Permite incluir opciones relativas a la autenticación de usuarios.
- **`AllowOverride Indexes`**. Permite incluir opciones relativas al indexado de directorios.
- ...

Una vez hecho esto (y reiniciado el servidor), podemos añadir un fichero con el nombre **`.htaccess`** en cada directorio en el que deseemos tener una configuración personalizada en algún aspecto. Este fichero contiene las opciones que deseamos modificar (sin necesidad de usar la directiva **`Directory`**, ya que este dato está implícito en la ubicación del fichero **`.htaccess`**).

·▷ Ejercicio 12: Proteger el acceso a un directorio desde `.htaccess`

Puede comprobar que la carpeta **`tiendas/`** no tiene ninguna restricción de acceso configurada. Si deseamos impedir el acceso a ella debemos incluir una opción **`Require`** tal y como hemos visto en ejercicios anteriores. Sin embargo, en este ejercicio vamos a indicar esa opción en el fichero **`tiendas/.htaccess`**. El contenido del mismo será:

```
Require all denied
```

Si ahora accede a la URL **`http://localhost/tiendas`** debería obtener un código de error 404-*Forbidden*.

Si en lugar de un error 404 obtiene un error 500 (*Internal server error*), puede deberse a un error durante el procesamiento del fichero **`.htaccess`** por parte de Apache:

- Error sintáctico. En este caso corrija la sintaxis.
- Opciones no permitidas en **`.htaccess`**. En este caso debe asegurarse de que la configuración global permite la configuración mediante ficheros **`.htaccess`**. Para ello añada las opciones adecuadas en su fichero de configuración **`/etc/apache2/conf-available/miconfig.conf`**.

Puede consultar el fichero de log de errores (**`/var/log/error.log`**) para ver los detalles del error producido.

Observe que el contenido de los ficheros **`.htaccess`** tiene efecto sin necesidad de reiniciar el servidor web. El uso de esta forma de configuración es menos eficiente que el uso de ficheros de configuración globales, aunque en situaciones en las que el desarrollador web no es administrador del sistema es la única forma de personalizar la configuración del servidor web. Esta es una característica que diferencia claramente al servidor Apache del servidor NGINX, su competidor más directo.

6. El directorio public_html de los usuarios

Hasta ahora ha estado creando contenidos para su servicio web alojándolos en **/var/www/html**. Sin embargo, esta carpeta pertenece al usuario administrador y está totalmente desaconsejado acceder a un sistema como *root* si no es para realizar tareas de administración. Además, es probable que si está desplegando una aplicación en un servidor de terceros no tenga privilegios de administrador por lo que le resultaría imposible instalar aplicaciones web en él.

Para resolver esta situación, lo que se hace es indicarle a Apache que, independientemente de que sirva páginas alojadas en **/var/www/html**, también sirva páginas alojadas en otras ubicaciones como, por ejemplo, directorios alojados en los *home* de los usuarios del sistema. Concretamente, por defecto, se establece que Apache exporta los directorios alojados en las carpetas llamadas **public_html** de los *home* de cada usuario.

Si en el servidor existiese un usuario llamado "**curioso**", este podría crear una carpeta en su *home* llamada **public_html** y alojar en ella sus páginas web. Así, cualquiera podría visitarlas escribiendo la URL **http://nombredelservidor/~curioso**

De nuevo, Apache tiene deshabilitada esta función por defecto. Para que sea operativa debe activar un módulo llamado **userdir** y configurarlo convenientemente. Dicha configuración se hace en el fichero **/etc/apache2/mods-available**.

·▷ Ejercicio 13: Activar el directorio **public_html** del usuario **tweb**

Para comenzar el ejercicio compruebe que esta funcionalidad no está activa. Acceda a la URL **http://localhost/~tweb**. Debería obtener un error 404-*Not Found*.

Active el directorio **public_html** del usuario **tweb** siguiendo estos pasos:

1. Asegúrese de que se identifica como el usuario tweb (y no como administrador).
2. Cree el directorio **public_html** en el *home* del usuario **tweb**.
3. Active la configuración de Apache que permite servir dicho directorio.

Ahora debería funcionar la URL **http://localhost/~tweb**

·▷ Ejercicio 14: Mover la práctica al directorio **public_html** del usuario **tweb**

En este ejercicio debe mover todos las carpetas y ficheros HTML de los ejercicios previos a la carpeta **/home/tweb/public_html**. Para ello siga estos pasos:

1. Mueva los ficheros.
2. Deshabilite su configuración personal (**/etc/apache2/conf-available/miconfig.conf**).
3. Verifique que tiene acceso a alguno de los directorios sin restricciones (por ejemplo **http://localhost/~tweb/hardware**).
4. Cree ficheros **.htaccess** en cada carpeta que requiera configuración específica y copie las opciones necesarias desde la configuración antigua (**miconfig.conf**).

7. Instalación de PHP

El servidor que se ha instalado permite servir páginas web estáticas (ficheros **.html**). Si necesita que el servidor pueda ejecutar aplicaciones PHP, es necesario añadir:

- Un intérprete PHP.
- El módulo de Apache para que este sepa cómo usar el intérprete PHP.

Para comprobar si ya dispone de este software haga lo siguiente:

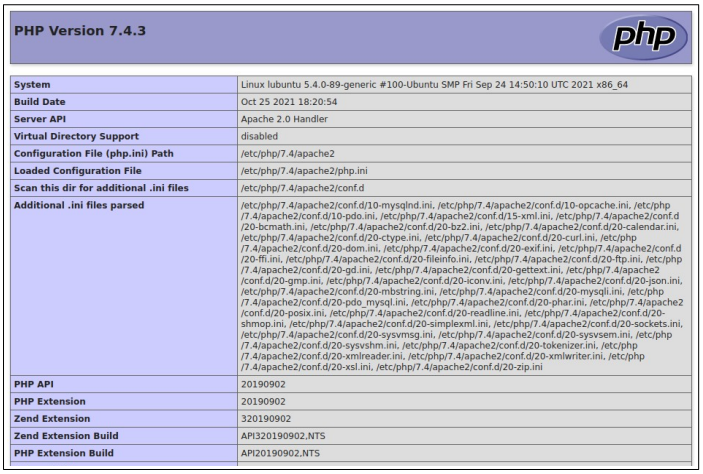
1. Cree un fichero llamado **/var/www/html/phpinfo.php** con el contenido:

```
<?php phpinfo(); ?>
```

2. Abra un navegador y pruebe a acceder a la URL **http://localhost/phpinfo.php**

Si ve una página en blanco se debe a que no tiene instalado el software para ejecutar *scripts* PHP. En este caso, si mira el código fuente de la página, podrá ver el contenido del fichero **phpinfo.php**.

En caso de que tenga instalado el módulo de PHP debería ver una página similar a la de la derecha, en donde el resultado de la ejecución de **phpinfo.php** muestra distintos aspectos de la configuración del módulo de PHP.



PHP Version 7.4.3

System	Linux lubuntu 5.4.0-89-generic #100-Ubuntu SMP Fri Sep 24 14:50:10 UTC 2021 x86_64
Build Date	Oct 25 2021 18:20:54
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.4/apache2
Loaded Configuration File	/etc/php/7.4/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.4/apache2/conf.d
Additional .ini files parsed	/etc/php/7.4/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.4/apache2/conf.d/10-opcache.ini, /etc/php/7.4/apache2/conf.d/10-pdo.ini, /etc/php/7.4/apache2/conf.d/15-xml.ini, /etc/php/7.4/apache2/conf.d/20-bcmath.ini, /etc/php/7.4/apache2/conf.d/20-bz2.ini, /etc/php/7.4/apache2/conf.d/20-calendar.ini, /etc/php/7.4/apache2/conf.d/20-ctype.ini, /etc/php/7.4/apache2/conf.d/20-curl.ini, /etc/php/7.4/apache2/conf.d/20-dom.ini, /etc/php/7.4/apache2/conf.d/20-exif.ini, /etc/php/7.4/apache2/conf.d/20-ffi.ini, /etc/php/7.4/apache2/conf.d/20-fileinfo.ini, /etc/php/7.4/apache2/conf.d/20-ftp.ini, /etc/php/7.4/apache2/conf.d/20-gd.ini, /etc/php/7.4/apache2/conf.d/20-gettext.ini, /etc/php/7.4/apache2/conf.d/20-gmp.ini, /etc/php/7.4/apache2/conf.d/20-iconv.ini, /etc/php/7.4/apache2/conf.d/20-json.ini, /etc/php/7.4/apache2/conf.d/20-mbstring.ini, /etc/php/7.4/apache2/conf.d/20-mysqli.ini, /etc/php/7.4/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.4/apache2/conf.d/20-phar.ini, /etc/php/7.4/apache2/conf.d/20-posix.ini, /etc/php/7.4/apache2/conf.d/20-readline.ini, /etc/php/7.4/apache2/conf.d/20-shmop.ini, /etc/php/7.4/apache2/conf.d/20-simplexml.ini, /etc/php/7.4/apache2/conf.d/20-sockets.ini, /etc/php/7.4/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.4/apache2/conf.d/20-sysvsem.ini, /etc/php/7.4/apache2/conf.d/20-sysvshm.ini, /etc/php/7.4/apache2/conf.d/20-tokenizer.ini, /etc/php/7.4/apache2/conf.d/20-xmlreader.ini, /etc/php/7.4/apache2/conf.d/20-xmlwriter.ini, /etc/php/7.4/apache2/conf.d/20-xsl.ini, /etc/php/7.4/apache2/conf.d/20-zip.ini
PHP API	20190902
PHP Extension	20190902
Zend Extension	320190902
Zend Extension Build	API320190902.NTS
PHP Extension Build	API20190902,NTS

De nuevo, dependiendo del S.O. que tengamos, el proceso de instalación puede variar. En Ubuntu este software está disponible en los repositorios oficiales:

```
apt-get install php
```

Si sigue sin ver la pantalla anterior, pruebe a instalar este otro paquete (o equivalente):

```
apt-get install libapache2-mod-php
```

Una vez instalados los paquetes, ya deberíamos poder ejecutar *scripts* PHP desde Apache.

Ejercicio 15: Prueba de ejecución de PHP

Cree un fichero llamado **/var/www/html/informacion.php** con el siguiente contenido:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Ejemplo con PHP</title>
</head>
<body>
  <h1>Usted se está conectando desde:</h1>
  <p>Dirección IP: <?php echo $_SERVER['REMOTE_ADDR'] ?? 'Sin definir';?></p>
  <p>Nombre de dominio: <?php echo $_SERVER['REMOTE_HOST'] ?? 'Sin definir';?></p>
  <p>Agente de usuario: <?php echo $_SERVER['HTTP_USER_AGENT'] ?? 'Sin definir';?></p>
  <h1>Datos del servidor web:</h1>
  <p>Dirección IP: <?php echo $_SERVER['SERVER_ADDR'] ?? 'Sin definir';?></p>
  <p>Nombre de dominio: <?php echo $_SERVER['SERVER_NAME'] ?? 'Sin definir';?></p>
  <p>Software: <?php echo $_SERVER['SERVER_SOFTWARE'] ?? 'Sin definir';?></p>
  <p>Protocolo: <?php echo $_SERVER['SERVER_PROTOCOL'] ?? 'Sin definir';?></p>
  <p>Firma del servidor: <?php echo $_SERVER['SERVER_SIGNATURE'] ?? 'Sin definir';?></p>
</body>
</html>
```

a continuación acceda a él desde un navegador con la URL **http://localhost/informacion.php**. Si el módulo de PHP se ha instalado correctamente debería visualizar alguna información sobre su ubicación y sobre el servidor.

7.1. Configuración del módulo de PHP

En Ubuntu, la configuración de este módulo está en **/etc/php/VERSION/apache2/php.ini** (donde **VERSION** dependerá de la versión del intérprete instalada, en nuestro caso es la 7.4). El fichero está autodocumentado por lo que es muy sencillo hacer cambios.

Algunos parámetros que pueden resultar de interés:

- Controlar el tipo de mensajes de error o avisos que se muestran al ejecutar código PHP desde el navegador. Dependiendo de si estamos en un servidor de desarrollo o de un servidor en producción deberemos tener una configuración u otra.
  Variables relevantes: **error_reporting, display_errors, display_startup_errors, log_errors**
- Tiempo máximo de ejecución permitido para un script.
  Variables: **max_execution_time, max_input_time**
- Limitación de la memoria que puede consumir un script.

Variables: `memory_limit`

- Si se permite o no subir ficheros desde un navegador.
  Variables: `file_uploads, upload_tmp_dir, max_file_uploads`
- Tamaño máximo de los ficheros que se pueden subir desde el navegador.
  Variables: `upload_max_filesize`
- Cantidad máxima de información que se puede transferir en un `POST`.
  Variables: `post_max_size`.
- Parámetros para configurar módulos de PHP (acceso a BBDD, ...)
- ...

En este curso es importante tener activa la visualización de todo tipo de mensajes de error que puedan ser de ayuda durante el proceso de desarrollo de software.

▷ Ejercicio 16: Activar visualización de mensajes de error en el navegador

Para comprobar que los mensajes de error se muestran correctamente, siga los siguientes pasos:

1. Cree un fichero `/var/www/html/phperror.php` con el siguiente contenido:

   ```
   <?php En PHP no se puede escribir en lenguaje natural ?>
   ```

2. Edite el fichero de configuración de PHP para que no muestre ningún tipo de mensaje de error (suponga que esta es la configuración básica de un servidor en producción) y cargue el fichero `phperror.php` desde un navegador y mire el resultado.
3. Edite de nuevo el fichero de configuración de PHP para que muestre todo tipo de mensajes de error (servidor de desarrollo). Cargue de nuevo el fichero `phperror.php` y observe el comportamiento.

7.2. Activar la carga de ficheros PHP desde public_html

En la configuración de PHP, por defecto, está deshabilitada la posibilidad de ejecutar *scripts* PHP alojados en el `public_html` por motivos de seguridad. Si desea poder incluir *scripts* PHP en esa ubicación, edite el fichero de configuración de dicho módulo de Apache (`php.conf`, `php7.2.conf`, etc) y asegúrese de que no hay ninguna directiva que lo impida. Este fichero podría tener código similar a este:

```
<IfModule mod_userdir.c>
    <Directory /home/*/public_html>
        php_admin_flag engine Off
    </Directory>
</IfModule>
```

Como ve, por defecto se deshabilita la ejecución de código PHP de los usuarios. Si es así: coméntelo y reinicie el servidor web.

▷ Ejercicio 17: Carga de ficheros PHP desde `public_html`

Mueva el fichero `/var/www/html/phperror.php` a `/home/tweb/public_html/` y a continuación intente cargarlo desde `http://localhost/~tweb/phperror.php`. Configure el servidor para que lo permita.

8. Configuración de un servidor web seguro (HTTPS)

Llegado a este punto, ya dispondrá de un servidor web Apache completamente operativo con el que trabajar. Ya puede implementar aplicaciones web sin mayor inconveniente, tanto sin son simples páginas web estáticas (HTML+CSS) como si son generadas por un servidor mediante algún lenguaje de *scripting* (PHP, Python, etc.). Sin embargo, uno de los aspectos que debe cuidar mucho hoy día es el de la seguridad de sus aplicaciones. Aparte de que la aplicación, a nivel de implementación, pueda incluir mecanismos para evitar usos inadecuados o accesos indebidos es vital que el propio servidor incluya también algunos mecanismos de seguridad básicos y genéricos para cualquier contenido que esté sirviendo.

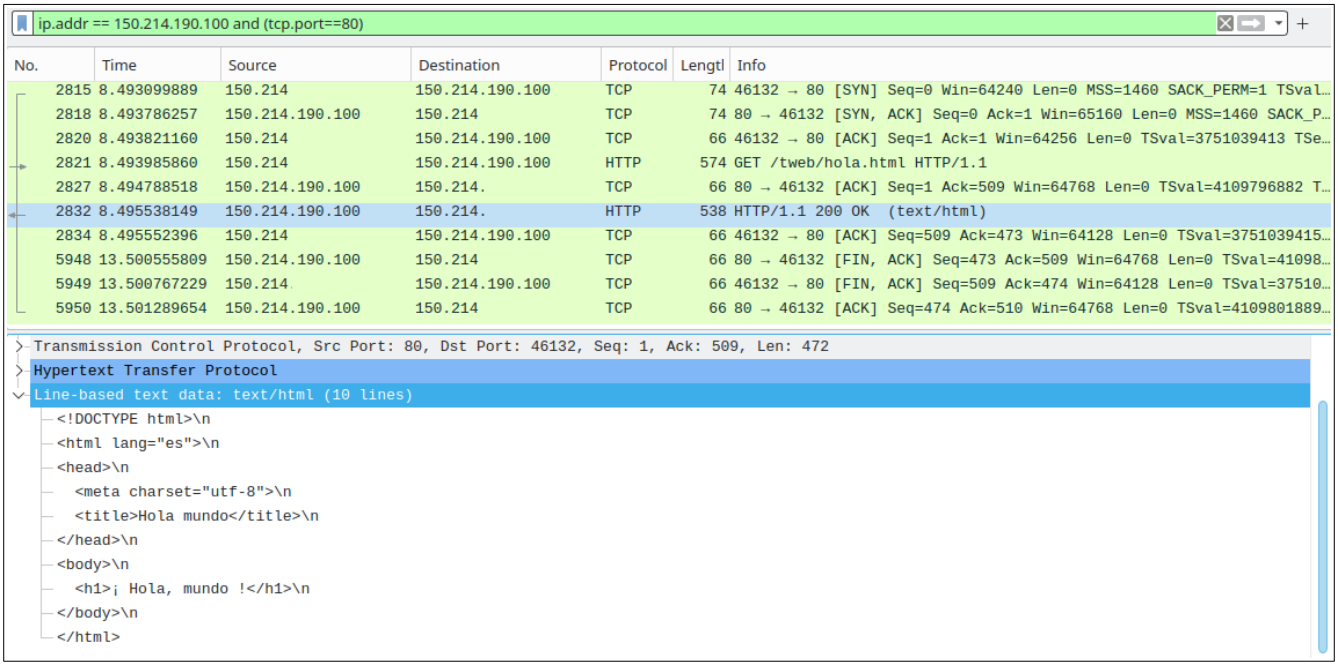
En particular, algo que debería hacer si tiene que instalar su propio servidor web, es configurarlo

para que todas las comunicaciones con él estén cifradas. De esta forma evitará ataques que podríamos calificar de triviales como, por ejemplo, el robo de credenciales.

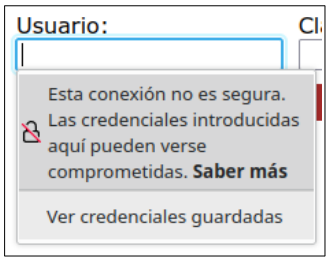
Para poder hacer algunos ejemplos de estas prácticas, en **`void.ugr.es`** puede acceder a algunas páginas usando el esquema HTTP. Si accede a **http://void.ugr.es/tweb/hola.html** verá que la caja con la URL se muestra como la que ve en la figura de la derecha.

Observe que aparece un candado tachado: eso indica que se está usando el esquema HTTP y que, por lo tanto, toda la comunicación cliente-servidor se hace en texto plano, sin cifrado de ningún tipo. Cualquiera que esté observando los paquetes que circulan entre ambas máquinas puede ver la información.

Si hace uso de alguna herramienta de captura de tráfico de red podrá verlo claramente. En la siguiente figura se muestra el tráfico de red con Wireshark[2] por el puerto 80 con el servidor **`void.ugr.es`**:



Observe que desde Wireshark vemos el contenido transmitido sin mayor problema. La gran brecha de seguridad que tenemos es más evidente si lo que estamos transfiriendo entre cliente y servidor es información sensible. Si accede a la URL **http://void.ugr.es/tweb/login.php** verá una página que simula un formulario para autenticarse en un sitio web.

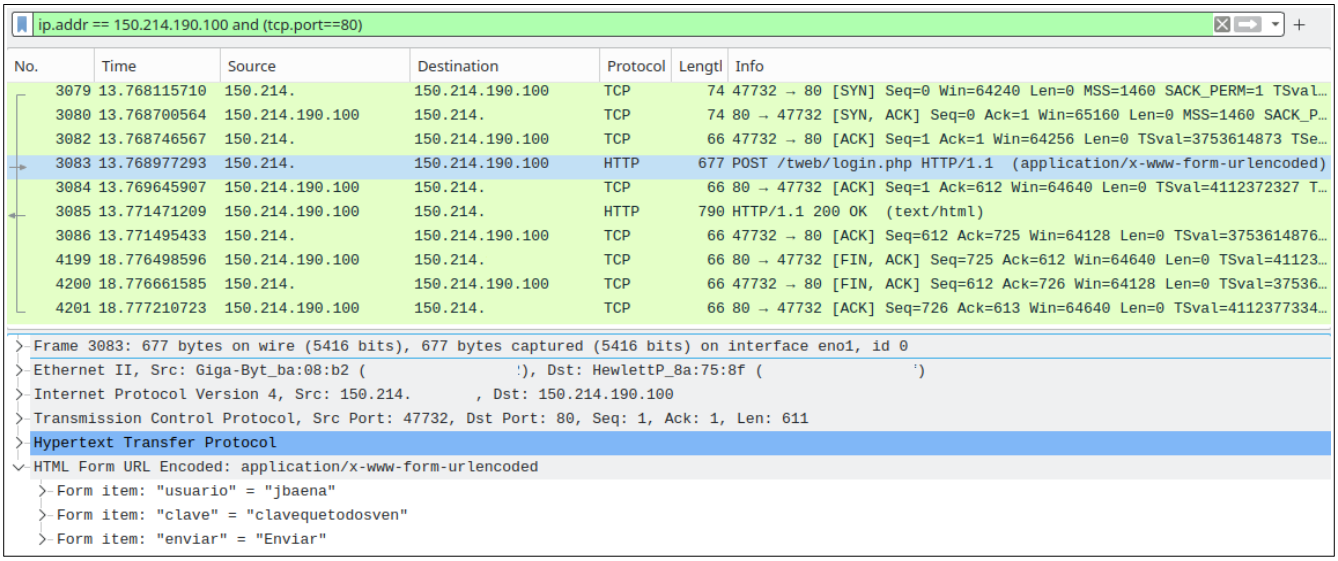


Observe que junto al candado tachado aparece una llave. Esto se debe a que el navegador ha detectado que el formulario está diseñado para autenticarse en una web y por tanto da la opción al usuario de almacenar las credenciales para no tener que escribirlas en futuros accesos. Además, al haber detectado el navegador esto, cuando pulsa en alguna de las cajas verá un mensaje como el de la figura de la derecha que hace aún más evidente que si seguimos adelante estamos

2 https://www.wireshark.org/

comprometiendo la seguridad de nuestros datos. El navegador está advirtiendo del riesgo que supone enviar credenciales en texto plano, sin cifrado, por la red. Aún así, como el servidor o la aplicación web no nos da ninguna otra opción, proseguimos, introducimos nuestras credenciales y pulsamos el botón de "Enviar":

Si hay alguien monitorizando la red e intercepta nuestros paquetes acabamos de "cederle gentilmente" nuestras credenciales. La siguiente figura muestra el tráfico de red en Wireshark, en el que podemos ver que en uno de los mensajes aparecen nuestro nombre de usuario y clave transmitidos:



Actualmente todos los servidores en internet deberían funcionar con HTTPS. Para obligarnos a ello hay iniciativas como, por ejemplo, la de Google que penaliza en sus algoritmos de posicionamiento a aquellas páginas que siguen usando HTTP. Además, ya se han dado casos de sanciones[3] a sitios web por usar HTTP en lugar de HTTPS al cometerse una infracción del artículo 32 del Reglamento General de Protección de Datos[4].

Cabe destacar que este esquema no afecta al desarrollo de las aplicaciones web ya que es transparente para ellas. Se trata únicamente de interponer una capa intermedia de cifrado para proteger el tráfico y no afecta a los mensajes del protocolo HTTP. HTTPS viene de "*HTTP over SSL*" (o TSL, que es una actualización de SSL). El cifrado lo incluye todo excepto la IP del servidor y el puerto de acceso al mismo, ya que estos son necesarios para la comunicación TCP/IP y deben ser visibles para hacer el enrutado de los mensajes.

Por tanto, tras la instalación y configuración básica del servidor web, lo que procede es activar el esquema HTTPS. Para ello lo primero que necesitamos es crear la infraestructura necesaria para el uso de TSL como capa de encriptación. Cuando un cliente solicita una conexión, se establece una negociación inicial con el siguiente método:

1. El cliente (navegador) solicita una conexión con el servidor y este le envía su certificado público SSL.
2. El cliente verificará el certificado para asegurarse de que el servidor es quien dice ser. El cliente genera un número aleatorio, lo cifra con la clave pública del servidor y se lo envía de vuelta.

3 https://www.aepd.es/es/documento/ps-00185-2020.pdf
4 https://www.aepd.es/es/informes-y-resoluciones/normativa

3. El servidor recibe el número (cifrado) y a partir de él genera una clave de sesión que envía al cliente.
4. A partir de este momento, cliente y servidor usarán dicha clave para cifrar todas las comunicaciones entre ambos.

Por defecto, Apache tiene activo el esquema HTTP por lo que no hace esta negociación cuando le llegan peticiones. Para hacer esta negociación y cifrar las comunicaciones hay que configurar Apache de manera que:

- Disponga de una clave privada y un certificado SSL público.
- Reconozca cuando una petición requiere cifrado y cuando no. Generalmente las peticiones que llegan por el puerto 80 no requieren cifrado y las que llegan por el puerto 443 sí.

8.1. Creación de clave privada y certificado público autofirmado

Una primera aproximación para resolver el problema del cifrado es que nosotros mismos creemos un certificado público y una clave privada. Podemos hacerlo con el siguiente comando:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/apache2/ssl/apache-selfsigned.key -out /etc/apache2/ssl/apache-selfsigned.crt
```

En donde:

- **`req -x509`** Indicamos que deseamos usar el estándar X.509 que especifica, entre otras cosas, el formato del certificado[5].
- **`-nodes`** Se usa para omitir el uso de una clave para leer el certificado. Si no se pone, Apache necesitaría que introdujésemos la clave cada vez que se reinicie y esto no es viable.
- **`-days 365`** Indicamos el tiempo de validez del certificado.
- **`-newkey rsa:2048`** indicamos que vamos a generar una nueva clave y un certificado usando RSA de 2048 bits.
- **`-keyout y -out`** Indicamos los ficheros en donde almacenar la clave y el certificado creados.

A continuación **`openssl`** nos hará algunas preguntas sobre la identidad del servidor (ubicación del servidor, nombre de dominio, email, etc.) y creará los dos ficheros. A continuación habría que configurar Apache para que sepa usar ambos ficheros y poder cifrar las comunicaciones. Concretamente hay que:

- Activar el módulo SSL:
  - **`a2enmod ssl`**
- Asegurarse de que en el fichero **`ports.conf`** se incluye la escucha al puerto 443, que es el habitual para usar este esquema.
- Configurar un host virtual en el puerto 443. Ya disponemos de un fichero preconfigurado para esto y bastaría con editarlo e indicar la ubicación de los ficheros de firma y certificado:
  - Editar **`/etc/apache2/sites-available/default-ssl.conf`**

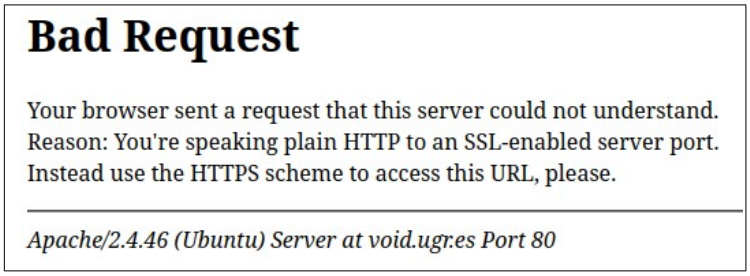
    ```
    SSLCertificateFile      /etc/apache2/ssl/apache-selfsigned.crt
    SSLCertificateKeyFile   /etc/apache2/ssl/apache-selfsigned.key
    ```
- Activar el host virtual:
  - **`a2ensite default-ssl`**
- Reiniciar Apache:
  - **`service apache2 restart`**

8.2. Creación de clave privada y certificado público mediante una CA

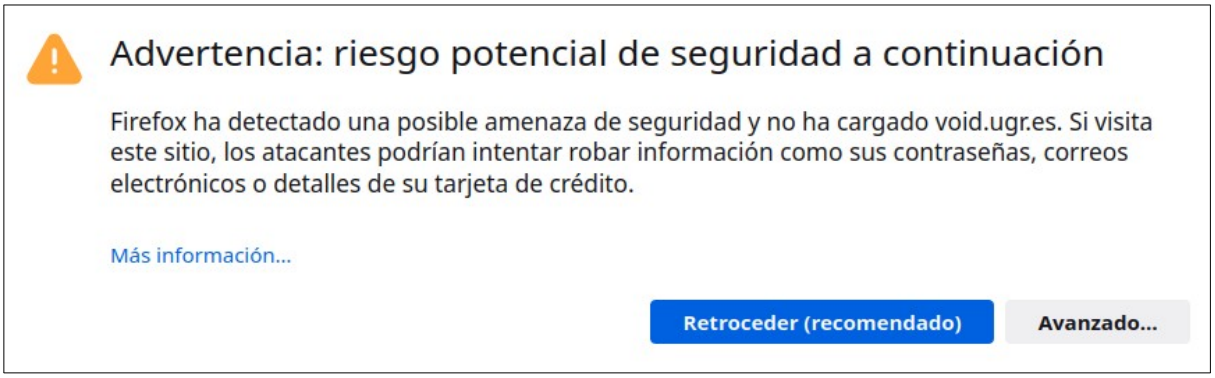
Para ilustrar esta sección se ha configurado un servidor virtual de Apache en el puerto 444 de **`void.ugr.es`** que usa un certificado autofirmado siguiendo los pasos explicados en la sección anterior. Dicho puerto únicamente permite el acceso por HTTPS por lo que si intentamos acceder a

5 https://es.wikipedia.org/wiki/X.509

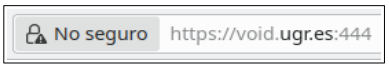
la URL http://void.ugr.es:444 obtendremos el mensaje siguiente:



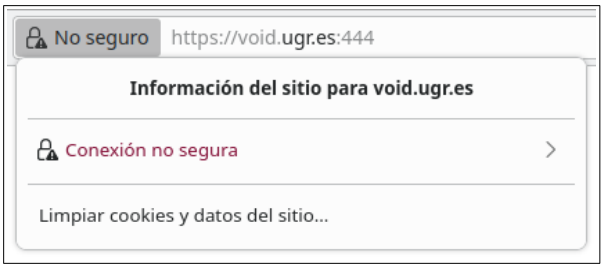
Si accedemos a https://void.ugr.es:444 (observe la "s" de "https") obtenemos el mensaje:

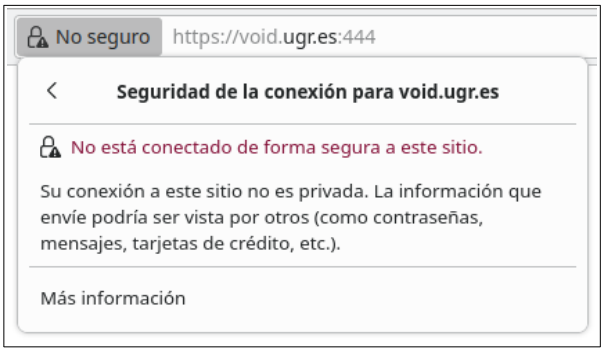


Además, en la caja con la URL vemos el icono del candado de esta forma:

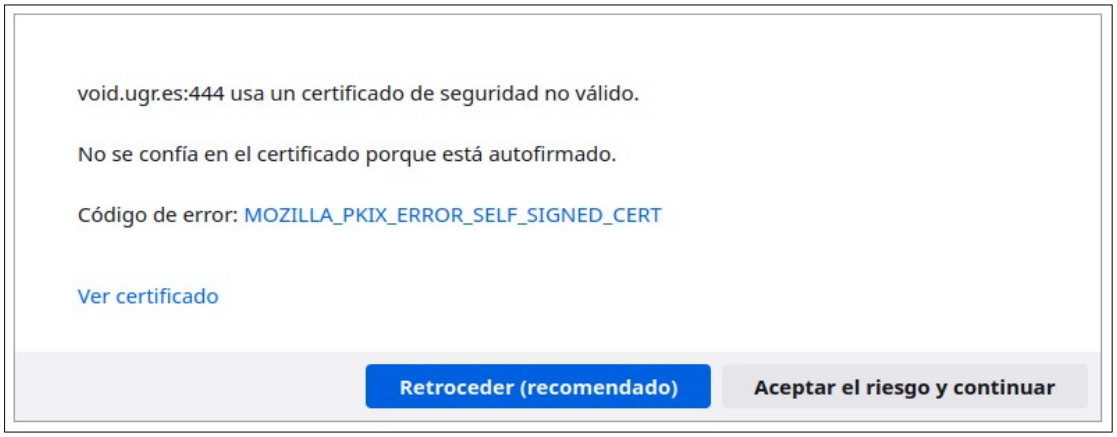


y si pulsamos sobre él se amplia la información:





Para poder seguir navegando por el sitio web hay que decírselo de forma expresa al navegador pulsando en "Avanzado...". Se mostrará información adicional y botones para continuar o no:



A partir de ahí podemos navegar sin problemas por el sitio web con las comunicaciones cifradas. Pero, si las comunicaciones están cifradas ¿a qué se deben esas advertencias? ¿qué riesgo existe?

Uno de los problemas que se nos pueden presentar en internet es que alguien suplante la identidad de un tercero con el que deseamos hacer alguna transacción. Imagine que un hacker instala un servidor y él mismo crea un certificado y lo firma. A continuación consigue acceder al ordenador de un usuario final (o incluso a nodos DNS de internet) y modifica su configuración para que cuando desde su navegador acceda a la URL de su banco, el nombre de dominio se resuelva dirigiéndolo hacia la IP del servidor web del atacante. La información que intercambian ambas máquinas está cifrada pero el servidor está suplantando la identidad del banco.

Por tanto, no es suficiente cifrar las comunicaciones para que nuestros datos sean seguros. También hace falta cerciorarse de que el servidor al que nos estamos conectando es quien dice ser realmente.

La forma de conseguir esto en nuestro contexto es mediante la intervención de un tercero: un agente de confianza que sea quien firma digitalmente el certificado y sobre el que un posible suplantador de identidad no tiene capacidad de actuación. A este agente de confianza se le denomina Autoridad Certificadora (CA) y es una empresa u organización de confianza que crea el certificado asociado a un servidor, lo firma y lo envía al servidor para que este lo use como certificado público.

Hay muchas CA, la mayoría son empresas con ánimo de lucro que, por tanto, cobran para dar estos certificados. Existen algunas alternativas gratuitas, entre ellas Let's Encrypt[6]. Esta es una CA sin ánimo de lucro que genera certificados gratuitos a cualquier persona que los solicite. Para ello debe conectarse a **https://certbot.eff.org/** y seguir las instrucciones. Esencialmente consisten en instalar un software llamado Certbot en el servidor que va a usar el certificado y ejecutarlo. Si se siguen las instrucciones de la web verá que tiene diversas posibilidades y que la más sencilla incluso modifica la configuración de Apache para habilitar el uso de HTTPS (genera el certificado, lo descarga, lo instala y configura Apache).

·▷ Ejercicio 18: Configuración de HTTPS en Apache

En la web **https://certbot.eff.org/** tiene varias opciones para proceder. Como algunos parámetros pueden depender de la configuración concreta de su servidor, debe comenzar indicando qué servidor va a configurar (Apache, Nginx, etc.) y en qué sistema (GNU/Linux, Windows, MacOS, etc.)

A continuación se muestran instrucciones para continuar:

- Cómo instalar Certbot en su sistema
- Cómo ejecutar Certbot ...
  - Para generar certificado e instalarlo
  - Solo para generar el certificado
- Los certificados emitidos por esta CA tienen un periodo de caducidad (3 meses) por lo que Certbot debe ejecutarse periódicamente. En los últimos pasos de esta web se indica cómo hacer que Certbot renueve automáticamente el certificado antes de que caduque.

Podemos optar por la generación e instalación automática. Esto incluye la modificación de los ficheros de configuración de Apache para que hagan uso de los certificados.

8.3. Redirección del tráfico HTTP a HTTPS

Aunque tenga activo el esquema HTTPS podrá seguir haciendo uso de HTTP para evitar errores como el primero mostrado en la sección 8.2. Además, las aplicaciones o páginas web podrán verse igualmente independientemente del esquema utilizado.

En la configuración actual de **void.ugr.es** puede acceder a un mismo recurso por ambas vías únicamente cambiando el puerto al que se conecta. Puede comprobarlo accediendo a estas dos

6 https://letsencrypt.org/es/

URL:

- http://void.ugr.es/tweb/hola.html
- https://void.ugr.es/tweb/hola.html

Esto es así porque en los dos host virtuales que tenemos configurados se ha establecido el mismo **DocumentRoot** (que apunta a **/var/www/html**). La diferencia estriba en cómo se transmite la página solicitada (cifrada o sin cifrar).

Podríamos cambiarlo para que cada host virtual sirva contenidos distintos pero en ese caso una de las dos URL dejaría de ser válida y podría generar problemas a los usuarios que por error pueden escribir **http://** en lugar de **https://** o viceversa. O, peor aun, si ya tenemos una aplicación web o muchas páginas con hiperenlaces de la forma **http://** deberíamos cambiarlos todos para que siga funcionando.

Sin embargo, el comportamiento de **void.ugr.es** es diferente si comparamos el resultado de hacer estas otras dos peticiones:

- http://void.ugr.es/index.html
- https://void.ugr.es/index.html

Aparentemente ha ocurrido lo mismo pero si nos fijamos en la caja de la URL del navegador veremos que en ambos casos lo que muestra es la petición HTTPS (aunque hayamos hecho la petición con HTTP). Lo que se ha hecho es configurar Apache para que redirija todo el tráfico procedente de HTTP a HTTPS. De esta forma, aunque el usuario se equivoque formando la URL el tráfico iría cifrado.

Hay varias formas de hacerlo. En el caso de **void.ugr.es** vemos que en unos casos hace la redirección y en otros no para poder ilustrar los ejemplos de este documento. Por ello vamos a optar por el uso de un módulo de Apache llamado **mod_rewrite** que permite a Apache reescribir las URL de las peticiones sobre la marcha haciendo uso de expresiones regulares y poder, de esta forma, redirigir solo cuando nos interese. Para que este método funcione hemos de activarlo:

```
a2enmod rewrite
```

En el caso de **void.ugr.es** hemos modificado el fichero **/etc/apache2/sites-available/000-default.conf** y añadido las siguientes líneas:

```
RewriteEngine on
RewriteCond %{HTTPS} off                              # Si el tráfico no es HTTPS
RewriteCond %{REQUEST_URI} !^/tweb/                   # Si la URL no comienza por /tweb/
RewriteRule ^/?(.*) https://%{SERVER_NAME}/$1 [R,L]   # Cambiar http://… por https://…
```

Sin entrar en mayor detalle, la idea es que si llega una petición que no es HTTPS y que no comienza por **/tweb/** reescriba la URL para cambiar **http://…** por **https://…**

·▷ Ejercicio 19: Redirección del tráfico HTTP a HTTPS

Replique el ejemplo anterior en su máquina para redirigir todo el tráfico que llegue por HTTP hacia el esquema HTTPS.

9. Entrega de la práctica

El profesor dará instrucciones concretas sobre lo que hay que incluir en la entrega de esta práctica así como los plazos para ello.

Es imprescindible que el estudiante haya realizado esta práctica para poder avanzar en las siguientes sesiones. Llegado a este punto deberá disponer de un servidor Apache+PHP configurado para exportar la carpeta **public_html** de los usuarios y con el esquema HTTPS activo.