Unidad 3

# Servicio web

## Servicios de red e internet

3

# Índice

# Introducción

Nosotros conocemos comúnmente internet como las páginas web, por eso es muchas veces la principal cara de una empresa y por tanto uno de los mayores aspectos a cuidar.

Aunque del diseño y de que la página web quede bonita, se encargan los diseñadores, y de la construcción de la página se encargan los desarrolladores, un administrador de sistemas debe de mantener el servicio en funcionamiento, hacer las primeras instalaciones y asegurarse de que hay una protección suficiente para los datos alojados en las páginas webs.

A lo largo de estos apartados, veremos las principales configuraciones de un servidor web y como proteger el sistema de manera más o menos eficiente.

# Al finalizar esta unidad

- Sabremos distinguir los fundamentos y protocolos en los que se basa el funcionamiento de un servidor web.
- Seremos capaces de instalar y configurar un servidor web,
- Podremos ampliar la funcionalidad del servidor mediante la activación y configuración de módulos.
- Conoceremos como crear y configurar sitios virtuales.
- Seremos capaces de configurar mecanismos de autenticación y control de acceso del servidor,
- Podremos obtener e instalar certificados digitales.
- Conoceremos como establecer mecanismos para asegurar las comunicaciones entre el cliente y el servidor.
- Sabremos realizar pruebas de monitorización del servicio.
- Seremos capaces de analizar los registros del servicio para elaborar estadísticas y resolver incidencias.

# 3.1. Características generales del servicio web

El servicio web es probablemente uno de los principales servicios en ser utilizado, porque, además, es uno de los primeros. Este servicio es usado para que el usuario final pueda ver información que tenemos depositada en servidores de manera cómoda y amigable. Esta información aparece recogida en las *páginas web* que se crean con este servicio y que necesitan de lenguajes de marcas y de lenguajes de programación para terminar de tener el formato ideal.

## 3.1.1. Protocolo HTTP

*HyperText Transfer Protocol*, más conocido como HTTP es el protocolo usado para las páginas web. Se aloja en la capa de aplicación y hace que se puedan visualizar las páginas web de forma más legible o con aspecto más amigable. Este protocolo no tiene estado, es decir, los paquetes se tratan como elementos independientes, Suele funcionar sobre TCP y en el puerto 80.

Vamos ahora a explicar una serie de conceptos que hay que tener en cuenta para cuando trabajemos en HTML.

### URI, Uniform Resource Identifier

La cadena de texto que nos ayuda a identificar cualquier recurso de una red, desde una imagen hasta una página web completa, se denomina URI. Se encuentra formada por los siguientes campos separados por barras "/":

| Campos de URI | | | | | | |
|---|---|---|---|---|---|---|
| Protocolo | :// | Nombre *host* | / | Nombre directorio | / | Nombre archivo |

Y estos hacen referencia a:

- **Protocolo:** nos indica que protocolo se usa para el acceso al recurso.
- **Nombre host:** indica el nombre de dominio o la dirección IP del equipo donde se encuentra alojado el recurso.
- **Nombre de directorio:** es toda la ruta de directorios que hay que seguir hasta llegar al directorio que contiene al recurso. Cuando no especificamos nada, se tratará del directorio *home* que tenga el servidor configurado por defecto.
- **Nombre de archivo:** nombre con el que tengamos guardado el recurso.

No todos los valores de acceso con URI son necesarios, algunos son optativos.

### URL, Uniform Resource Locator

Es una cadena de texto que nos indica el camino que debemos seguir hasta un recurso in internet. Realmente es un URI particular que debe de llevar siempre todos losa campos que existen, porque si alguno no aparece, se ponen los valores por defecto.

### URN, Uniform Resource Name

Esta cadena de texto también identifica un recurso sin indicar donde se encuentra alojado, es decir, es un nombre único para dicho recurso.

## 3.1.2. Funcionamiento del protocolo HTTP

El funcionamiento de este protocolo se basa en el procedimiento pregunta-respuesta. Este procedimiento se basa en que el cliente realizará una petición de información a un servidor web y el servidor devuelve un código de estado que indica el resultado a la petición a veces acompañado de información, como puede ser, una página web. Toda información que acompañe a la repuesta se identifica mediante la especificación MIME.

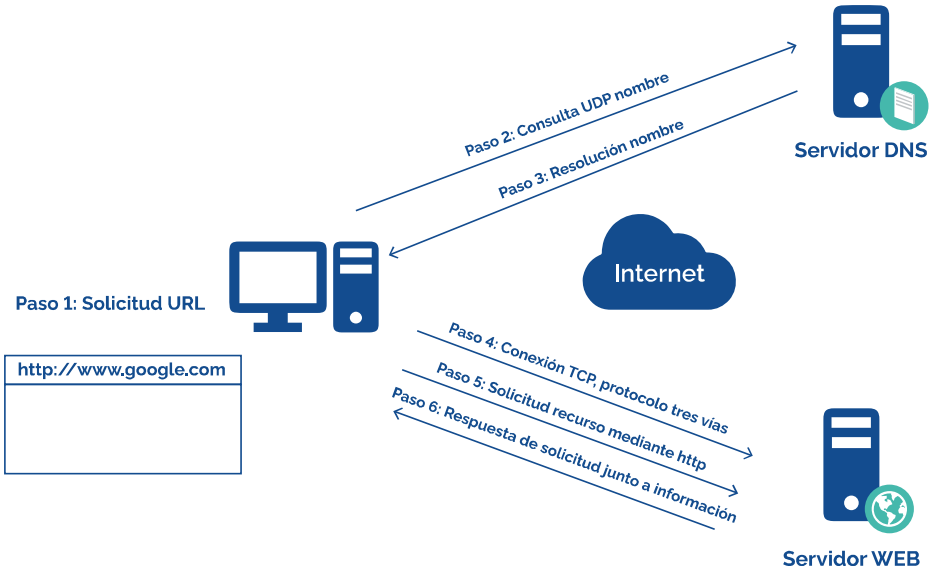


Imagen 1. Procedimiento de petición y respuesta de protocolo HTTP

El proceso de solicitud es el siguiente:

1. Lo primero que hacemos es una solicitud UDP al DNS para conocer la dirección IP de destino.
2. Una vez conocida, se inicia una conexión TCP con el servidor web y se solicita el recurso.
3. Se realiza un protocolo de tres vías o *handshake* entre cliente y servidor.
4. Se negocian en la misma conexión los puertos de origen y de destino para poder identificar el par *socket*.
5. Comienza el intercambio de la información para cada uno de los recursos solicitados.

## Peticiones HTTP

Los comandos que más se usan en las líneas de inicio de mensajes HTTP son `GET` para indicar la solicitud de un archivo y que se abra en el cliente y `POST`, que nos indica que se ha enviado información hacia el servidor, independientemente del modo. Justo después nos encontramos con el nombre del recurso que va a ser objeto del comando y, por último, la versión del protocolo que se usa en la petición.

Los encabezados que siguen en el mensaje se clasifican como:

- **General headers.** Son los valores que afectan al mensaje en conjunto.
- **Request headers.** Son los valores que añaden ya cierto nivel de especificación sobre los valores generales.
- **Entity headers.** Son los valores aplicados al cuerpo del mensaje, por lo que, si no existe el cuerpo, o está vacío, no son obligatorios.

Es común que en las peticiones `GET`, no lleven contenido, pero puede pasar también con otros comandos.

## Respuestas HTTP

Las primeras líneas de las respuestas las llamamos *líneas de estado* y se encuentran formadas por:

- **La versión del protocolo.**
- **Código de estado de la petición.** Se trata de un número entero que nos indicará una respuesta u otra a la petición dependiendo del valor que tome. Tenemos cinco categorías principales para los mensajes de estados, indicando el primer digito la respuesta a la petición principal y los siguientes la especificación dentro del grupo de mensajes:
  - 1XX: son los mensajes informativos.
  - 2XX: son los mensajes s que indican la correcta comunicación.
  - 3XX: son los mensajes que nos indican que ha habido una redirección, es decir, que el recurso ha sido movido.
  - 4XX: son los errores del cliente.
  - 5XX: son los errores en el servidor.

Los mensajes más comunes y que más solemos ver son:

| Código | Valor | Descripción |
|---|---|---|
| 200 | OK | La solicitud ha sido exitosa y nos devuelve la información solicitada. |
| 201. | Created | La respuesta ha sido exitosa y se ha creado un recurso al generar el resultado. |
| 300 | Multiple choice | Se ha dado más de una respuesta y el cliente debe de seleccionar alguna. |
| 301 | Moved permanently | Se ha movido el recurso solicitado a una nueva localización. No es necesario reescribir la localización. |
| 302 | Moved temporary | Se ha movido el recurso y se ha respondido indicando una nueva localización que lo mismo no está disponible. |
| 400 | Bad request | La petición tenía una sintaxis no entendible por parte del servidor |
| 401 | Unauthorized | Se necesita autorización para el acceso al recurso |
| 403 | Forbidden | Si no tenemos permisos suficientes para el acceso al servidor, se nos rechaza la entrada sin necesidad de autenticación. |
| 404 | Not Found | No se encuentra el recurso. |
| 407 | Proxy Authentication Required | Parecido al 401 pero con la autenticación llevándose a cabo desde un proxy. |
| 500 | Internal Server Error | Error producido en el servidor a nivel general. |
| 503 | Service Unavailable | No se gestiona la petición si el servidor está caído |

Por último, el encabezado nos devuelve la información que nos sea útil y en el cuerpo se incluye los datos de respuesta a nuestra petición.

# 3.2. Configuración básica de un servidor web

Tenemos varios métodos de configuración de un servidor web, como *Nginx*, pero nosotros en esta unidad vamos a ver la instalación de *Apache* en Ubuntu Server.

Como siempre, esto se va a ver en base a ejemplos.

## 3.2.1. Instalación Apache

La versión de Apache que vamos a usar es *Apache2*, que es la más usada en los últimos años y sabemos que funciona de manera correcta.

Para realizar la instalación seguimos los siguientes pasos:

1. *Logueamos* como *root* en un terminal.
2. Actualizamos repositorios y paquetes.
3. Instalamos Apache con el comando:

   ```
   apt install apache2
   ```

```
root@servidor:~# apt install apache2
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  apache2-bin apache2-data apache2-utils bzip2 libapr1 libaprutil1 libaprutil1-dbd-sqlite3
  libaprutil1-ldap libjansson4 liblua5.3-0 mailcap mime-support ssl-cert
Paquetes sugeridos:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser bzip2-doc
Se instalarán los siguientes paquetes NUEVOS:
  apache2 apache2-bin apache2-data apache2-utils bzip2 libapr1 libaprutil1 libaprutil1-dbd-sqlite3
  libaprutil1-ldap libjansson4 liblua5.3-0 mailcap mime-support ssl-cert
0 actualizados, 14 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 2.116 kB de archivos.
Se utilizarán 8.510 kB de espacio de disco adicional después de esta operación.
```

Imagen 2. Instalación de *Apache2*

4. Una vez que sabemos que el servicio está instalado, vamos a comprobar que esté activo:

   ```
   systemctl status apache2
   ```

```
root@servidor:~# systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2022-06-27 12:31:41 UTC; 24s ago
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 2416 (apache2)
      Tasks: 55 (limit: 2289)
     Memory: 5.0M
        CPU: 19ms
     CGroup: /system.slice/apache2.service
             ├─2416 /usr/sbin/apache2 -k start
             ├─2418 /usr/sbin/apache2 -k start
             └─2419 /usr/sbin/apache2 -k start

jun 27 12:31:41 servidor systemd[1]: Starting The Apache HTTP Server...
jun 27 12:31:41 servidor apachectl[2407]: AH00558: apache2: Could not reliably determine the server>
jun 27 12:31:41 servidor systemd[1]: Started The Apache HTTP Server.
lines 1-16/16 (END)
```

Imagen 3. Estado de *Apache2*

5. Ahora, nos podemos ir al directorio donde se alojan los archivos de configuración de Apache, que es */etc/apache2* y veríamos lo siguiente:


Imagen 4. Directorio */etc/apache2*

6. En este directorio tenemos:

> `apache2.conf:` es el principal fichero de configuración y su nombre a veces varía dependiendo de la versión de Apache y del sistema operativo usado.

> `conf-available:` el directorio donde se alojan los ficheros de configuración especifica de Apache2.

> `conf-enabled:` este directorio contiene enlaces simbólicos hasta el directorio *conf-available*.

> `envvars:` el fichero que contiene las variables de entorno de Apache2.

> `magic:` las instrucciones que determinan los tipos MIME y se basan en los primeros bytes de los archivos.

> `mods-available:` el directorio que contiene los módulos que se pueden añadir al servicio de manera adicional.

> `mods-enabled:` contiene enlaces simbólicos hacia *mods-available.*

> `ports.conf:` fichero de configuración de los puertos de escucha del servidor.

> `sites-available:` es el directorio en el que se alojan los ficheros para la configuración de *hosts* virtuales. Cada *host* va a tener su propio fichero de configuración.

> `sites-enabled:` Directorio que contiene los enlaces simbólicos hasta el anterior directorio.

Además, dentro del fichero *apache2.conf*, podemos definir principalmente las siguientes directivas:

> `ServerRoot.` Indica cual es el directorio principal de almacenamiento del servicio, además de sus errores y ficheros de registro.

> `Timeout.` El número de segundos máximo que pueden pasar entre petición del cliente y respuesta del servidor.

> `KeepAlive.` Es la directiva que activa las conexiones TCP persistentes.

> `ErrorLog.` Define el fichero de almacenamiento de los *logs*.

> `LogLevel.` Identifica cual es el nivel de gravedad de los distintos eventos.

> `Include.` Esta directiva, que ya hemos visto en otros servicios, indica que ficheros con contenido se tienen también en cuenta en este fichero. Es decir, simula que ese mismo contenido estuviera en este mismo fichero.

> `LogFormat.` Indica cual es el formato para incluir eventos dentro de los registros.

> `ErrorDocument.` Define cuales son las páginas personalizadas para los errores del sistema.

Algunas directivas ya aparecen en el fichero a modo de comentario, pero otras habrá que añadirlas de manera manual.

## 3.2.2. Configuración básica

Apache funciona en su mayoría por *host* virtuales, de modo que cada uno de los *hosts* representa un *site* al completo, o una página. Esto es porque cada uno tiene su propio fichero de configuración, y puede que hasta una estructura de directorios para el solo. Una vez el servicio esta instalado, se crea por defecto un primer *host virtual* como sitio principal del servidor web.

El fichero que se crea por defecto es *000-default.conf* y se aloja en */etc/apache2/sites-available*.

```
root@servidor:/etc/apache2# cd sites-available/
root@servidor:/etc/apache2/sites-available# ls
000-default.conf  default-ssl.conf
root@servidor:/etc/apache2/sites-available#
```

Imagen 5. Directorio */etc/apache2/sites-available*

El contenido del fichero es adecue a la siguiente imagen:

```
<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.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        # 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>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Imagen 6. Fichero *0000-default.conf*

Podemos ver que se definen una serie de directivas dentro de este *host* virtual, entre las que destacan:

- `ServerName`**.** El nombre de domino o FQDN que deben de usar los visitantes para poder visualizar los recursos web. Si creamos varios *hosts* virtuales, deben de usar nombres distintos.
- `ServerAdmin`**.** Es el correo electrónico del administrador del servicio.
- `DocumentRoot`**.** El subdirectorio donde se van a alojar los recursos que son accesibles desde los clientes de la web. En este directorio se pueden crear otros subdirectorios para el almacenamiento de información din problema, pero por defecto siempre será */var/www/html*.
- `Include`**.** Igual que en anteriores ficheros.

Si nos vamos al directorio */var/www/html*, podríamos ver que por defecto se crea un *index.html* que determina la página de inicio del servicio por defecto.

```
root@servidor:/etc/apache2/sites-available# cd /var/www/html/
root@servidor:/var/www/html# ls
index.html
root@servidor:/var/www/html# _
```

Imagen 7. Directorio */var/www/html*

Esta página de inicio tendría el siguiente formato como página *html*:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xht
<html xmlns="http://www.w3.org/1999/xhtml">
  <!--
    Modified from the Debian original for Ubuntu
    Last updated: 2016-11-16
    See: https://launchpad.net/bugs/1288690
  -->
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Apache2 Ubuntu Default Page: It works</title>
    <style type="text/css" media="screen">
  * {
    margin: 0px 0px 0px 0px;
    padding: 0px 0px 0px 0px;
  }

  body, html {
    padding: 3px 3px 3px 3px;

    background-color: #D8DBE2;

    font-family: Verdana, sans-serif;
    font-size: 11pt;
    text-align: center;
  }

  div.main_page {
    position: relative;
    display: table;

    width: 800px;
```

Imagen 8. *index.html* por defecto

Lógicamente, más adelante, veremos como ir cambiando estas páginas.

## 3.2.3. Comprobación del funcionamiento

Vamos a ver un ejemplo muy simple de como comprobar si el servidor web está funcionando:

1. Lo primero que hacemos es dirigirnos a otra máquina que se encuentre en la misma red que la nuestra.
2. Dentro de la máquina, abrimos un navegador web.
3. En el navegador, colocamos la dirección IP que tiene asignada nuestro servidor, que en este caso es 10.0.0.24.
4. Si todo funciona de manera correcta, veríamos la siguiente página:

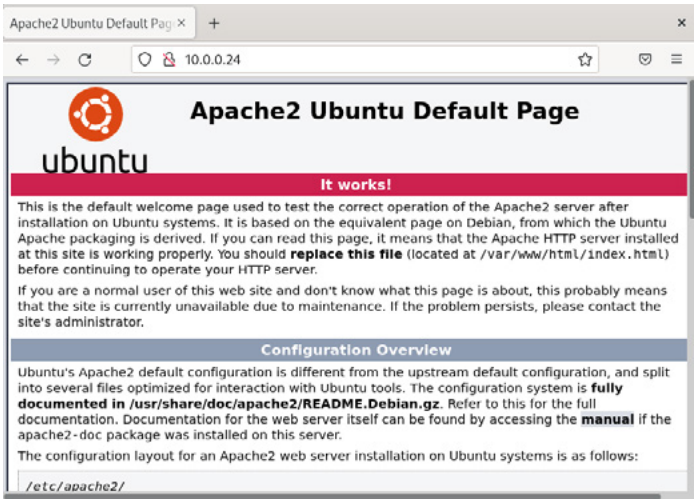


Imagen 9. Página de bienvenida de Apache

Esta página es realmente lo que se visualiza al desarrollar el fichero *index.html* que vimos en anteriores apartados.

**PARA TENER EN CUENTA...**

La página inicial de apache nos dice ya que el servidor está funcionando.

## 3.2.4. Cambiar página de error tipo 404 por defecto

Cuando tenemos un error porque no podemos llegar a un recurso específico de la red, el error más común es "*Error 404 Not Found*", pero esto se puede cambiar.

Vamos a explicar los pasos para llegar a cambio estos cambios:

1. Nos dirigimos al directorio */var/www/html*.
2. En este directorio, creamos con **nano** un archivo llamado *ejemploerror.html* que quedaría del siguiente modo:

```
 GNU nano 5.6.1                                  ejemploerror.html
<html>
<title>Error de documento</title>
<h1>Error de documento</h1>
<p>Esto es un ejemplo de página de error</p>
</html>_
```

Imagen 10. *ejemploerror.html*

3. Ahora, editamos el fichero */etc/apache2/apache2.conf* añadiendo la siguiente directiva:

```
ErrorDocument 404 /ejemploerror.html
```

```
ErrorDocument 404 /ejemploerror.html
```

Imagen 11. Configuración que añadir en *apache2.conf*

4. Guardamos el archivo y reiniciamos el servicio *apache2*.
5. Nos dirigimos a otra máquina de la misma red y en un navegador web, ponemos la dirección del servidor */algoinventado* para que nos de error aposta, y debería de verse como en la siguiente imagen:

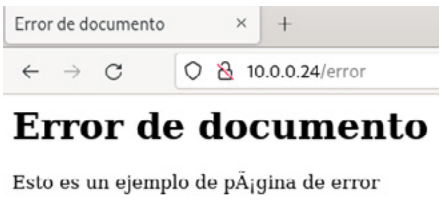


Imagen 12. Comprobación del error

**PARA TENER EN CUENTA...**

Si vemos la imagen de comprobación del error de página, no se ven bien las tildes porque en la estructura del documento HTML no hemos indicado correctamente el idioma.

# 3.3. Módulos: instalación, configuración y uso

Apache permite añadir implementaciones de funcionalidad y configuración mediante los módulos.

Estos *módulos* vienen en algunos casos incluidos por defecto, pero en toros muchos tendremos que añadirlos nosotros.

Los módulos los tenemos disponibles en el directorio */etc/apache2/mods-available*.

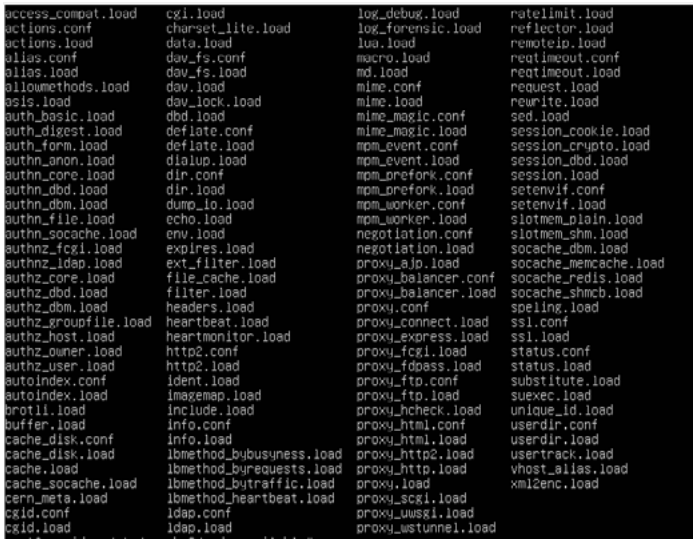


Imagen 13. Directorio *mods-available*

Como podemos ver en la anterior imagen, para cada módulo, tenemos dos ficheros distintos:

- **Módulo.load:** es el fichero contenedor de la directiva de carga del módulo en el servicio.
- **Módulo.conf:** es el fichero de configuración del módulo que se usa para las configuraciones específicas únicamente de dicho módulo.

Para activar un módulo, tenemos varios modos, el más usado es mediante la herramienta nativa **a2enmod**. Al usar este comando sin argumentos, nos aparecerán todos los módulos que podemos habilitar para que seleccionemos uno.


Imagen 14. a2enmod

Por lo general, si conocemos el nombre del módulo que queremos que se habilite, el comando sería del siguiente modo:

```
a2enmod módulo
```

Si queremos ver que módulos podríamos instalar de manera adicional, (que no aparezcan aquí), deberíamos de hacer una búsqueda en los repositorios, con el comando:

```
apt search libapache2-mod
```

Que nos devolverá un resultado como el siguiente:

```
Ordenando...
Buscar en todo el texto...
libapache-mod-jk-doc/impish 1:1.2.48-1 all
  Documentación del paquete libapache2-mod-jk

libapache2-mod-apparmor/impish 3.0.3-0ubuntu1 amd64
  changehat AppArmor library as an Apache module

libapache2-mod-apreq2/impish 2.13-7build1 amd64
  generic Apache request library - Apache module

libapache2-mod-auth-cas/impish 1.2 1 amd64
  CAS authentication module for Apache2

libapache2-mod-auth-gssapi/impish 1.6.3-1 amd64
  GSSAPI Authentication module for Apache2

libapache2-mod-auth-kerb/impish 5.4-2.5 amd64
  apache2 module for Kerberos authentication

libapache2-mod-auth-mellon/impish 0.17.0-1ubuntu1 amd64
  SAML 2.0 authentication module for Apache

libapache2-mod-auth-openid/impish 0.8-5build1 amd64
  OpenID authentication module for Apache2

libapache2-mod-auth-opencid/impish 2.4.9 1 amd64
  OpenID Connect authentication module for Apache

libapache2-mod-auth-pgsql/impish 2.0.3-6.1ubuntu2 amd64
  Module for Apache2 which provides PostgreSQL authentication

libapache2-mod-auth-plain/impish 2.0.52 amd64
  Módulo para Apache2 que provee de autenticación en texto plano

libapache2-mod-auth-pubtkt/impish 0.13-1 amd64
```

Imagen 15. apt search libapache2-mod

Por último, si queremos deshabilitar un módulo, funcionaría de igual manera que para habilitarlo, pero esta vez el comando es `a2dismod`, Al igual que con el comando de habilitación, se puede usar con o sin argumentos.

```
root@servidor:/etc/apache2/mods-available# a2dismod
Your choices are: access_compat alias auth_basic authn_core authn_file authz_core authz_host authz_u
ser autoindex deflate dir env filter mime mpm_event negotiation reqtimeout setenvif status
Which module(s) do you want to disable (wildcards ok)?
```

Imagen 16. a2dismod

## 3.3.1. Módulo de monitorización del servicio

Si queremos monitorizar un servicio web, tenemos diversas herramientas, pero para Apache, disponemos de dos módulos básicos para poder monitorizar y comprobar el estado, que son el de información y el de estado.

Para explicar ambos dos usaremos ejemplos con un ordenador con IP 10.0.0.51 estática que se conecta al servidor web y ve dichos módulos (la dirección del servidor es 10.0.0.24).

### Módulo de información

Para activar el primero de los módulos, debemos de seguir los siguientes pasos:

1. Entramos en una terminal y nos *logueamos* como *root*.
2. Nos dirigimos al directorio */etc/apache2/mods-available*.
3. Lanzamos el siguiente comando para habilitar el módulo de información:

   ```
   a2enmod info
   ```

```
root@servidor:/etc/apache2/mods-available# a2enmod info
Enabling module info.
To activate the new configuration, you need to run:
  systemctl restart apache2
root@servidor:/etc/apache2/mods-available# _
```

Imagen 17. a2enmod info

4. Una vez que vemos que se ha habilitado el módulo correctamente, retrocedemos un directorio con el comando:

   ```
   cd ..
   ```

5. En este directorio, abrimos el fichero *apache2.conf* y añadimos una directiva como se muestra a continuación:

   ```
   <Location /ruta_información>
     SetHandler     server-info
     Require ip     127.0.0.1
     Require ip     IP_equipos_conexión
   </Location>
   ```

```
#Información sobre módulo

<Location /info_servidor>
        SetHandler      server-info
        Require ip      127.0.0.1
        Require ip      10.0.0.51
</Location>
```

Imagen 18. Directiva de información

   Es muy importante respetar el campo *SetHandler* porque es el que indica que módulo se va a usar en dicha directiva.

6. Guardamos el fichero y reiniciamos el fichero.
7. Nos dirigimos a la máquina con la IP antes indicada y en el navegador web escribimos:

   ```
   Dirección_servidor/ruta_información
   ```

   Si todo funciona correctamente, nos debería de aparecer algo como en la siguiente imagen:

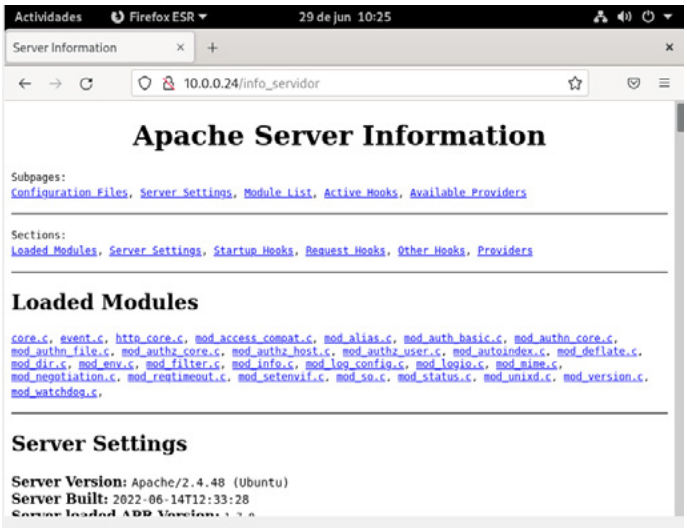


Imagen 19. Comprobación del módulo *info*

## Módulo de estado

Para activar el módulo de estado los pasos serían:

1. Una vez en el terminal y como *root*, habilitar el módulo con el comando:

   ```
   a2enmod status
   ```

```
root@servidor:/etc/apache2/mods-available# a2enmod status
Module status already enabled
root@servidor:/etc/apache2/mods-available# _
```

Imagen 20. a2enmod status

2. Añadir la directiva en el archivo *apache2.conf* del siguiente modo:

   ```
   <Location /ruta_estado>
   SetHandler server-status
     Require ip     127.0.0.1
     Require ip     IP_equipo_conexión
   </Location>
   ```

```
#Estado del servicio

<Location /estado_servidor>
        SetHandler server-status
        Require ip 127.0.0.1
        Require ip 10.0.0.51
</Location>_
```

Imagen 21. Directiva para el estado del servidor

3. En el equipo con la anterior IP, comprobar el funcionamiento con la ruta *IP_servidor/ruta_estado* en el navegador web.

4. Debe de aparecer una imagen como la siguiente:

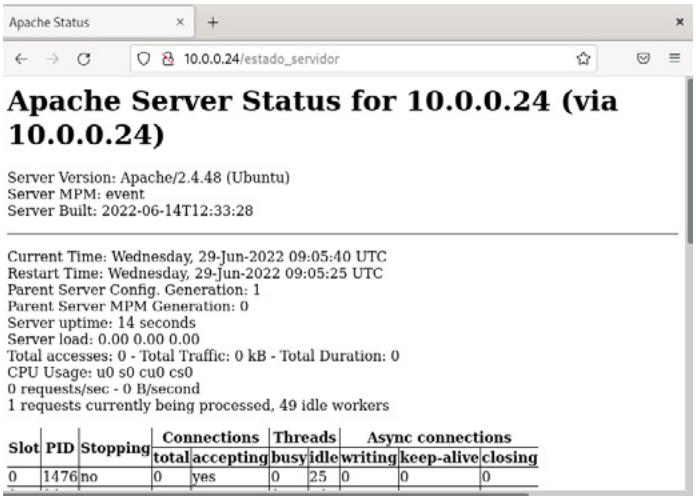


Imagen 22. Comprobación de funcionamiento del módulo de estado

## 3.3.2. Módulo PHP y MySQL

Posiblemente el módulo más usado a la hora de enseñar el funcionamiento de Apache es su módulo de PHP, que se usa para que se pueda integrar el lenguaje de *scripting* PHP en páginas HTML. Este siempre va de la mano del módulo de MySQL para que podamos acceder a la base de datos desde el lenguaje de programación del servidor. Dichos módulos o paquetes no vienen siempre por defecto en el servidor, y habrá que instalarlos o bien como paquetes o bien añadiendo módulos como vimos en apartados anteriores.

Los pasos que seguir son:

1. Como siempre, accedemos al terminal como usuario *root*.
2. Instalamos los paquetes necesarios:

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

```
root@servidor:~# apt install php libapache2-mod-php
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  libapache2-mod-php8.0 libsodium23 php-common php8.0 php8.0-cli php8.0-common php8.0-opcache
  php8.0-readline
Paquetes sugeridos:
  php-pear
Se instalarán los siguientes paquetes NUEVOS:
  libapache2-mod-php libapache2-mod-php8.0 libsodium23 php php-common php8.0 php8.0-cli
  php8.0-common php8.0-opcache php8.0-readline
0 actualizados, 10 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 4.955 kB de archivos.
Se utilizarán 20,6 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] _
```

Imagen 23. Instalación de paquetes 1

De manera opcional, podemos instalar los siguientes paqutes: **php-cli** y **php-cgi.**

```
root@servidor:~# apt install php-cli php-cgi
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  php8.0-cgi
Paquetes sugeridos:
  php-pear
Se instalarán los siguientes paquetes NUEVOS:
  php-cgi php-cli php8.0-cgi
0 actualizados, 3 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 1.583 kB de archivos.
Se utilizarán 9.765 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] _
```

Imagen 24. Instalación de paquetes 2

3. Si queremos conectar la PHP con la base de datos alojada en MySQL debemos de instalar el paquete de la base de datos, que es **php-mysql**.

```
root@servidor:~# apt install php-mysql
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  php8.0-mysql
Se instalarán los siguientes paquetes NUEVOS:
  php-mysql php8.0-mysql
0 actualizados, 2 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 138 kB de archivos.
Se utilizarán 492 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] _
```

Imagen 25. Instalación de paquetes 3

4. Para comprobar que todo lo anteriormente instalado está habilitado (se habilitan por defecto en la instalación), debemos de realizar lo siguiente:

   a. Lo primero es dirigirnos al directorio */etc/apache2/mods-enabled* y comprobar que aparecen los módulos correspondientes a *php8.0*.

```
root@servidor:~# cd /etc/apache2/mods-enabled/
root@servidor:/etc/apache2/mods-enabled# ls
access_compat.load  authz_host.load  dir.load     mpm_prefork.conf  reqtimeout.load
alias.conf          authz_user.load  env.load     mpm_prefork.load  setenvif.conf
alias.load          autoindex.conf   filter.load  negotiation.conf  setenvif.load
auth_basic.load     autoindex.load   info.conf    negotiation.load  status.conf
authn_core.load     deflate.conf     info.load    php8.0.conf       status.load
authn_file.load     deflate.load     mime.conf    php8.0.load
authz_core.load     dir.conf         mime.load    reqtimeout.conf
root@servidor:/etc/apache2/mods-enabled# _
```

Imagen 26. Módulos habilitados

   En caso de no estar habilitados, los habilitamos.

   b. El otro modo de comprobación es volviendo a ver en el navegador, la página de estado, donde aparecen los módulos habilitados.

5. Una vez hecho, se va a realizar una comprobación del funcionamiento del servicio:

   a. Nos dirigimos al directorio */var/www/html* y creamos un archivo terminado en *.php* que tenga la siguiente información.

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

Imagen 27. Función *phpinfo*

   Este archivo despliega una serie de informaciones sobre PHP en Apache2.

   b. En el navegador de nuestro cliente, introducimos lo siguiente:

```
IP_servidor/archivo.php
```

   c. Si todo funciona de manera correcta, nos debe de aparecer una imagen como la siguiente:

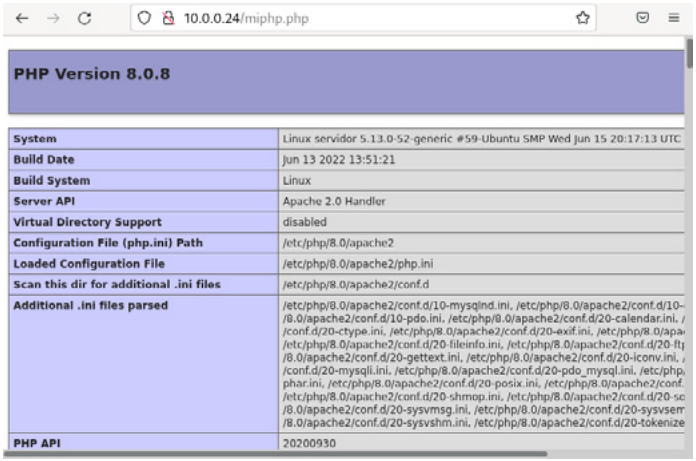


Imagen 28. Salida de *phpinfo* en navegador

# 3.4. Hosts virtuales: creación, configuración y utilización

En el apartado anterior, se vio que automáticamente se genera un *host* virtual por defecto con un *site* alojado al completo. Vamos a ver a continuación su creación al completo y con ejemplos.

## 3.4.1. Concepto de host virtual

Al instalar Apache como el servicio web, se hace sobre un *host* físico donde almacenamos las páginas y ficheros que se muestran en el directorio */var/www/html*. En dicho directorio, se incluyen el fichero *index.html* y los demás componentes de un único *site*.

Si tenemos la casuística de querer almacenar varios *istes* distintos con sus propios *index* de una única máquina, debemos de crear un subdirectorio por cada *iste* y después generar los dicheros *index* en cada uno de los distintos directorios.

Lo que queremos conseguir con esto, es que cada uno pueda tener una distinta configuración de modo que parezcan webs totalmente diferentes. Cada nombre de acceso al *site* será diferente uno de otro.

Para poder generar de manera correcta todo esto, es necesario que haya una definida estructura en estos directorios, recomendado tener siempre n listado con los nombres de los *hosts* virtuales y sus directorios de almacenamiento.

Vamos a ver ahora, como realizar un ejemplo de esta configuración en Ubuntu con Apache2.

## 3.4.2. Creación y configuración

Vamos a explicar ahora la creación de un supuesto sitio *www.universae.lan* en nuestro servidor, que será un *host* virtual. Hay que tener en cuenta que si queremos varios *hosts* se haría de igual manera.

Los pasos que seguir son los siguientes:

1. Creamos el directorio donde va a estar alojado el sitio de *universae.lan*, con el comando:

   ```
   mkdir -p /var/www/html/universae.lan
   ```

```
root@servidor:~# mkdir -p /var/www/html/universae.lan
root@servidor:~# ls /var/www/html/
ejemploerror.html  index.html  miphp.php  universae.lan
root@servidor:~#
```

Imagen 29. Creación del directorio

2. Una vez creados los directorios, cambiamos el propietario de este usando el comando `chown`:

   ```
   chown -R $USER:USER directorio
   ```

```
root@servidor:~# chown -R $USER:$USER /var/www/html/universae.lan/
root@servidor:~# _
```

Imagen 30. Cambio de propietario del directorio

   Esto realmente solo se realizará en caso de que no se creen los *sites* como usuario *root*.

3. Dentro del directorio creamos un archivo *index.html*.

4. Editamos el documento del modo que queramos, en nuestro caso:

```
<html>
        <head>
                <title> Site de Universae </title>
        </head>
        <body>
                <h1> ¡Bienvenido a  Universae! </h1>
                <p> Podemos comprobar que el site funciona correctamente </p>
        </body>
</html>
```

Imagen 31. Archivo index.html de nuestro site

5. Copiamos el archivo *000-default.conf* del directorio */etc/apache2/sites-available* en el mismo directorio pero con el nombre de nuestro *site* y terminado en *.conf*.

```
root@servidor:~# cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/unive
rsae.lan.conf
root@servidor:~# ls /etc/apache2/sites-available/
000-default.conf  default-ssl.conf  universae.lan.conf
root@servidor:~# _
```

Imagen 32. Copia de los ficheros de *site*

6. Editamos el archivo y añadimos:

   ```
   ServerAdmin correo_administrador
   DocumentRoot ruta_directorio_site
   ServerName nombre_site
   ServerAluas      www.nombre_site
   ```

   Todas estas directivas se añaden dentro de *<VirtualHost *:80>*.

```
<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.example.com

        ServerAdmin admin@localhost
        DocumentRoot /var/www/html/universae.lan
        ServerName universale.lan
        ServerAlias www.universae.lan_

        # 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>
```

Imagen 33. Configuración del *site*

7. Lanzamos el comando de habilitación del sitio:

   ```
   a2eniste archivo_sitio.conf
   ```

8. Deshabilitamos el sitio por defecto para que no haya colisiones:

   ```
   a2dissite 000-default.conf
   ```

```
root@servidor:~# a2ensite universae.lan.conf
Enabling site universae.lan.
To activate the new configuration, you need to run:
  systemctl reload apache2
root@servidor:~# a2dissite 000-default.conf
Site 000-default disabled.
To activate the new configuration, you need to run:
  systemctl reload apache2
root@servidor:~# _
```

Imagen 34. Habilitar y deshabilitar los sites

9. Para comprobar el funcionamiento, nos vamos a un cliente conectado directamente con el servidor.

   a. En este, como usuario *root* añadimos al servidor la entrada pertinente en el fichero */etc/hosts* de modo que quedaría como la siguiente imagen:

```
  GNU nano 6.2                          /etc/hosts *
127.0.0.1       localhost
127.0.1.1       debian
10.0.0.24       www.universae.lan
```

Imagen 35. Fichero */etc/hosts* del cliente

   b. En un navegador, metemos dicha dirección y vemos que nos sale la información específica que hemos añadido en nuestro nuevo *host* virtual o *site*.


Imagen 36. Comprobación del *site* en el navegador del cliente

**PARA TENER EN CUENTA...**

La configuración del fichero hosts del cliente se puede evitar si se definen las direcciones de los sitios en los archivos del DNS.

# 3.5. Autenticación y control de acceso

Nos referimos a autenticación como el control de acceso a un directorio en concreto alojado en nuestro servicio web. Vamos a explicar esto: es una medida tomada para delimitar el acceso a ciertos recursos del servidor web solicitando un usuario y una contraseña. Estos recursos son subdirectorios alojados en */var/www/html*.

El principal módulo que incorpora Apache por defecto es *mod_auth_basic* que si se habilita nos da la opción de especificar credenciales básicas para poder acceder a ciertos recursos.

Para configurar el módulo de Apache, se usa la herramienta `htpasswd`, que tiene los siguientes usos:

- `htpasswd -c fichero_usuarios_contraseñas usuario`. Con esta opción, se crea un fichero con el nombre que hemos especificado y el usuario también indicado. Cuando ejecutamos este comando, nos pedirán la contraseña para el usuario.
- `htpasswd fichero_usuarios_contraseñas usuario`. Realiza exactamente lo mismo que el comando anterior, pero sin solicitar contraseña, es decir, el usuario se crea sin contraseña.
- `htpasswd -D fichero_usuarios_contraseñas usuario`. Se elimina la entrada de usuario con el nombre que hayamos especificado del fichero que indiquemos.

Todas las contraseñas especificadas y alojadas en el fichero se encontrarán cifradas, no obstante, el cliente introduce las credenciales en texto plano y viaja hasta el servidor también en texto plano.

Vamos a ver un ejemplo guiado de como solicitar autenticación de un directorio llamado *private*:

1. Lo primero es crear dicho directorio en el directorio designado como raíz para los recursos de apache, */var/www/html*:

   ```
   mkdir /var/www/html private
   ```

   ```
   root@servidor:~# mkdir /var/www/html/private
   root@servidor:~#
   ```

   Imagen 37. Creación del directorio privado

2. Vamos a habilitar el módulo de autenticación:

   ```
   a2enmod auth_basic
   ```

   ```
   root@servidor:~# a2enmod auth_basic
   Considering dependency authn_core for auth_basic:
   Module authn_core already enabled
   Module auth_basic already enabled
   root@servidor:~#
   ```

   Imagen 38. Módulo de autenticación habilitado

3. Una vez que tenemos el modulo habilitado, lanzamos el comando de creación del fichero y de las credenciales de usuario, el comando sería:

   ```
   htpasswd -c contraseña.htpasswd profesor
   ```


Imagen 39. Asignación de contraseña cifrada a un usuario y almacenada en un fichero

Podemos observar, que nos ha solicitado dos veces seguidas la contraseña para crear el nuevo usuario. Además, se ha añadido el fichero en el directorio en el que estamos posicionados.

4. Podemos ver que la contraseña se crea cifrada dentro del fichero.


Imagen 40.Contraseña cifrada

5. Ahora nos vamos al fichero de configuración de nuestro *site* (en nuestro caso seguimos con el del apartado anterior), y añadimos lo siguiente dentro de la directiva <*VirtualHost *:80*>:

```
<Location /private>
    AuthType basic
    AuthName “Usuario y Contraseña”
    AuthUserFile   /var/www/html/private/contra-
    seña.htpasswd
    Require valid-user
</Location>
```

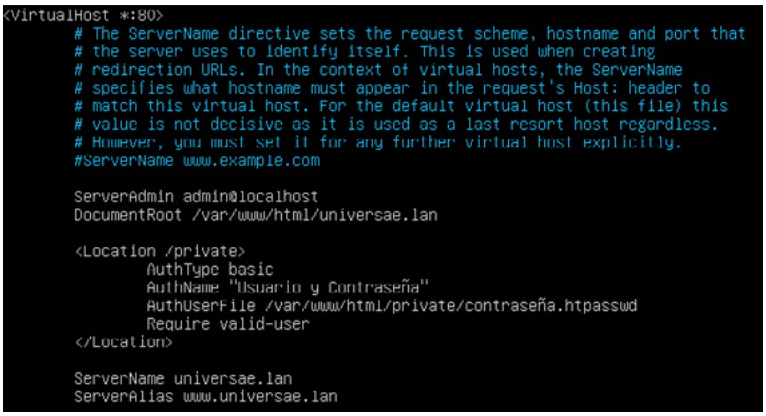


Imagen 41. Directiva *Location*

En la directiva definimos:

- Que directorio va a ser el que va a tener control de acceso.
- El tipo de autenticación.
- El nombre de la autenticación.
- La ruta del fichero anteriormente definido para la autenticación.
- Que solo pueden acceder usuarios autenticados.

6. Ahora comprobamos que efectivamente en un navegador cliente, nos solicita usuario y contraseña.

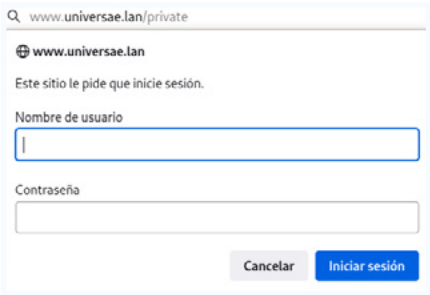


Imagen 42. Solicitud de credenciales en terminal web

# 3.6. Certificados y servidores de certificados

Cuando hacemos una consulta a una página web, casi siempre lo hacemos usando el protocolo *http/1.x*, que no nos oferta seguridad alguna. Esto también conlleva, que cuando visitamos un servidor, en primera instancia no podemos determinar que este sea original o si por el contrario está haciendo un uso fraudulento e intenta el robo de información.

Para poder poner algo más de protección al servicio web, usamos los siguientes mecanismos:

- **Certificados de autenticidad.** Se tratan de ficheros que otorgan las entidades reconocidas para asegurar que estamos trabajando con un servidor legítimo.
- **Protocolo SSL. SSL.** Hace referencia a *Secure Socket Layer* y es un protocolo alejado en la capa de aplicación que genera una comunicación cliente-servidor con los datos encriptados. Se usa en este caso un algoritmo de encriptación del tipo clave pública y clave privada.

Cuando entramos en una página web con la comunicación encriptada por el protocolo SSL, el protocolo cambia a HTTPS, y en la dirección veremos *https://*, también cambia su puerto de petición por el 443.

## 3.6.1. Obtención e instalación

Si queremos usar los dos mecanismos que hemos indicado antes para habilitar que la comunicación sea encriptada en Apache, tenemos que realizar los siguientes pasos:

1. Lo primero es, en un terminal como *root* habilitar el módulo de configuración de ssl:

   ```
   a2enmod ssl
   ```

```
root@servidor:~# a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create self-signed certifica
tes.
To activate the new configuration, you need to run:
  systemctl restart apache2
root@servidor:~#
```

Imagen 43. Habilitando el módulo de ssl

2. Creamos un directorio específico para almacenar los certificados que vamos a crear:

   ```
   mkdir directorio
   ```

```
root@servidor:~# mkdir /etc/apache2/ssl
root@servidor:~# ls /etc/apache2/
apache2.conf    conf-enabled  magic         mods-enabled  sites-available  ssl
conf-available  envvars       mods-available  ports.conf    sites-enabled
root@servidor:~# _
```

Imagen 44. Crear el directorio para almacenar los ficheros

3. Creamos los ficheros, el certificado y la clave privada:

```
openssl req -x509 -nodes -days 365 -newkey
rsa:2048 -keyout fichero_clave_privada.key -out
fichero_certificado.crt
```

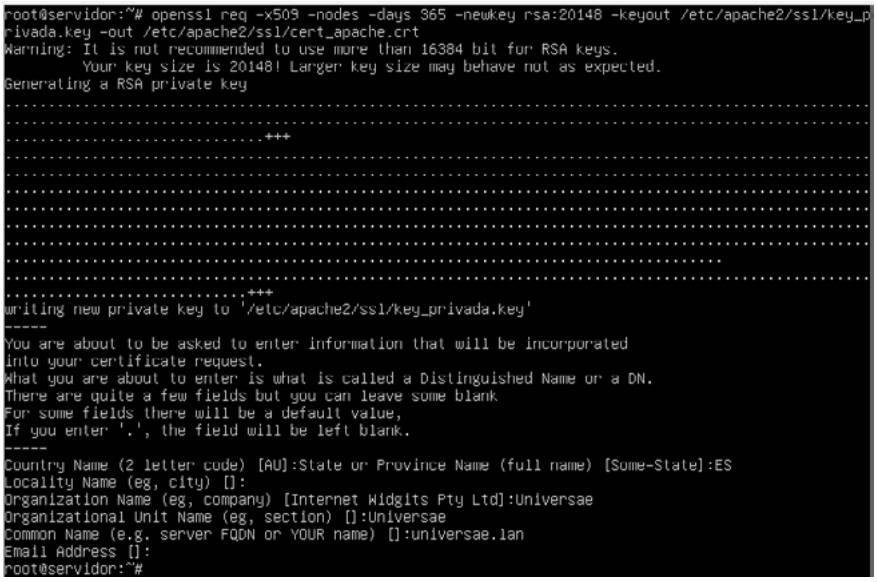


Imagen 45. Creación del certificado y el fichero de clave privada

En este punto, debemos tener en cuenta varias cosas:

- No siempre estará instalado *Openssl*, en caso de que no, habrá que instalarlo.
- Los parámetros indicados a tener en cuenta son:
  - `req -x509`*:* nos indica que el certificado que creemos crear es con el estándar de clave pública x509.
  - `-nodes`*:* hace referencia a no guardar el archivo con clave, así Apache no solicita credenciales cada vez que inicia.
  - `-days 365`*:* el certificado va a tener validez durante un año entero.
  - `-newkey rsa:2048`*:* la clave pública que queremos generar será de 2048 bits.
  - `-keyout`: indica la localización de la clave privada.
  - `-out`: indica la localización del certificado.
- Si todo está correctamente escrito, el mismo comando nos solicitará las siguientes cuestiones:
  - Nombre de estado.
  - Nombre de localidad.
  - Nombre de la organización.
  - Nombre del departamento.
  - Nombre común del certificado.
  - Dirección de correo electrónico.

  Todas estas preguntas son opcionales a la hora de responderlas.

4. Hacemos una copia de seguridad del archivo original de *site* de ssl-:

   ```
   cp /etc/apache2/sites-available/default-ssl.conf /
   etc/apache2/sites-available/default-ssl.conf.orig
   ```

```
root@servidor:~# cp /etc/apache2/sites-available/default-ssl.conf /etc/apache2/sites-available/defau
lt-ssl.conf.orig
root@servidor:~#
```

Imagen 46. Copia de seguridad

5. Ahora, editamos el fichero *default-ssl.conf* y debe de quedar como en la imagen siguiente:

```
<IfModule mod_ssl.c>
        <VirtualHost _default_:443>
                ServerAdmin admin@localhost
                ServerName www.universae.lan
                DocumentRoot /var/www/html

                # 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

                #   SSL Engine Switch:
                #   Enable/Disable SSL for this virtual host.
                SSLEngine on

                #   A self-signed (snakeoil) certificate can be created by installing
                #   the ssl-cert package. See
                #   /usr/share/doc/apache2/README.Debian.gz for more info.
                #   If both key and certificate are stored in the same file, only the
                #   SSLCertificateFile directive is needed.
                SSLCertificateFile      /etc/apache2/ssl/cert_apache.crt
                SSLCertificateKeyFile /etc/apache2/ssl/key_privada.key
```

Imagen 47. Fichero default-ssl.conf

Hay que tener en cuenta que aquí no se ve el final de la directiva, pero está, y que los parámetros realmente importantes son:

- `SSLEngine on`
- `SSLCertificateFile fichero_certificado`
- `SSLCertificateKeyFile fichero_clave_privada`

6. Habilitamos el *site* de SSL:

   ```
   a2ensite default-ssl.conf
   ```

```
root@servidor:/etc/apache2/sites-available# a2ensite default-ssl.conf
Enabling site default-ssl.
To activate the new configuration, you need to run:
  systemctl reload apache2
root@servidor:/etc/apache2/sites-available# _
```

Imagen 48. Habilitando el *site* de ssl

7. Ahora, reiniciamos el servicio de Apache2.

8. Si queremos comprobar el funcionamiento, escribimos alguna de las anteriores direcciones (las de apartados siguientes), que sepamos que funcionan de manera correcta y cambiamos *http* por *https*.

**PARA TENER EN CUENTA...**

Hay ocasiones en las que, dependiendo de la versión del navegador o de su funcionamiento, podemos encontrarnos con que la conexión segura nos muestra un error.

www.universae.com