Unidad 3. Implantación de aplicaciones web en servidores web

Módulo: Despliegue de aplicaciones web (2ºDAW - Semipresencial)

Profesora: Isabel Soriano

*El **resultado de aprendizaje** asociado a esta unidad es el siguiente:*
RA 2. Implanta aplicaciones web en servidores web, evaluando y aplicando criterios de configuración para su funcionamiento seguro.

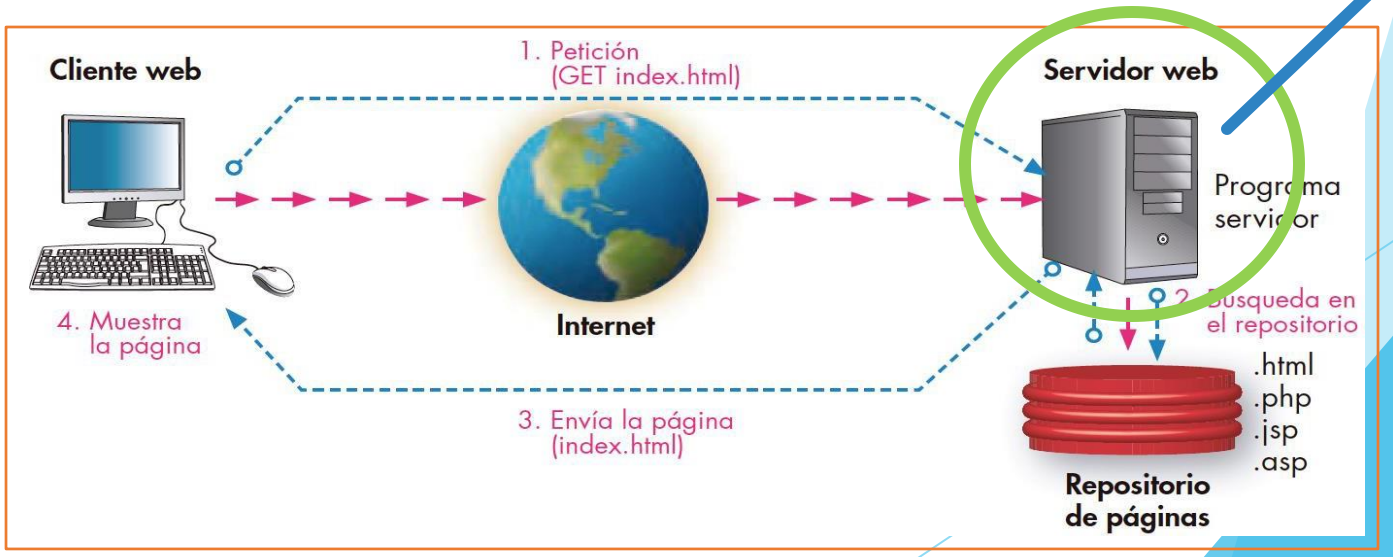
ÍNDICE

1. Introducción.
2. Protocolos web: HTTP y HTTPS.
3. Servidor web Apache.
4. Instalación de Apache.
5. Configuración general de Apache.
6. Configuración de un sitio web en Apache.
7. Prueba en el navegador (cliente).
8. Módulos en Apache.
9. Servidor seguro en Apache.
10. Control de acceso.

1 - Introducción

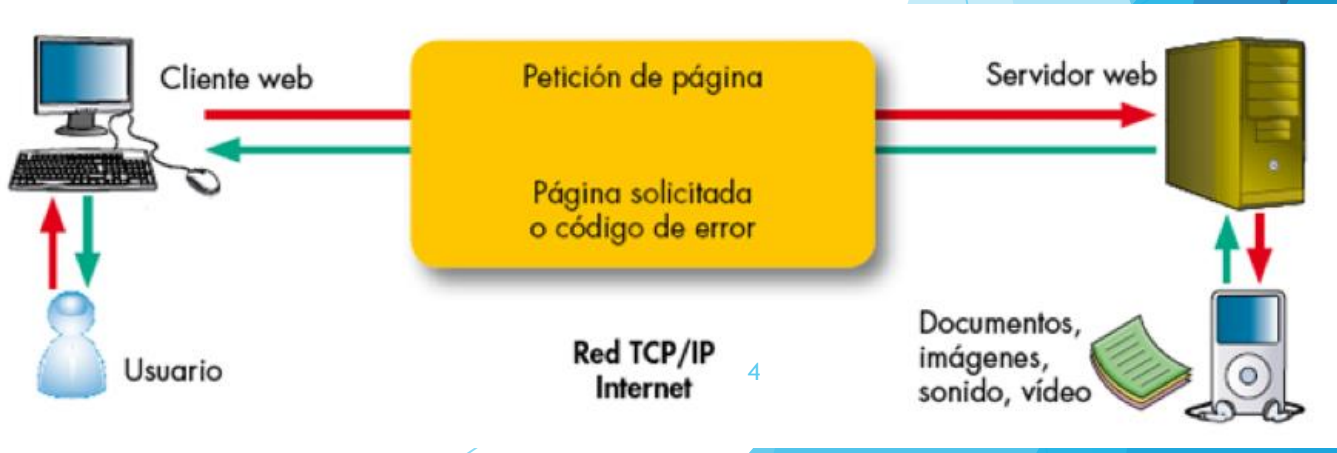
Tal y como se muestra en la imagen, para poder mostrar páginas web en un cliente, es necesario tener un servidor web funcionando correctamente. El servidor web es el componente más importante de la arquitectura web. Éste, se encarga de recibir las peticiones y dar respuesta al cliente. En esta unidad se estudiará el servidor web Apache.

Por otro lado, para la comunicación serán necesarios diferentes protocolos. Los protocolos web son HTTP y HTTPS.



2 – Protocolos web: HTTP y HTTPS

- **HTTP: Protocolo de Transferencias de HiperTexto.**
- Se usa para las transacciones de la web.
- Usa por defecto el puerto 80.
- No tiene memoria, por lo que se inventaron sistemas de almacenamiento de variables (cookies).
- El protocolo HTTP es un protocolo sin estado (no guarda información sobre las conexiones anteriores).
- Si el documento que quiere el cliente, lo encuentra, lo envía y si no envía un código de error. En cualquier caso, cierra la conexión.



2 – Protocolos web: HTTP y HTTPS

Los mensajes tienen la siguiente estructura:

- **Línea inicial con:**
  - **Para las peticiones:** la acción requerida por el servidor (método de petición) seguido de la URL del recurso y la versión de HTTP que soporta el cliente.
  - **Para respuestas:** la versión del HTTP usado seguido del código de respuesta (que indica qué ha pasado con la petición seguido de la URL del recurso) y de la frase asociada a dicho retorno.
- Las **cabeceras** del mensaje que terminan con una línea en blanco. Son metadatos.
- **Cuerpo del mensaje**. Es opcional. Su presencia depende de la línea anterior del mensaje y del tipo de recurso al que hace referencia la URL. Típicamente tiene los datos que se intercambian cliente y servidor.

2 – Protocolos web: HTTP y HTTPS

PETICIONES

MÉTODO DE PETICIÓN	Explicación
GET	Se utiliza para solicitar un recurso específico del servidor web.
POST	Se utiliza para enviar datos al servidor web. Por ejemplo, cuando introducimos datos en un formulario web.
PUT	Se utiliza para actualizar un recurso específico del servidor web.
DELETE	Se utiliza para eliminar un recurso del servidor web.

RESPUESTAS

CÓDIGO DE RESPUESTA	Explicación
1XX	Respuestas informativas. Indica que la petición ha sido recibida y se está procesando.
2XX	Respuestas correctas. Indica que la petición ha sido procesada correctamente.
3XX	Respuestas de redirección. Indica que el cliente necesita realizar más acciones para finalizar la petición.
4XX	Errores causados por el cliente. Indica que ha habido un error en el procesado de la petición a causa de que el cliente ha hecho algo mal.
5XX	Errores causados por el servidor. Indica que ha habido un error en el procesado de la petición a causa de un fallo en el servidor.

2 – Protocolos web: HTTP y HTTPS

HTTPS

- El protocolo HTTPS trabaja sobre SSL/TLS para la seguridad. Se utiliza la encriptación asimétrica de clave pública y privada para establecer la comunicación y luego una encriptación simétrica para intercambiar datos.

- Este protocolo utiliza el puerto 443.

- Mediante el protocolo HTTPS la información se cifra antes de enviarla.

- El protocolo HTTPS necesita certificados para funcionar correctamente. Estos certificados sirven para confiar en que los servidores web son legítimos. Con la instalación de Apache se crea un certificado. Desde Linux también se pueden crear certificados. Aunque estos certificados son válidos, puede ser que el navegador web de el usuario no confíe en ellos. Esto es porque los navegadores web tienen una lista de Entidades Certificadoras en las que confían. Estas entidades certificadoras verifican, autentifican y dan validez a los certificados de los servidores web.

3 - Servidor web Apache

¿Qué es un servidor?

- Un servidor es un programa que se ejecuta interrumpidamente en un ordenador.
- A veces se llama servidor al ordenador que ejecuta el programa servidor.
- Existen muchos tipos de servidores según la tarea que realizan: web, correo electrónico, transferencia de archivos, etc.
- Un servidor siempre se mantiene a la espera de peticiones por parte de un cliente.
- Si es un servidor web las peticiones las hace un navegador (browser). Cuando recibe una petición siempre la contestará de manera adecuada.
- Si ocurre un error la respuesta será un mensaje que informará del mismo.

3 - Servidor web Apache

¿Por qué hemos elegido Apache?

- El servidor web Apache es uno de los más utilizados en el mundo.

- Es open source y gratuito y multiplataforma.-

- Históricamente ha sido el servidor con mayor cuota de mercado de internet

NOTA: Está desarrollado y mantenido por una comunidad de usuarios de Apache Software Foundation.

Es un servidor web **MODULAR**. Cada módulo tiene una funcionalidad y se puede activar o desactivar.

- mod_ssl: comunicaciones seguras vía TLS.
- mod_rewrite: reescrituraa de direcciones (URL's amigables…).
- mod_auth_ldap: autentificar usuarios contra un servidor LDAP.
- mod_php: intérprete de código PHP .
- mod_python: intérprete de código Python.

3 - Servidor web Apache

Pasos para crear y configurar un sitio web en un servidor Apache:

1. Instalar el servicio web Apache.
2. Verificar el funcionamiento del servicio.
3. Configurar Apache (archivos de configuración general).
4. Configurar el sitio web:
   - 4.1 Crear el directorio del sitio web (normalmente en /var/www).
   - 4.2 Agregar contenido del sitio web (ficheros html, css, imágenes, etc.).
   - 4.3 Crear el archivo de configuración del sitio web (Virtual Host).
   - 4.4 Habilitar el sitio web.
   - 4.5 Reiniciar el servicio.
5. Probar el acceso al sitio web desde un navegador (cliente).

PASO ADICIONAL: Configurar el servidor DNS.
Como ya vimos en la Unidad 2, el servidor DNS sirve para obtener una dirección IP a partir de un nombre de dominio.

En un servidor en producción se deben configurar los servidores DNS para permitir la redirección de diferentes dominios a la misma dirección IP.

3 – Servidor web Apache

Además, serán necesarios otros elementos como:

- **Sistema de nombres de dominio (DNS):** permite que el nombre del sitio se asocie a la dirección IP del servidor Apache.
- **Cortafuegos:** debe permitir el tráfico entrante en los puertos 80 y 443.
- **Uso de certificados:** si el sitio usa HTTPS habrá que crear los certificados correspondientes.
- **Base de datos:** si el sitio es dinámico, se necesita un servicio de base de datos como MySQL o MariaDB.
- **PHP:** para sitios con contenido dinámico, Apache debe integrarse con un módulo o intérprete de PHP.

4 - Instalación de Apache

El servidor Apache lo vamos a configurar en la máquina virtual llamada 'serverWEB'.

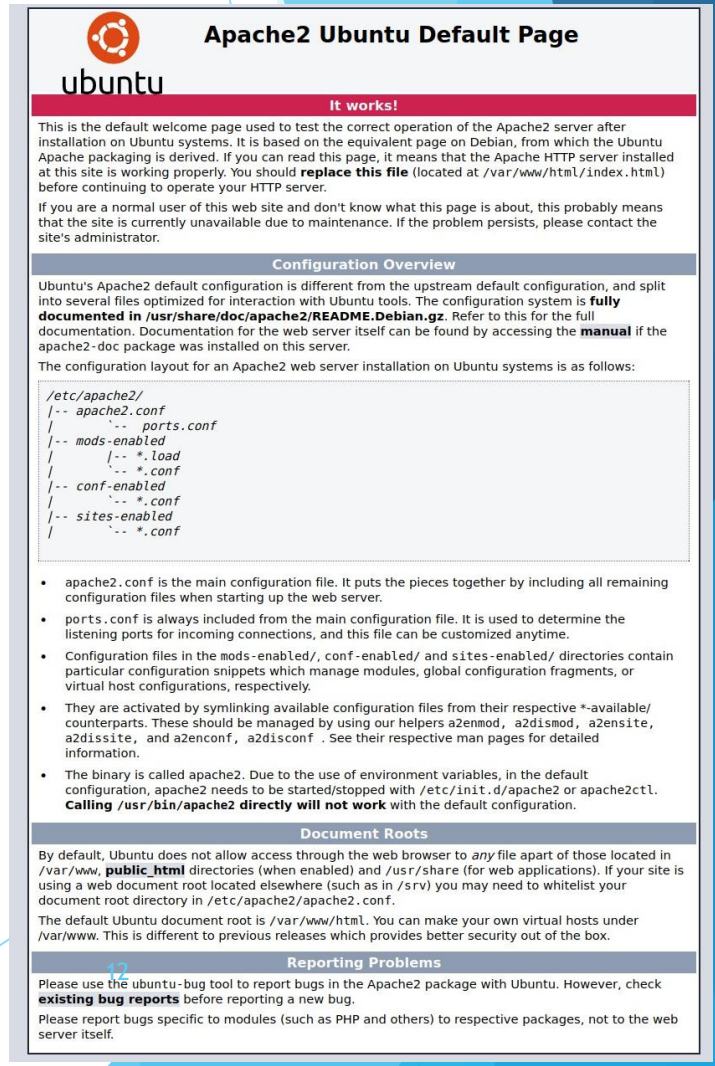
En la práctica de la unidad 1, ya instalamos la pila LAMP, donde se instalaba Apache. Por tanto, la parte de la instalación de Apache ya la tenemos hecha. Comprueba que está activo y funcionando correctamente: **# sudo systemctl status apache2**

En caso de no tener instalado Apache, ejecuta este comando:

#sudo apt install apache2

Por otro lado, enciende la máquina cliente y comprueba desde el navegador que aparece la página por defecto de Apache:

http://IP_DEL_SERVIDOR *(la IP debe ser la indicada en el esquema de red, la 10.10.5.20).*



4 - Instalación de Apache

En principio el servidor web sólo puede alojar una aplicación web.

Por defecto Apache utiliza el **directorio /var/www/ como repositorio** (directorio) de la aplicación web.

Todas las peticiones que se realizan a un servidor deben ser a un recurso en concreto → un archivo único en el repositorio.

Es habitual que en la petición no se indique un recurso, como por ejemplo en http://instagram.com. En estos casos el servidor devuelve el archivo que se llama index.

4 - Instalación de Apache

Como el directorio **/var/www/** se crea en la instalación de Apache tanto él como todo su contenido pertenece al **usuario root**.

Si queremos poder añadir, editar y borrar archivos y directorios dentro de él podemos cambiar los permisos para que todos los usuarios puedan hacer cambios.

 # sudo chmod 777 –R /var/www/

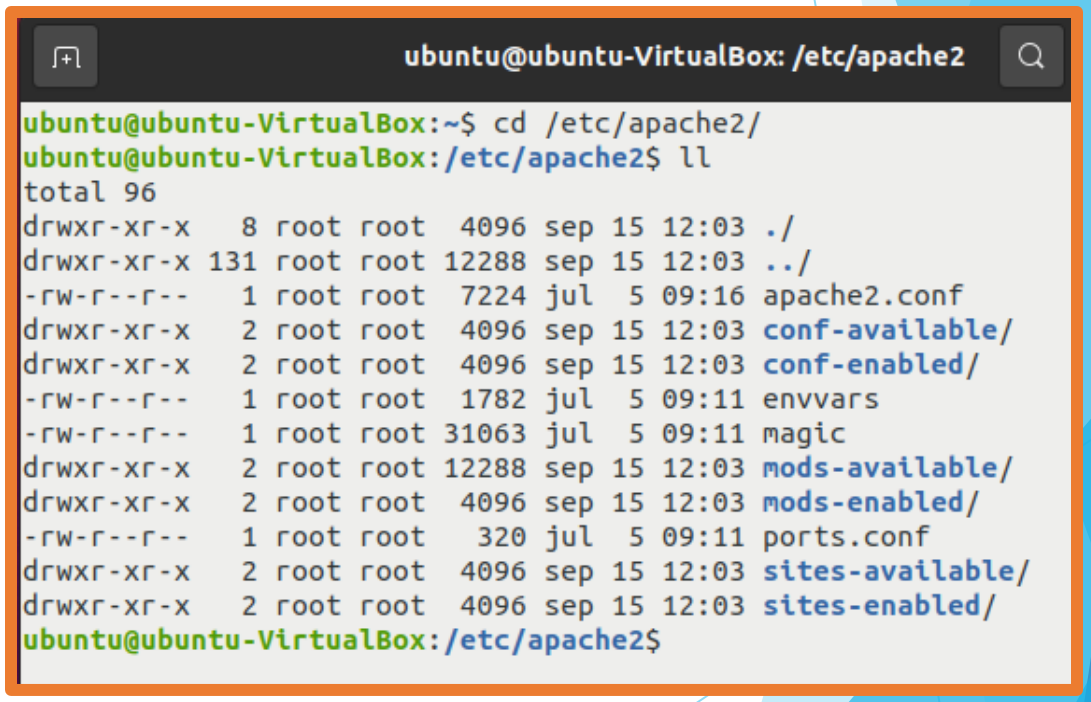
Esta configuración sólo es recomendable en entornos de desarrollo. Nunca hacer en casos reales, no se deben dar todos los permisos.

5 - Configuración general de Apache

La configuración de Apache se realiza principalmente mediante el archivo **apache2.conf** que se encuentra en el directorio: **/etc/apache2**

En este directorio se encuentran todos los archivos de configuración

Antes de hacer algún cambio es recomendable hacer una **copia de seguridad** del archivo de configuración que se vaya a modificar.



5 - Configuración general de Apache

En la siguiente tabla, se muestran los **archivos de configuración más importantes**, para configurar un servidor Apache.

¡ATENCIÓN! **Cualquier cambio en el fichero apache2.conf afecta a todo el servidor**, no solo a un sitio en particular.

Nombre del fichero	Ruta	Función
`apache2.conf`	`/etc/apache2/apache2.conf`	Archivo principal de configuración de Apache.
`ports.conf`	`/etc/apache2/ports.conf`	Define los puertos en los que escucha Apache (80, 443, etc.).
`000-default.conf`	`/etc/apache2/sites-available/000-default.conf`	Configuración del sitio web por defecto (VirtualHost).
`default-ssl.conf`	`/etc/apache2/sites-available/default-ssl.conf`	Configuración del sitio por defecto con soporte SSL/TLS.
`.htaccess`	Dentro de `/var/www/html/` o del directorio del sitio	Permite aplicar configuraciones locales (redirecciones, permisos, reescrituras).
`envvars`	`/etc/apache2/envvars`	Define variables de entorno usadas por Apache (usuario, grupo, etc.).
Archivos en `sites-available/`	`/etc/apache2/sites-available/`	Contienen configuraciones de distintos sitios virtuales.
Archivos en `sites-enabled/`	`/etc/apache2/sites-enabled/`	Enlazan los sitios activados que Apache cargará al iniciar.
Archivos en `mods-available/`	`/etc/apache2/mods-available/`	Configuración de módulos disponibles (rewrite, ssl, etc.).
Archivos en `mods-enabled/`	`/etc/apache2/mods-enabled/`	Enlazan los módulos activados que se cargan en Apache.

5 - Configuración general de Apache

El **fichero apache2.conf** es el **archivo principal** de configuración del servidor web Apache en sistemas basados en Debian/Ubuntu.

Su función es definir las directrices globales que determinan el comportamiento general del servidor, tales como:

- Los directorios donde se encuentran los sitios web y los módulos.
- Las políticas de seguridad (permisos, usuarios y grupos con los que se ejecuta el servicio).
- La carga de otros archivos de configuración (por ejemplo, incluye referencias a ports.conf, sites-enabled/, mods-enabled/).
- Parámetros de rendimiento como el número de procesos hijos, el uso de memoria o los límites de conexiones.

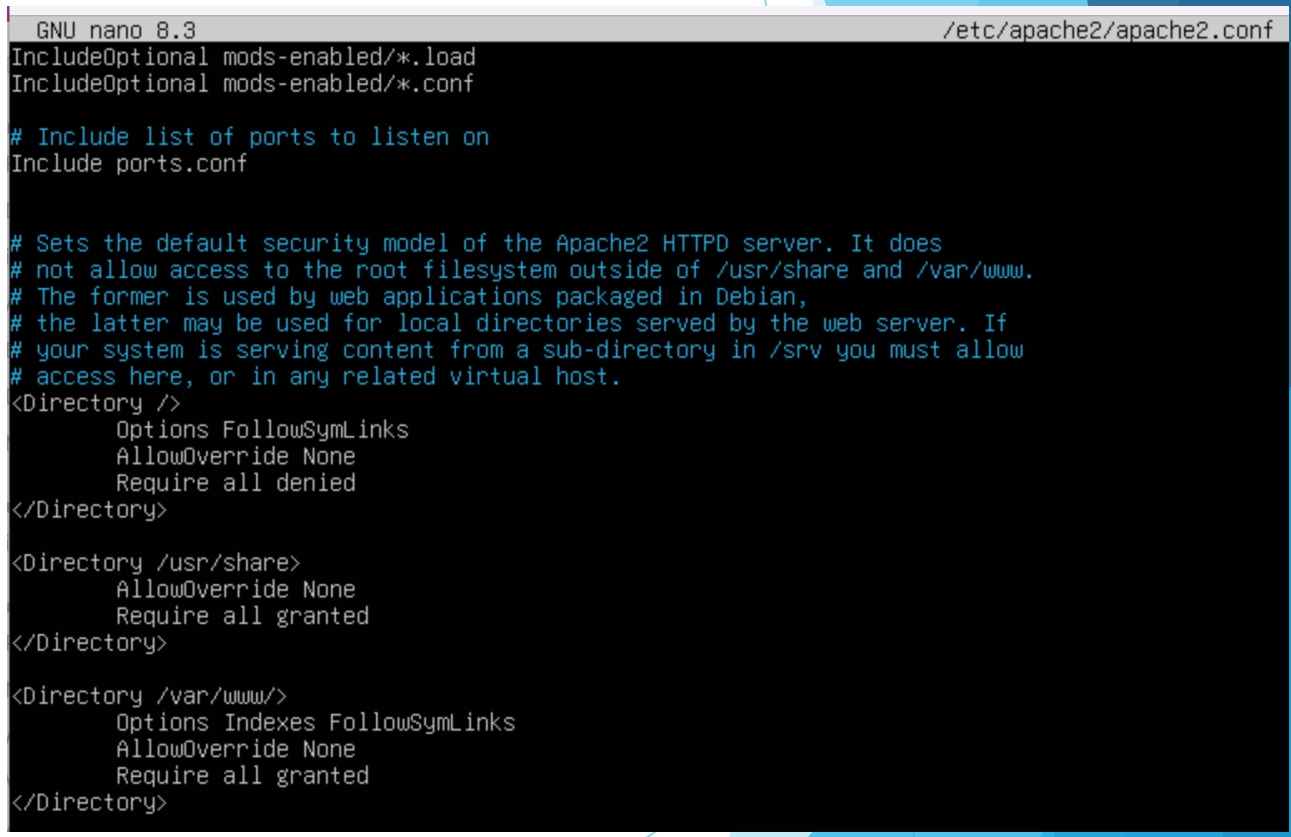
5 - Configuración general de Apache

Para cambiar la configuración es necesario abrir el archivo como administrador.

 # sudo nano /etc/apache2/apache2.conf

Como este archivo es grande, para facilitar la configuración se ha incluido la directiva *Include* para extraer partes de la configuración a otros archivos. *IncludeOptional* significa que, si no existen, no se produce error. Por ejemplo, la configuración de los puertos se hace en ports.conf.

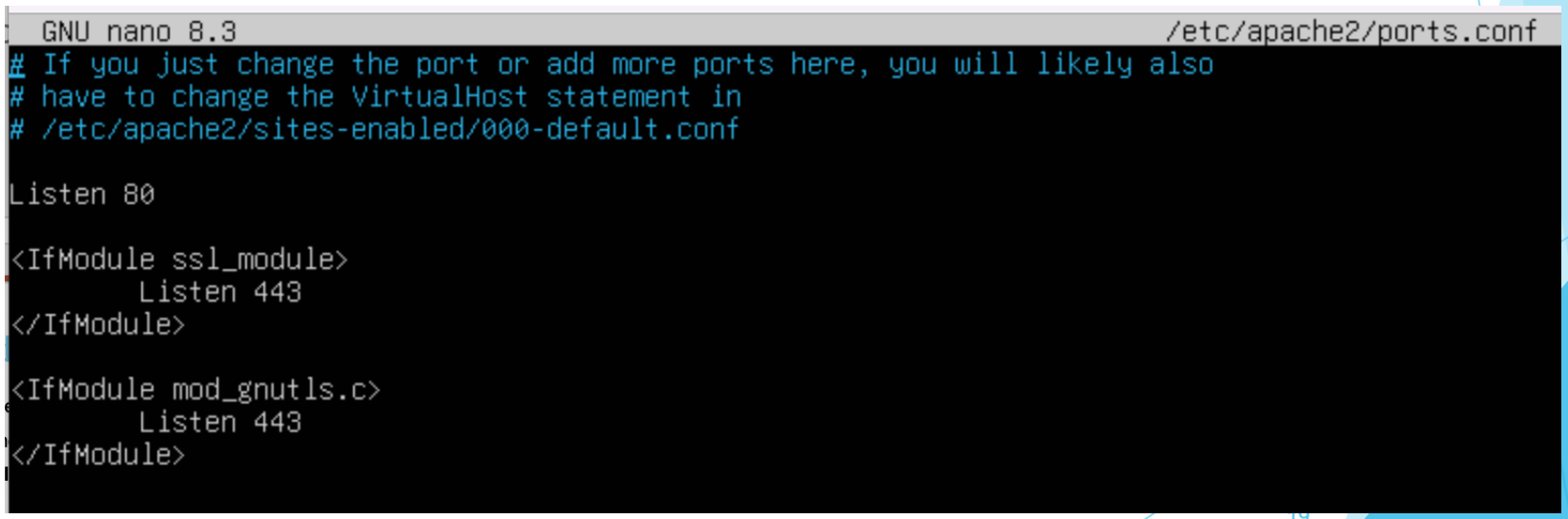
Por otro lado, tenemos los bloques 'Directory' que definen qué directorios del sistema son accesibles por Apache.



5 - Configuración general de Apache

Así la configuración de los puertos de escucha se encuentra en el archivo **ports.conf.** Podemos acceder a este archivo y modificar sus puertos o añadir nuevos. Por defecto, **HTTP tiene el puerto 80 y HTTPS el puerto 443.**

Para añadir un puerto se debe indicar la directiva ***Listen PUERTO.***



Actividades

Aules

En este tema, se van a ir proponiendo actividades durante la teoría. Al finalizar la unidad, se subirán las soluciones en Aules.

ACTIVIDAD 1 - PUERTO DE ESCUCHA EN APACHE

Realiza los siguientes pasos:

- Cambia el puerto de escucha del servidor web por el siguiente:

  2diaNacimientoMesNacimiento ejemplo: 30 de noviembre → 23011

- Comprueba mediante el navegador web que el servidor web sólo es accesible con el nuevo puerto → ejemplo: http://IP_servidorweb:23011

Para que los cambios se apliquen, debes ejecutar este comando:
#sudo systemctl reload apache2

5 - Configuración general de Apache

¡IMPORTANTE! Se puede definir el bloque 'Directory' tanto en el fichero general 'apache2.conf' como en el fichero de configuración del VirtualHost (ej: prueba.conf). Si se aplica en apache2.conf, aplicarán sus reglas a todos los sitios web del servidor.

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

- **Directory /** → Se aplica al directorio raíz del sistema (/).
- **Options FollowSymLinks** → Permite seguir enlaces simbólicos.
- **AllowOverride None** → Prohíbe el uso de archivos .htaccess aquí.
- **Require all denied** → Niega todo acceso a este nivel (seguridad máxima).

Es una medida de seguridad: impide que Apache sirva archivos fuera de /var/www o /usr/share.

```
<Directory /usr/share>
    AllowOverride None
    Require all granted
</Directory>
```

- **Directory /usr/share** → Se aplica a este directorio, donde suelen instalarse archivos web de paquetes del sistema.
- **AllowOverride None** → No permite .htaccess.
- **Require all granted** → Permite acceso de lectura a todos los usuarios (Apache puede servir contenido desde ahí).

Esto es necesario, por ejemplo, para servir documentación o interfaces web instaladas por paquetes.

5 - Configuración general de Apache

```
<Directory /var/www/>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
```

- **Directory /var/www/** → Este es el directorio raíz por defecto donde se almacenan los sitios web.
- **Options Indexes FollowSymLinks:**
  - **Indexes** → Permite listar el contenido de una carpeta si no hay index.html.
  - **FollowSymLinks** → Permite seguir enlaces simbólicos dentro del sitio.
- **AllowOverride None** → Los archivos **.htaccess** no tendrán efecto.
- **Require all granted** → Apache permite servir contenido desde aquí.

Aquí es donde colocarás tus proyectos web (por ejemplo, /var/www/prueba.com).

Para que se aplique la configuración indicada en el archivo **.htaccess**, se debe indicar dentro de **'Directory'** con la línea: **AllowOverride All**.

En el caso de indicarlo dentro del VirtualHost, sería por ejemplo lo siguiente:

```
<Directory /var/www/prueba.com>
    AllowOverride All
</Directory>
```

5 - Configuración general de Apache

FICHERO .htaccess

.htaccess significa literalmente “hypertext access”, y no es un archivo del sistema interno de Apache, sino un archivo de **configuración local** que Apache permite leer dinámicamente desde cada carpeta del sitio web.

Apache está configurado para permitir que los usuarios del hosting pongan pequeños fragmentos de configuración (acceso, redirecciones, reescrituras...) sin tener permisos sobre el servidor. Lo consigue gracias a una directiva del servidor llamada **AllowOverride All.** Por ejemplo, si alojas una aplicación web en un proveedor como Hostinger o Ionos y quieres que las URLs sean más limpias, puedes crear un .htaccess con reglas de reescritura. Todo ello, sin acceder a ficheros propios del servidor como apache2.conf.

Si se trabaja en un servidor propio o corporativo al que se tiene acceso, NO es recomendable habilitar .htaccess.
Todo lo que se define en este fichero local, se puede definir también en los ficheros de configuración apache2.conf (dentro de ‘Directory’) y en el fichero del VirtualHost.

5 - Configuración general de Apache

FICHERO .htaccess

Un archivo **.htaccess** es un fichero que **contiene reglas de configuración locales**, como, por ejemplo:

- Reescritura de URLs (RewriteEngine). URLs amigables.
- Protección con contraseña (Auth).
- Redirecciones (Redirect).
- Bloqueo por IP o dominio.
- Archivos de error personalizados.

Para que funcionen correctamente las reglas del archivo .htaccess (como las redirecciones o las URLs amigables), se debe activar previamente el módulo **mod_rewrite**, de la siguiente manera: **#sudo a2enmod rewrite**. Si no lo activas, cualquier regla de reescritura o redirección en .htaccess será ignorada.

Una vez hecho esto, debemos recargar el servicio: **#sudo systemctl reload apache2.**

5 - Configuración general de Apache

FICHERO .htaccess – Reescritura de URLs (URLs amigables)

La reescritura de URLs (URL rewriting) es una técnica que permite que las direcciones web que ve el usuario (en el navegador) sean más limpias, cortas y fáciles de entender, aunque por detrás Apache esté sirviendo archivos con nombres diferentes o con parámetros.

¿Por qué se usa la reescritura de URLs en Apache?	
❖ URLs más limpias y amigables	Mejoran la experiencia del usuario al ser fáciles de leer y recordar.
❖ Mejora del SEO (posicionamiento web)	Los buscadores prefieren URLs cortas, claras y con palabras clave.
❖ Oculta la estructura interna del servidor (archivos, carpetas, parámetros)	Aumenta la seguridad y evita mostrar la organización interna del sitio.
❖ Permite redirecciones y control avanzado de acceso	Facilita mover, reorganizar o proteger recursos sin romper enlaces.
❖ Mejora la profesionalidad y estética del sitio	Las URLs parecen de un sistema más moderno y bien diseñado.

5 - Configuración general de Apache

FICHERO .htaccess – Reescritura de URLs (URLs amigables)

EJEMPLO

Apache recibe una petición como: **https://www.prueba.com/productos/ordenadores**

y **la traduce internamente** a: **https://www.prueba.com/productos.php?categoria=ordenadores**

```
/var/www/prueba.com/
├── .htaccess
├── index.php
└── productos.php
```

Estructura del sitio web

Esta "traducción interna" la hace el módulo **mod_rewrite** de Apache (el cual debe estar activado), y las reglas se guardan en el archivo **.htaccess**. Para que tenga efecto el archivo .htaccess, en el VirtualHost o en la configuración de Apache (apache2.conf), se debe haber añadido la directiva 'AllowOverride All'.

En el archivo .htaccess (/var/www/prueba.com/.htaccess) se debe indicar lo siguiente:

1. Crea una URL amigable como la indicada anteriormente.
2. Reescribe internamente. Sirve index.php al escribir /inicio

```
# Activar el motor de reescritura de Apache
RewriteEngine On

# 1️⃣ Reescribir URLs limpias para productos
# Si el usuario entra en /productos/ordenadores
# Apache servirá internamente productos.php?categoria=ordenadores
RewriteRule ^productos/([a-zA-Z0-9_-]+)/?$ productos.php?categoria=$1 [L]

# 2️⃣ Redirigir /inicio a index.php
RewriteRule ^inicio/?$ index.php [L]
```

5 - Configuración general de Apache

FICHERO .htaccess – Redirecciones

Una redirección le indica al navegador que la página que intenta visitar se ha movido o debe consultarse en otra dirección. Apache, al recibir la petición, responde con un código HTTP 3xx, diciéndole al cliente (navegador o motor de búsqueda) que vaya a otra URL.

Tipos de redirecciones más comunes		
Código	**Significado**	**Uso habitual**
301 (permanente)	La página se ha movido definitivamente.	SEO, cambio de dominio o estructura.
302 (temporal)	La página está temporalmente en otro sitio.	Pruebas, mantenimiento o migraciones.
307 (temporal)	Similar al 302, pero más estricto con el método HTTP.	Casos avanzados (POST).

5 - Configuración general de Apache

FICHERO .htaccess – Redirecciones

Existen 2 formas de definir una redirección en .htaccess:

(*en el ejemplo, lo que hace es que cualquier acceso a /antigua redirige a /nueva. 301 indica que la redirección es permanente*).

1. **Usar directiva 'Redirect'** (debe estar habilitado el módulo **mod_alias**).

Ejemplo:

```
Redirect 301 /antigua https://www.prueba.com/nueva.
```

2. **Con 'RewriteRule'** (más flexible, usa mod_rewrite). Permite usar expresiones regulares y condiciones (RewriteCond). Es la más usada en sitios dinámicos.

Ejemplo:

```
RewriteEngine On
RewriteRule ^antigua/?$ https://www.prueba.com/nueva [L,R=301]
```

5 - Configuración general de Apache

FICHERO .htaccess – Redirecciones

También se pueden hacer otro tipo de redirecciones como las que se indican en estos ejemplos:

 # Redirección HTTP → HTTPS

RewriteCond %{HTTPS} off

RewriteRule ^(.*)$ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]

 # Redirección sin www → con www

RewriteCond %{HTTP_HOST} !^www\. [NC]

RewriteRule ^(.*)$ https://www.%{HTTP_HOST}/$1 [L,R=301]

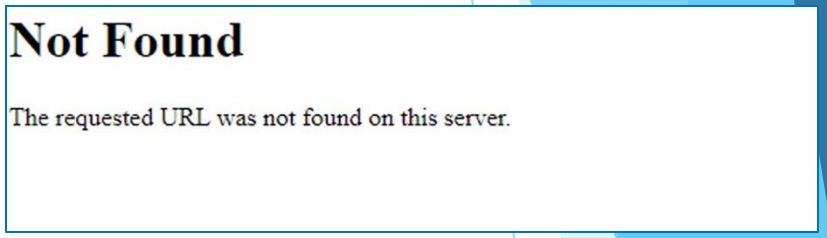
 # Redirección de página antigua

RewriteRule ^productos/antiguo-modelo/?$ https://www.prueba.com/productos/nuevo-modelo [L,R=301]

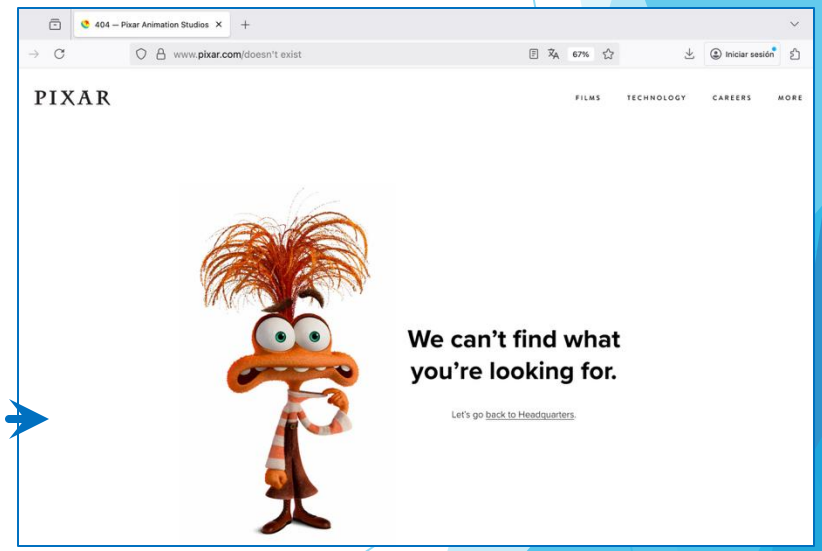
5 - Configuración general de Apache

FICHERO .htaccess – Archivos de error personalizados

Cuando ocurre un error HTTP (por ejemplo 404 o 403), Apache devuelve una página muy básica como esta:



Esto no queda profesional, ni ayuda al usuario. Para mejorar la experiencia, puedes crear páginas personalizadas (por ejemplo, error404.html, error403.html, etc.) y decirle a Apache que las muestre automáticamente cuando se produzcan esos errores. Por ejemplo, en Pixar puedes encontrar esta página de error personalizada:



5 - Configuración general de Apache

FICHERO .htaccess – Archivos de error personalizados

Códigos de error HTTP más comunes:

Código	Error	Cuándo ocurre
400	Bad Request	Petición mal formada o inválida
401	Unauthorized	Se requiere autenticación
403	Forbidden	Acceso denegado (sin permisos)
404	Not Found	El recurso no existe
500	Internal Server Error	Error interno del servidor
503	Service Unavailable	Servidor temporalmente no disponible

5 - Configuración general de Apache

FICHERO .htaccess – Archivos de error personalizados

¿Cómo se configura en .htaccess?

La sintaxis general es: **ErrorDocument <código_error> <ruta_del_archivo_o_mensaje>**

Según el tipo de error, puedo crear un archivo personalizado diferente.

Se debe indicar la ruta relativa desde la raíz del sitio web, que empieza con /.

Ejemplo:

ErrorDocument 404 /errores/error404.html

ErrorDocument 403 /errores//error403.html

ErrorDocument 500 /errores/error500.html

```
/var/www/prueba.com/
├── .htaccess
├── index.php
├── productos.php
└── errores/
    ├── error404.html
    ├── error403.html
    └── error500.html
```

Estructura del sitio web

5 - Configuración general de Apache

Finalmente, y de manera general, si se realiza alguna modificación de la configuración de Apache se debe reiniciar el servidor para que los cambios sean efectivos.

```
# sudo systemctl reload apache2
# sudo systemctl status apache2
```

Si se hacen cambios en los archivos del repositorio /var/www/ no hace falta reiniciar el servidor ya que no se cambian archivos de configuración.

5 - Configuración general de Apache

Detrás de cada aplicación web debe existir un servidor web que responda a las peticiones de los clientes y ofrezca los recursos pertinentes.

Hay aplicaciones web tan grandes que necesitan un único servidor web para ellas, incluido es posible que necesiten más de un servidor web.

Pero otras aplicaciones son más pequeñas y no aprovechan todo el potencial del servidor web. **¿Podríamos configurar el servidor web para tener más de una aplicación web dentro de él?**

5 - Configuración general de Apache

Como se ha visto anteriormente un servidor web solo tiene un repositorio (directorio) donde buscar los recursos que quieren los clientes.

Si se ponen los archivos de dos aplicaciones web diferentes al mismo repositorio ¿cómo podría diferenciar el servidor web a qué aplicación web pertenece cada archivo?

¡No se puede! → por ejemplo: archivos index

¿Y qué hacemos en situaciones como las siguientes?

Configurar el servidor web para tener diferentes páginas web con su dominio correspondiente dentro del servidor → HOSTS VIRTUALES.

6 - Configuración de un sitio web en Apache.

La configuración de un sitio web consta de estos **pasos**:

1. Crear el directorio del sitio web.
2. Agregar contenido del sitio web.
3. Crear el archivo de configuración del sitio web (VirtualHost).
4. Habilitar el sitio web.
5. Reiniciar el servicio.

¡RECUERDA!
PASO ADICIONAL: Configurar el servidor DNS.
Como ya vimos en la Unidad 2, el servidor DNS sirve para obtener una dirección IP a partir de un nombre de dominio.

En un servidor en producción se deben configurar los servidores DNS para permitir la redirección de diferentes dominios a la misma dirección IP.

6 - Configuración de un sitio web en Apache.

PASO 1. CREAR EL DIRECTORIO DEL SITIO WEB

Una vez instalado Apache y realizadas las configuraciones generales necesarias, debemos crear y configurar nuestros sitios web en el servidor. Para ello, el primer paso, es crear el directorio del sitio web. Por convención, los sitios se guardan en /var/www/.

1. **Crea una carpeta para tu sitio web** (*recomendado que sea el nombre de tu página*):

#sudo mkdir –p /var/www/mipagina

2. Asignar los permisos adecuados:

#sudo chown -R $USER:$USER /var/www/mipagina

#sudo chmod -R 755 /var/www

EJEMPLO

```
sudo mkdir -p /var/www/prueba.com
sudo chown -R www-data:www-data /var/www/prueba.com
sudo chmod -R 755 /var/www
```

6 - Configuración de un sitio web en Apache.

PASO 1. CREAR EL DIRECTORIO DEL SITIO WEB

NOTA:

En **$USER:$USER** debemos poner **www-data:www-data**.

www-data *es el usuario/grupo con el que Apache accede a los archivos del servidor web. Si tienes problemas de permisos al cargar una página, asegúrate de que los archivos sean accesibles por www-data. Se usa este usuario por varias razones: por un lado, evita que Apache tenga permisos de superusuario (root), reduciendo riesgos si hay una vulnerabilidad y por otro lado, los archivos web pertenecen a www-data, impidiendo que otros usuarios del sistema los modifiquen sin permisos adecuados.*

El uso de -R es para que los permisos se apliquen recursivamente a todos los archivos y subdirectorios.

6 - Configuración de un sitio web en Apache.

PASO 2. AGREGAR CONTENIDO DEL SITIO WEB

Dentro de la carpeta que hemos creado en el paso 1, debemos guardar todos los archivos de nuestra página web. Para hacer una primera prueba, vamos a crear un fichero index.html.

1. Crear el fichero dentro de la carpeta:

#sudo nano /var/www/mipagina/index.html

2. Añadir contenido al fichero index.html:

EJEMPLO

6 - Configuración de un sitio web en Apache.

PASO 3. CREAR EL ARCHIVO DE CONFIGURACIÓN DEL SITIO WEB (VIRTUALHOST).

Como Apache ya tiene creado un archivo de configuración de un host virtual: **000-default.conf.** Lo más conveniente es duplicar el archivo, cambiarle el nombre (*el que se indica es un ejemplo*) y luego editarlo con la nueva configuración. Dentro del directorio **sites-available**:

EJEMPLO

```
$ sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/prueba.conf
$ sudo nano /etc/apache2/sites-available/prueba.conf
```

En el archivo de configuración por defecto del host virtual de Apache podemos ver las directivas indispensables para un host virtual.

6 - Configuración de un sitio web en Apache.

PASO 3. CREAR EL ARCHIVO DE CONFIGURACIÓN DEL SITIO WEB (VIRTUALHOST).

Un **VirtualHost** en Apache es una configuración que permite que un mismo servidor Apache aloje múltiples sitios web en una sola máquina. Apache permite configurar **dos tipos principales de VirtualHost**: **Basado en Nombre** (permite que varios dominios compartan la misma dirección IP) y **basado en IP** (se usa cuando cada sitio tiene una IP diferente). Lo vamos a hacer basado en nombre.

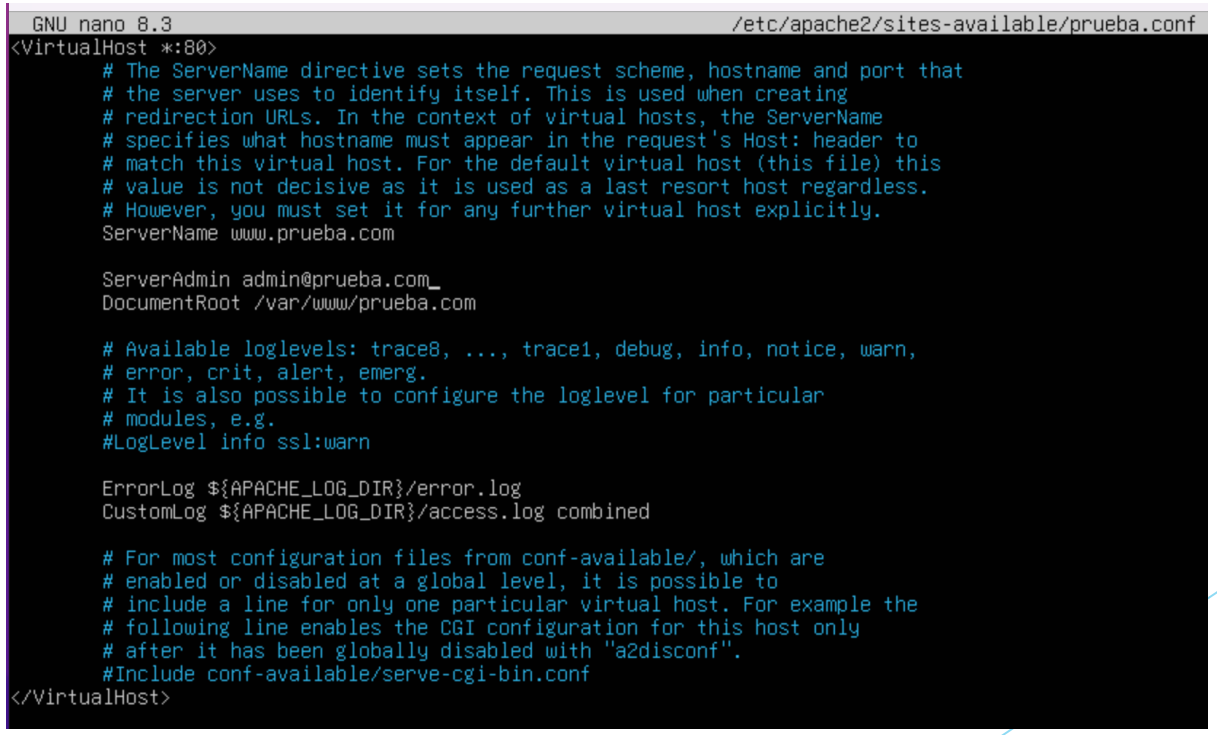
- Cuando se configuran hosts virtuales se debe hacer un archivo de configuración para cada uno de ellos.
- Toda la configuración que no se incluya en el archivo del host virtual se heredará del archivo principal de configuración de Apache: apache2.conf
- Así, si en un servidor web todos los hosts virtuales deben tener la misma directiva se configurará en el archivo principal.
- Por el contrario, si una configuración solo tiene que afectar a un host virtual, ésta se incluirá en el archivo de este host virtual.

6 - Configuración de un sitio web en Apache.

PASO 3. CREAR EL ARCHIVO DE CONFIGURACIÓN DEL SITIO WEB (VIRTUALHOST).

En este archivo debemos crear el host virtual. En el ejemplo se llama 'prueba.conf'. Debemos modificar las siguientes directivas: **ServerName** (dominio principal del VirtualHost), **ServerAdmin** y **DocumentRoot.** Además, se puede añadir la directiva **ServerAlias** si quieres tener dominios adicionales que sirvan al mismo sitio.

EJEMPLO



6 - Configuración de un sitio web en Apache.

PASO 3. CREAR EL ARCHIVO DE CONFIGURACIÓN DEL SITIO WEB (VIRTUALHOST).

Dentro de la directiva **<VirtualHost *:80>** se indica que el virtualhost usará cualquier IP disponible en el servidor (al poner *). Si tuviéramos varias interfaces y quisiéramos especificar solo una de ellas, deberíamos poner la IP en el lugar del asterisco. Por otro lado, se indica que escuchará por el puerto 80. Por tanto, deberemos acceder a través del protocolo HTTP. El puerto se puede modificar si es necesario.

EJEMPLO

```
 GNU nano 8.3                                  /etc/apache2/sites-available/prueba.conf
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        ServerName www.prueba.com

        ServerAdmin admin@prueba.com_
        DocumentRoot /var/www/prueba.com

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
</VirtualHost>
```

6 - Configuración de un sitio web en Apache.

PASO 4. HABILITAR EL SITIO WEB.

Una vez creado el sitio web (VirtualHost), debes activarlo:

#sudo a2ensite mipagina.conf

Con la ejecución de este comando se creará un enlace a la configuración en el directorio **sites-enabled**.

Por otro lado, se recomienda desactivar el sitio web por defecto de Apache:

#sudo a2dissite 000-default.conf

PASO 5. REINICIAR EL SERVICIO.

Como último paso, siempre debes recargar el servicio:

#sudo systemctl reload apache2

7 - Prueba en el navegador (cliente)

Una vez configurado el sitio web en el servidor Apache, debemos abrir la máquina cliente y desde el navegador acceder al sitio web:

http://dominio_o_IP/

EJEMPLO. Acceso con la dirección IP (10.10.5.20) del servidor web.

Recuerda que el **firewall** debe permitir los puertos que hayamos configurado para el servidor web (80, 443, etc.)

sudo ufw enable
sudo ufw allow 80
sudo ufw status

¿Se podría acceder poniendo http://www.prueba.com?

Actividades

Aules

ACTIVIDAD 2– CONFIGURACIÓN DEL SERVIDOR DNS

En la Unidad 2, configuramos un servidor DNS primario en la máquina virtual ‘server’. Realiza los pasos necesarios para configurar el servidor DNS y que desde el navegador podamos acceder al sitio web del servidor web usando la URL **http://www.prueba.com**.

Actividades

Aules

¡ATENCIÓN! Ten en cuenta que debes hacer modificaciones tanto en el servidor web, como en el servidor DNS y en el cliente. Recuerda que al tener nuevos dominios tienes que añadirlos correctamente.

ACTIVIDAD 3 – CONFIGURACIÓN DE UN SITIO WEB

- Crea un host virtual que responda al dominio **empresa.com** y al dominio **localhost.empresa.com**.
- Usa el puerto por defecto y responde a todas las interfaces.
- El directorio de este host virtual debe estar en **/var/www/empresa**.
- Cuando se acceda al directorio del servidor mediante el navegador web, el primer recurso que debe devolver debe ser un archivo llamado **empresa.html**.
- El archivo **empresa.html** debe tener la estructura de un documento HTML5 y en el body debe poner "*Bienvenido a la web de Empresa*" y debajo tu nombre completo.
- Comprueba que puedes acceder al servidor con el navegador web, tanto usando la IP del servidor como usando el dominio **http://dominio/** y **http://IP/.**
- Configura el **archivo .htaccess** para que cuando se solicite un recurso no existente (por *ejemplo: www.localhost.empresa/contacto*), responda con un mensaje de error personalizado, el cual se encuentra en el archivo **error.html**. Este archivo debe estar creado previamente dentro del directorio **/var/www/empresa/errores.** Al cargar el error, debe decir '*El recurso solicitado no se encuentra en el servi*dor' y debajo tu nombre completo.

8 - Módulos en Apache.

Los módulos dan al servidor la capacidad de activar y desactivar funcionalidades de manera sencilla. Mediante los módulos también se da una estructura a la configuración del servidor.

Apache tiene preinstalados un conjunto de módulos en el directorio **mods-available.**

De todos estos módulos hay activados algunos por defecto en el directorio **mods- enabled**.

```
ubuntu@ubuntu-VirtualBox:/etc/apache2$ ll mods-enabled/
total 8
drwxr-xr-x 2 root root 4096 sep 21 15:57 ./
drwxr-xr-x 8 root root 4096 sep 21 17:46 ../
lrwxrwxrwx 1 root root   36 sep 15 12:03 access_compat.load -> ../mods-available/access_compat.load
lrwxrwxrwx 1 root root   28 sep 15 12:03 alias.conf -> ../mods-available/alias.conf
lrwxrwxrwx 1 root root   28 sep 15 12:03 alias.load -> ../mods-available/alias.load
lrwxrwxrwx 1 root root   33 sep 15 12:03 auth_basic.load -> ../mods-available/auth_basic.load
lrwxrwxrwx 1 root root   33 sep 15 12:03 authn_core.load -> ../mods-available/authn_core.load
lrwxrwxrwx 1 root root   33 sep 15 12:03 authn_file.load -> ../mods-available/authn_file.load
lrwxrwxrwx 1 root root   33 sep 15 12:03 authz_core.load -> ../mods-available/authz_core.load
lrwxrwxrwx 1 root root   33 sep 15 12:03 authz_host.load -> ../mods-available/authz_host.load
lrwxrwxrwx 1 root root   33 sep 15 12:03 authz_user.load -> ../mods-available/authz_user.load
lrwxrwxrwx 1 root root   32 sep 15 12:03 autoindex.conf -> ../mods-available/autoindex.conf
lrwxrwxrwx 1 root root   32 sep 15 12:03 autoindex.load -> ../mods-available/autoindex.load
lrwxrwxrwx 1 root root   30 sep 15 12:03 deflate.conf -> ../mods-available/deflate.conf
lrwxrwxrwx 1 root root   30 sep 15 12:03 deflate.load -> ../mods-available/deflate.load
lrwxrwxrwx 1 root root   26 sep 15 12:03 dir.conf -> ../mods-available/dir.conf
lrwxrwxrwx 1 root root   26 sep 15 12:03 dir.load -> ../mods-available/dir.load
lrwxrwxrwx 1 root root   26 sep 15 12:03 env.load -> ../mods-available/env.load
lrwxrwxrwx 1 root root   29 sep 15 12:03 filter.load -> ../mods-available/filter.load
lrwxrwxrwx 1 root root   27 sep 15 12:03 mime.conf -> ../mods-available/mime.conf
lrwxrwxrwx 1 root root   27 sep 15 12:03 mime.load -> ../mods-available/mime.load
lrwxrwxrwx 1 root root   32 sep 15 12:03 mpm_event.conf -> ../mods-available/mpm_event.conf
lrwxrwxrwx 1 root root   32 sep 15 12:03 mpm_event.load -> ../mods-available/mpm_event.load
lrwxrwxrwx 1 root root   34 sep 15 12:03 negotiation.conf -> ../mods-available/negotiation.conf
lrwxrwxrwx 1 root root   34 sep 15 12:03 negotiation.load -> ../mods-available/negotiation.load
lrwxrwxrwx 1 root root   33 sep 15 12:03 reqtimeout.conf -> ../mods-available/reqtimeout.conf
lrwxrwxrwx 1 root root   33 sep 15 12:03 reqtimeout.load -> ../mods-available/reqtimeout.load
lrwxrwxrwx 1 root root   31 sep 15 12:03 setenvif.conf -> ../mods-available/setenvif.conf
lrwxrwxrwx 1 root root   31 sep 15 12:03 setenvif.load -> ../mods-available/setenvif.load
lrwxrwxrwx 1 root root   29 sep 15 12:03 status.conf -> ../mods-available/status.conf
lrwxrwxrwx 1 root root   29 sep 15 12:03 status.load -> ../mods-available/status.load
```

8 – Módulos en Apache.

Si el módulo no está instalado primero se debe instalar con el comando:

 # sudo apt-get install NOMBRE_DEL_MÓDULO

Para activar un módulo ya instalado se debe utilizar el siguiente mando:

 # sudo a2enmod NOMBRE_DEL_MÓDULO

Con la ejecución de este comando se creará un enlace a la configuración al directorio mods-enabled.

Para desactivar un módulo se debe utilizar el siguiente comando:

 # sudo a2dismod NOMBRE_DEL_MÓDULO

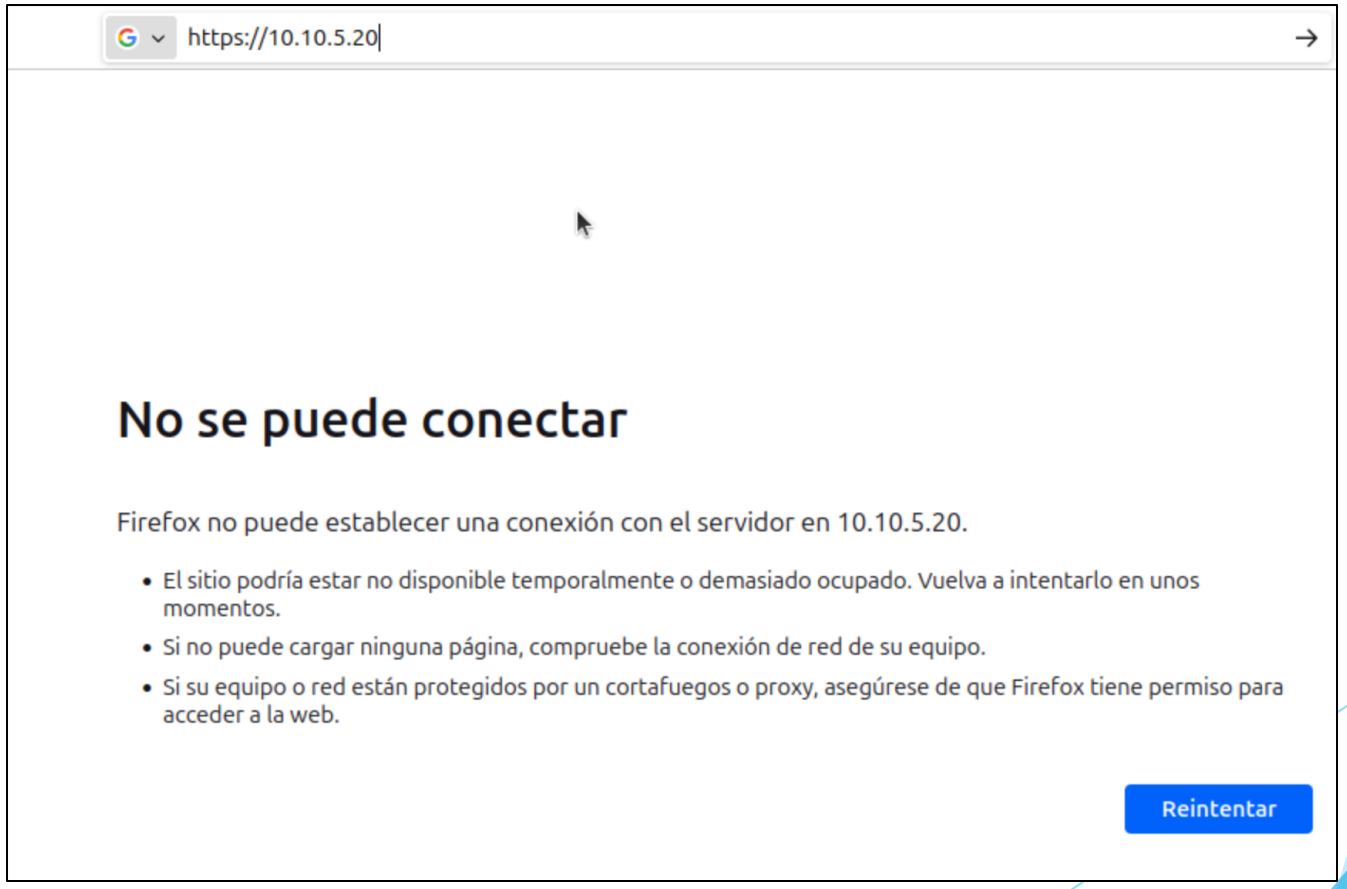
Para desinstalar un módulo primero se debe desactivar y luego se debe utilizar el siguiente comando:

 # sudo apt-get remove NOMBRE_DEL_MÓDULO

Con la ejecución de este comando se creará un enlace a la configuración al directorio mods-disabled.

9 - Servidor seguro en Apache

Si probamos a acceder desde el navegador a nuestro sitio web de manera segura, usando HTTPS, veremos que no es posible: **https://dominio_o_IP/**



9 - Servidor seguro en Apache

Para que funcione, deberemos realizar algunas modificaciones en nuestro servidor web Apache. SSL permite la encriptación.

Pasos:

1. Instalar OpenSSL: **#sudo apt install openssl –y**
2. Crear el certificado y la clave del servidor (*en NOMBRE se recomienda poner algo relativo a tu sitio web*):

#sudo openssl req –x509 –nodes –days 365 –newkey rsa:2048 –keyout /etc/ssl/private/NOMBRE.key -out /etc/ssl/certs/NOMRE.crt

Debes añadir información durante la creación de los certificados (al menos el 'Common Name').

3. Configurar el VirtualHost para SSL *(en la página siguiente).*
4. Habilitar SSL: **# sudo a2enmod ssl**
5. Reiniciar el servicio web Apache: **#sudo systemctl restart apache2.**

9 - Servidor seguro en Apache

3. Configurar el VirtualHost para SSL

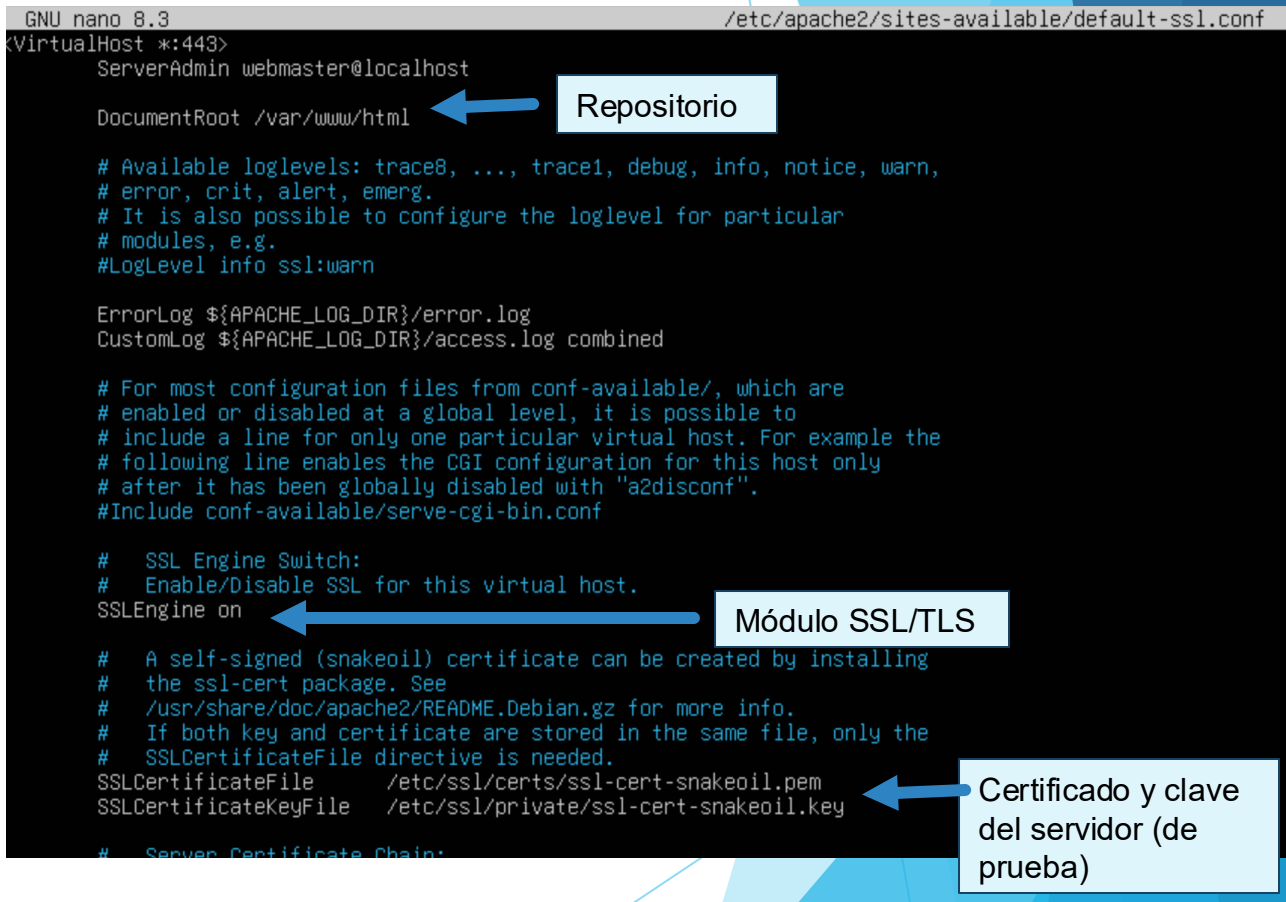
Apache ya tiene configurado un host virtual seguro. Su configuración se encuentra en el archivo: **/etc/apache2/sites-available/default-ssl.conf** como se muestra en la imagen.

Podemos crear el fichero de configuración de nuestro VirtualHost a partir del que ya existe por defecto:

#sudo cp /etc/apache2/sites-available/default-ssl.conf /etc/apache2/sites-available/NOMBRE.conf

(en NOMBRE, siguiendo el ejemplo de la teoría, podríamos poner 'pruebassl.conf').

Dentro de **'NOMBRE.conf'** debemos crear el host virtual al que accederemos usando HTTPS (puerto 443). En repositorio debemos indicar donde está nuestro sitio web. En certificado y clave, se debe modificar indicando los generados y no los de prueba.



9 - Servidor seguro en Apache

3. Configurar el VirtualHost para SSL

Una vez creado el sitio web (VirtualHost), debes activarlo:

#sudo a2ensite NOMBRE.conf

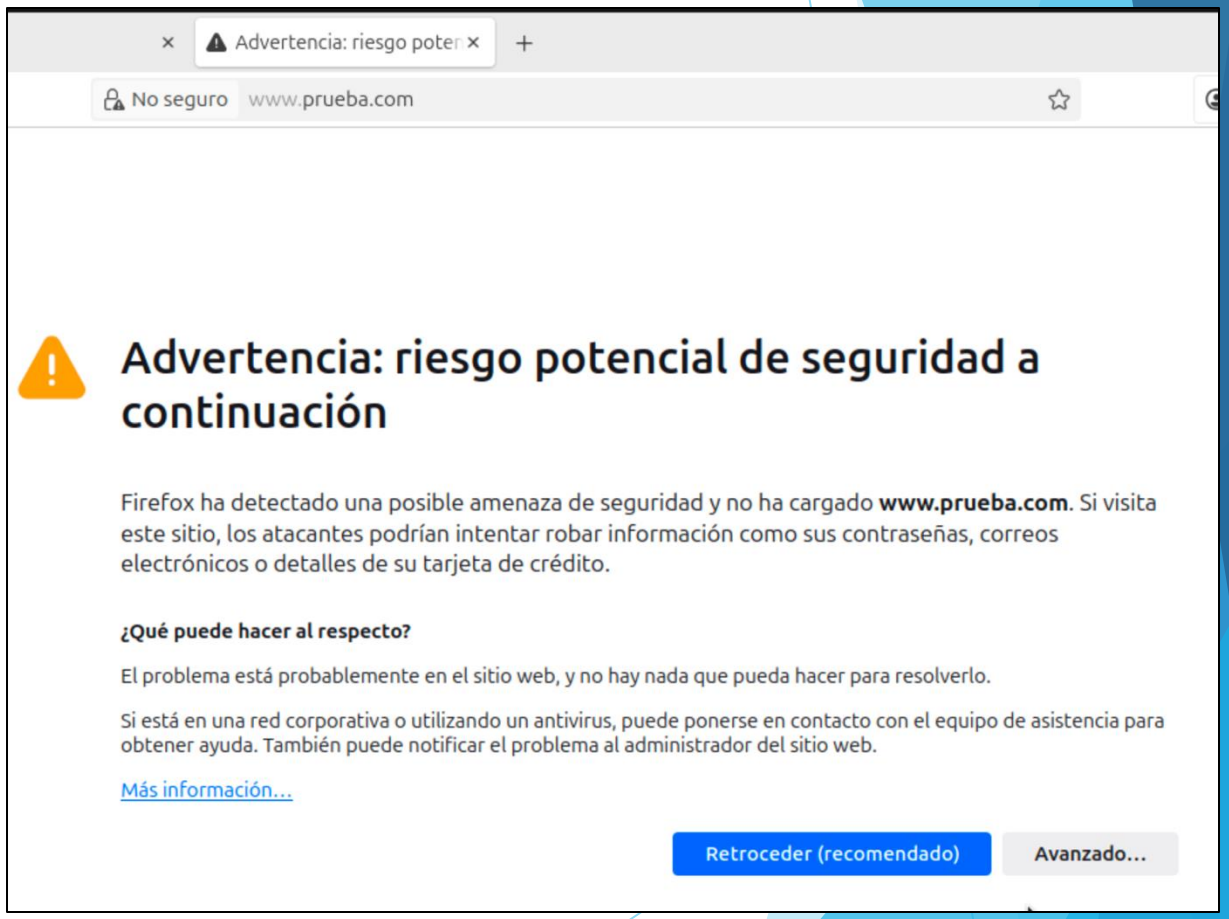
Con la ejecución de este comando se creará un enlace a la configuración en el directorio **sites-enabled**.

Por otro lado, se recomienda desactivar el sitio web por defecto de Apache:

#sudo a2dissite default-ssl.conf

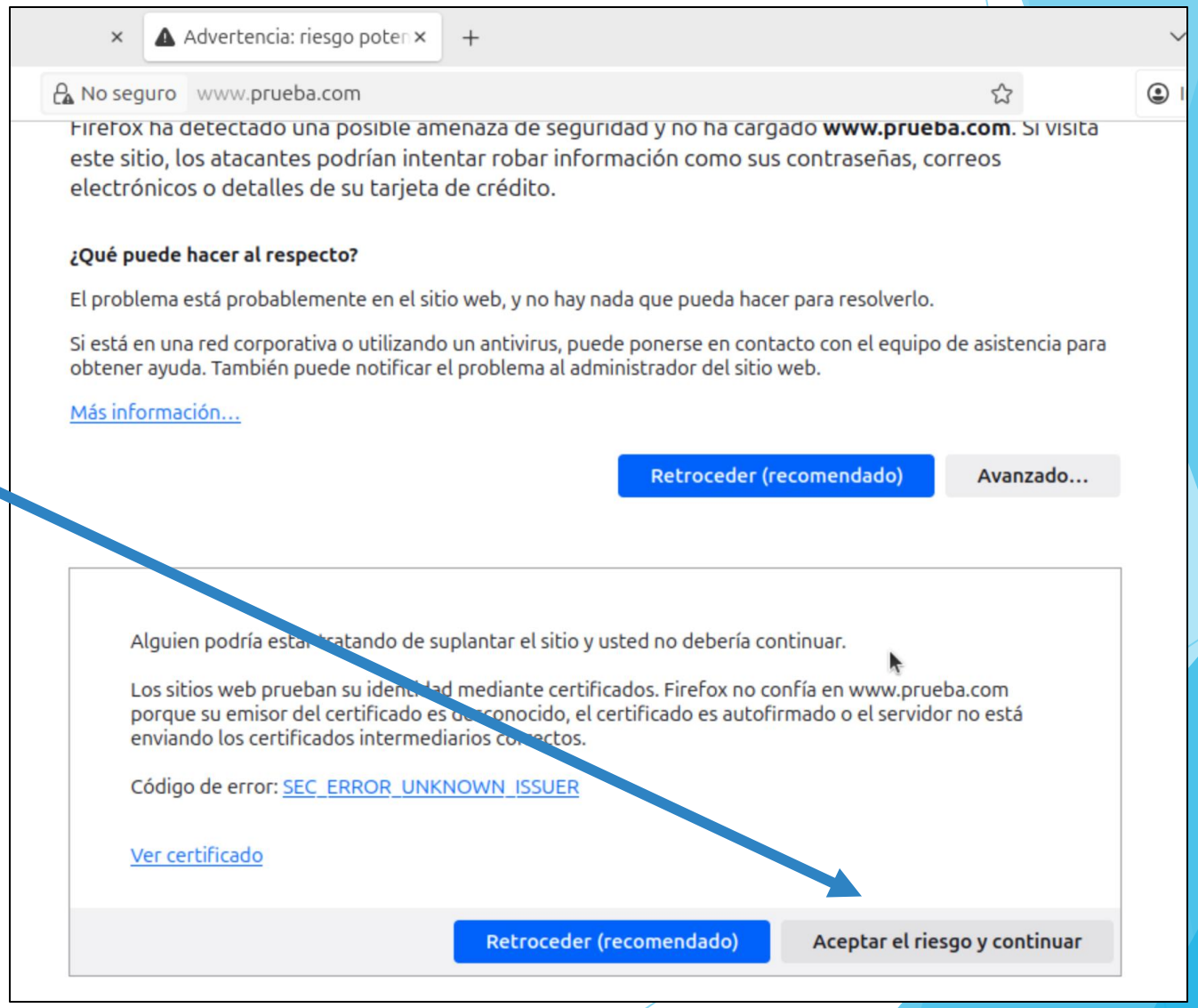
9 - Servidor seguro en Apache

Si intentamos acceder a nuestro sitio web usando el protocolo HTTPS, veremos que cuando se accede a una aplicación web con un certificado no firmado por una Autoridad Certificadora el navegador muestra una página como la siguiente:



9 - Servidor seguro en Apache

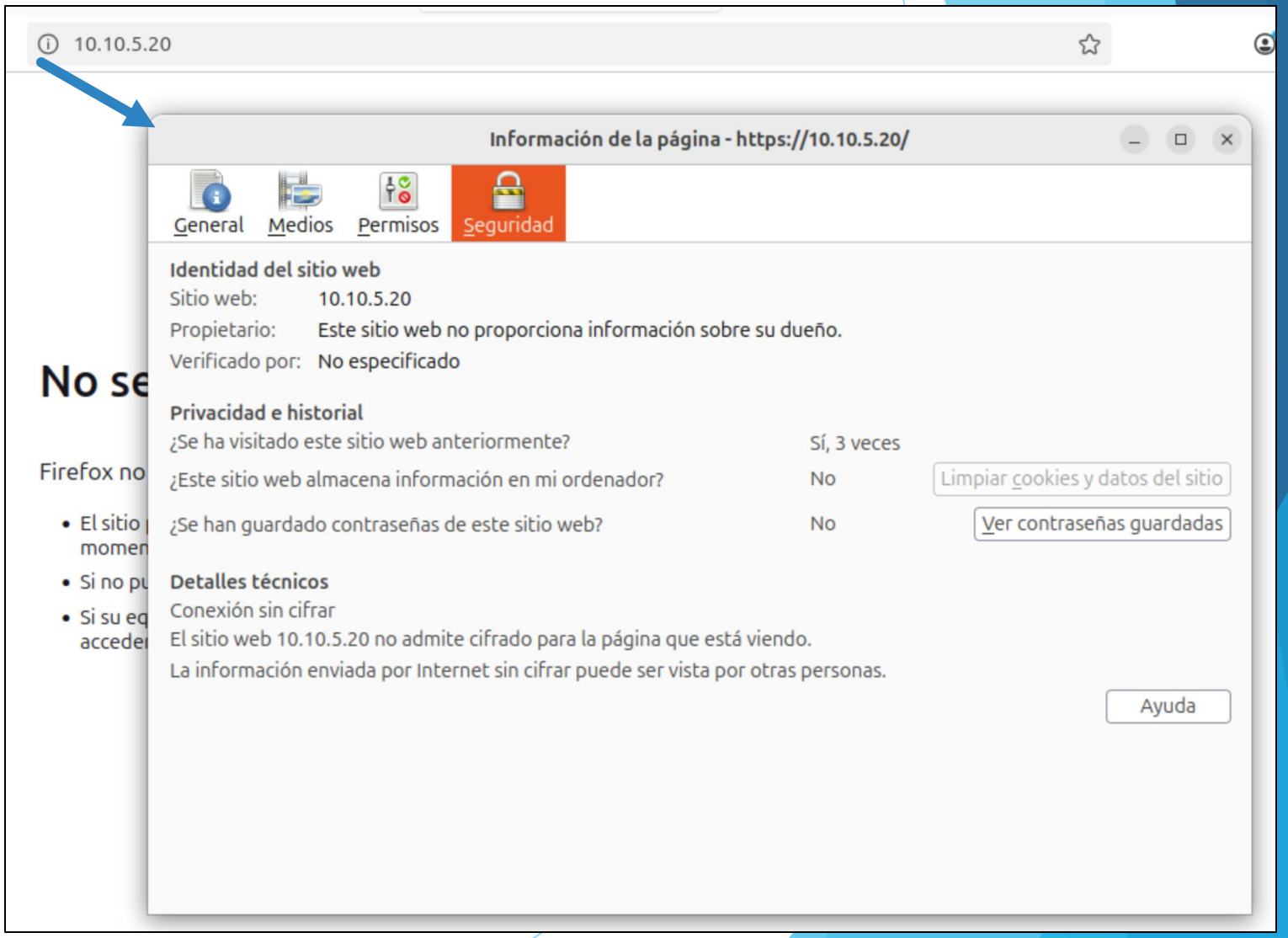
En ese caso, se debe aceptar el certificado como certificado válido.



9 - Servidor seguro en Apache

Aunque se acepte un certificado no validado el navegador web muestra una alerta de servidor no seguro.

Se puede consultar la información del certificado de una aplicación web segura.



9 - Servidor seguro en Apache

Suponiendo que hemos creado un certificado para www.prueba.com (ejemplo), podríamos abrirlo y ver algo como lo que se muestra en la imagen.

Actividades

Aules

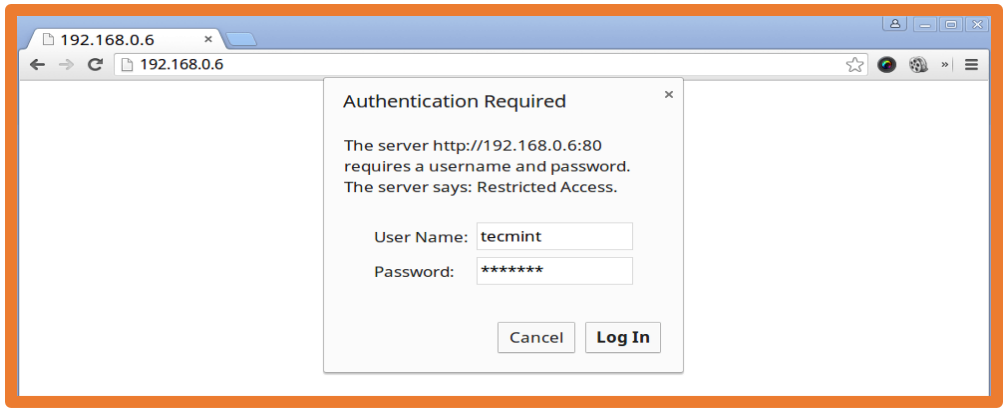
ACTIVIDAD 4 – CONFIGURACIÓN DE UN SITIO WEB SEGURO

A lo largo de la práctica, como ejemplo, hemos ido creando el sitio web prueba.com. Pero, solo se podía acceder usando HTTP (*acceso no seguro*). Basándote en los pasos explicados en el apartado 9 de la teoría, configura el servidor web Apache para poder acceder al sitio web prueba.com desde el navegador usando HTTPS (https://www.prueba.com). Crea un certificado con tu nombre.

Ten en cuenta que al ser un certificado autofirmado y no ser un certificado público, te dirá que no es seguro. Debes aceptar para poder acceder a la web.

10 - Control de acceso

El servidor web Apache da la oportunidad de permitir el acceso según la IP del cliente o con un usuario y contraseña. Estos métodos de control de acceso no son muy recomendables porque bloquean la petición.



Como se puede ver en el ejemplo, el pop-up de autentificación sale antes de que se cargue la página web. El usuario puede no saber en qué aplicación web se encuentra. Con las tecnologías actuales es más recomendable diseñar un sistema de autentificación con la base de datos.

10 - Control de acceso

RESTRICCIÓN POR DIRECCIÓN IP

En la directiva Directory se puede utilizar la opción **Require ip** para indicar qué direcciones IP pueden acceder al repositorio. Controla quién puede acceder al contenido de una carpeta concreta del servidor web, según el ejemplo, al directorio */var/www/html/profesor*.

EJEMPLO

<Directory /var/www/html/profesor>

Options Indexes

FollowSymLinks

AllowOverride None

Require ip 127.0.0.1 → Solo el propio servidor (localhost) puede acceder.

Require ip 192.168.1.16 → También puede entrar un cliente en esa IP de la red local.

</Directory>

10 - Control de acceso

RESTRICCIÓN POR DIRECCIÓN IP

Apache permitirá el acceso a los archivos dentro de /var/www/html/profesor solo si la petición proviene de:

- La máquina local (127.0.0.1, es decir, el propio servidor).
- El cliente con IP 192.168.1.16 (por ejemplo, el PC del profesor).

Cualquier otro equipo (por ejemplo 192.168.1.50, o desde fuera de la red) recibirá un error 403 Forbidden.

Por ejemplo, se puede tener esta estructura de la web, donde se pueda acceder siempre a la página principal (index) pero a la información dentro del directorio /profesor solo pueda acceder un equipo en concreto.

```
/var/www/html/
├── index.html
└── profesor/
```

10 - Control de acceso

AUTENTICACIÓN HTTP BASIC Y HTTP DIGEST

HTTP Basic es el método más simple de autenticación en Apache. Funciona enviando las credenciales (usuario y contraseña) codificadas en base64 dentro de la cabecera HTTP.

Flujo:

1. El usuario intenta acceder a una zona protegida.
2. Apache responde con 401 Unauthorized y una cabecera WWW-Authenticate: Basic realm="Zona privada".
3. El navegador muestra un cuadro de diálogo pidiendo usuario y contraseña.
4. El navegador envía las credenciales codificadas en base64.
5. Apache las compara con el archivo .htpasswd.
6. Si coinciden → acceso permitido; si no → 401 de nuevo.

HTTP Digest es otro método más seguro que evita enviar la contraseña en texto claro (ni siquiera codificada en base64).Pero Digest ya casi no se usa, porque:

- No es compatible con todos los navegadores modernos.
- HTTPS (TLS) ya cifra la comunicación, haciendo **Basic sobre HTTPS** lo suficientemente seguro y más simple.

10 - Control de acceso

AUTENTICACIÓN HTTP BASIC SOBRE HTTPS

Por tanto, vamos a ver cómo se debe configurar la autenticación HTTP Basic sobre HTTPS, autenticación simple (usuario/contraseña) **pero protegida por cifrado TLS**, para que las credenciales viajen seguras.

Primero se debe tener en cuenta que hay unos **requisitos previos**:

Requisito	Comando / Acción
Apache instalado y actualizado	sudo apt install apache2
Paquete de administración adicional	sudo apt install apache2-utils
HTTPS configurado (certificado activo)	sudo a2enmod ssl + VirtualHost en *:443
Módulos de autenticación activos	sudo a2enmod auth_basic authn_file
Reiniciar/reload	sudo systemctl reload apache2

Segundo, debemos crear el archivo de credenciales (usuarios y contraseñas) en **/etc/apache2/.htpasswd**:

EJEMPLO **#sudo htpasswd -c /etc/apache2/.htpasswd profesor**

Al ejecutar el comando pedirá que pongas una contraseña para el usuario ‘profesor’ y la guardará cifrada en el archivo .htpasswd.

10 - Control de acceso

AUTENTICACIÓN HTTP BASIC SOBRE HTTPS

Se puede realizar la configuración de HTTP Basic en el fichero de configuración (.conf) o en el archivo .htaccess. Recuerda que este fichero .htaccess solo es recomendable en casos de hosting compartido.

CONFIGURACIÓN EN EL FICHERO .conf

En el ejemplo lo que se va a configurar es que solo usuarios con contraseña válida puedan acceder desde el navegador al contenido del directorio 'profesor' (por ejemplo, a https://www.prueba.com/profesor).

EJEMPLO. En /etc/apache2/sites-available/prueba.conf

- Aplica HTTPS obligatorio (toda la comunicación cifrada).
- Protege /var/www/prueba.com/profesor con usuario y contraseña.
- El navegador mostrará un cuadro de autenticación.
- Credenciales seguras gracias a la capa SSL/TLS.

```
<VirtualHost *:443>
    ServerName www.prueba.com
    DocumentRoot /var/www/prueba.com

    SSLEngine on
    SSLCertificateFile /etc/ssl/prueba.com/prueba.crt
    SSLCertificateKeyFile /etc/ssl/prueba.com/prueba.key
    SSLCertificateChainFile /etc/ssl/prueba.com/ca-chain.crt

    <Directory /var/www/prueba.com/profesor>
        AuthType Basic
        AuthName "Zona privada (profesor)"
        AuthUserFile /etc/apache2/.htpasswd
        Require valid-user
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/prueba_ssl_error.log
    CustomLog ${APACHE_LOG_DIR}/prueba_ssl_access.log combined
</VirtualHost>
```

10 - Control de acceso

AUTENTICACIÓN HTTP BASIC SOBRE HTTPS

CONFIGURACIÓN EN EL FICHERO .htaccess

Esta opción es útil si no puedes editar el .conf directamente (por ejemplo, en hosting compartido). Debes permitir .htaccess en tu VirtualHost:

EJEMPLO

Primero, en **/etc/apache2/sites-available/prueba.conf,** añade estas directivas (*solo funcionará si AllowOverride AuthConfig está habilitado*):

```
<Directory /var/www/prueba.com/profesor>
    AllowOverride AuthConfig
    Require all granted
</Directory>
```

Segundo, en el directorio **/var/www/prueba.com/profesor/.htaccess**, crea un archivo:

```
AuthType Basic
AuthName "Zona privada (profesor)"
AuthUserFile /etc/apache2/.htpasswd
Require valid-user
```

10 - Control de acceso

AUTENTICACIÓN HTTP BASIC SOBRE HTTPS

Se puede **combinar la restricción de acceso por una dirección IP con la autenticación por usuario y contraseña.** Si quieres que solo una IP específica (por ejemplo, la del profesor) pueda acceder además de tener que autenticarse, dentro del fichero apache2.conf o del VirtualHost, deberías indicar lo siguiente:

EJEMPLO

```
<Directory /var/www/prueba.com/profesor>
    AuthType Basic
    AuthName "Zona privada (profesor)"
    AuthUserFile /etc/apache2/.htpasswd
    <RequireAll>
        Require ip 192.168.1.16
        Require valid-user
    </RequireAll>
</Directory>
```

Eso exige dos condiciones simultáneas:

- Conectarse desde la IP 192.168.1.16
- Identificarse con usuario/contraseña válida.

Si queremos obligar a que solo se use HTTPS, dentro del VirtualHost se puede incluir una redirección (Redirect)

```
<VirtualHost *:80>
    ServerName www.prueba.com
    Redirect permanent / https://www.prueba.com/
</VirtualHost>
```

Actividades

Aules

ACTIVIDAD 5 – CONTROL DE ACCESO

Configura el servidor web para implementar el control de acceso sobre el dominio www.prueba.com. El objetivo es proteger áreas restringidas del sitio web combinando autenticación HTTP Basic con restricción por dirección IP, aplicando las configuraciones directamente en el archivo del sitio (/etc/apache2/sites-available/prueba.conf), sin utilizar el archivo .htaccess.

El sitio web dispondrá de una página pública accesible a todos los usuarios y una zona privada ubicada en /var/www/prueba.com/privado, que requerirá autenticación mediante usuario (TU NOMBRE) y contraseña (1234), además de limitar el acceso únicamente al cliente de la red interna.

Crea dentro del directorio 'privado' un fichero llamado tunombre.html donde muestres tus datos personales.

Comprueba desde el cliente, que puedes acceder con tus credenciales a la zona privada de la web. Muestra qué ocurre cuando no indicas bien las credenciales y no tienes acceso.