

# DESPLIEGUE DE APLICACIONES WEB

## Tema 2. Configuración y administración de Servidores Web

- 1.- Funcionamiento de un servidor Web
  - 1.1.- Servicio de ficheros estáticos
  - 1.2.- Contenido dinámico
  - 1.3.- Protocolo HTTP y HTTPS
  - 1.4.- Tipos MIME
    - 1.4.1.- Configurar el servidor para enviar los tipos MIME correctos.
- 2.- Hosts virtuales. Creación, configuración y utilización
  - 2.1.- Virtualhosts basados en nombre
  - 2.2.- Virtualhosts basados en IP
  - 2.3.- Virtualhosts basados en varios servidores principales
- 3.- Módulos
  - 3.1.- Operaciones sobre módulos
- 4.- Acceso a carpetas seguras
  - 4.1.- Certificados digitales, AC y PKI
  - 4.2.- Módulo ssl para apache
  - 4.3.- Crear un servidor virtual seguro en Apache (I)
    - 4.3.1.- Crear un servidor virtual seguro en Apache (II)
    - 4.3.2.- Crear un servidor virtual seguro en Apache (III)
  - 4.4.- Comprobar el acceso seguro al servidor
- 5.- Autenticación y control de acceso
  - 5.1.- Autenticar usuarios en apache mediante LDAP
- 6.- Monitorización del acceso: Archivos de registro (logs)
  - 6.1.- Directivas para archivos de registro
  - 6.2.- Rotación de los archivos de registro (I)
    - 6.2.1.- Rotación de los archivos de registro (II)
- 7.- Despliegue de aplicaciones sobre servidores Web

# 1.– Funcionamiento de un Servidor Web.

¿Alguna vez te has parado a pensar qué existe detrás de una página web? ¿Por qué al escribir un nombre en un navegador puedes visionar una página web? ¿Por qué no tienes acceso a determinadas páginas? ¿De qué modo puedes impedir el acceso a determinados sitios de una página: por directorio, por usuario? ¿Cómo se puede establecer una comunicación segura en una transición bancaria?

Hoy en día utilizamos Internet como una herramienta común: para el trabajo, para el ocio... Pero sin duda el elemento fundamental que usamos no es otro que el navegador, gracias al cual podemos sacar partido a todo lo que se encuentra en Internet: comprar entradas para el cine, acceder a nuestra cuenta bancaria, averiguar el tiempo que hará el fin de semana... pero nada de esto tendría sentido si detrás de cada página web a la que accedemos no existiera un servidor web, el cual permite que la página esté accesible 24x7 (24 horas al día y 7 días a la semana, es decir, siempre).

La configuración del servidor web dependerá de las páginas web que ofrezca, así la configuración no será la misma si la página posee contenido estático o no, o si se necesita que modifique el contenido según la interacción del usuario, o si se necesita de comunicación segura en la transición de información, o si se debe tener en cuenta el control de acceso a determinados sitios de la página. Por lo tanto según las páginas web que se ofrezcan el servidor web deberá estar configurado para tal fin: con soporte PHP, con soporte de cifrado, con soporte de control de acceso, etc.

¿Un Servidor Web pueda alojar varias páginas web o solamente una? ¿puede alojar varios sitios (*Conjunto de páginas web*), *dominios de Internet* (Nombre por el cual se reconoce a un grupo de dispositivos o equipos conectados a la red. Éstos pueden ser nombres locales, no existentes en Internet, pero son mayoritariamente utilizados para su uso en Internet, por ejemplo: *debian.org*) O solamente uno, esto es, *permite hosts virtuales* (Dominios independientes que se pueden alojar en un mismo servidor web)?

*Un servidor web puede alojar varias páginas, sitios, dominios de Internet, pero hay que tener en cuenta que la elección del servidor web será muy importante para la configuración y administración de uno o múltiples sitios.*

# 1.– Funcionamiento de un Servidor Web.

También tenemos que pensar que todo puede crecer y lo que ahora era un servidor web que ofrecía  $x$  número de páginas necesitamos que ofrezca  $x^*y$ , con lo cual tenemos que prever la escalabilidad del servidor web, y también la estabilidad: ¿cómo se comporta ante múltiples conexiones simultáneas?

De nada servirá tener instalado un servidor web sin saber cómo se va a comportar ofreciendo el servicio, con lo cual será muy importante previamente y durante el funcionamiento del servidor establecer unas pruebas de funcionamiento del mismo y registrar lo acontecido.

Por todo lo anteriormente comentado, en este tema veremos cómo configurar y administrar el servidor Apache (apache2), que podrá soportar: páginas web estáticas, dinámicas, hosts virtuales, seguridad mediante cifrado, autenticación y control de acceso, modularización y monitorización de archivos de registro.

# 1.1 – Servicio de Ficheros Estáticos.

Si al acceder a una página web no es necesaria la intervención de código en el lado del servidor o en el lado del cliente entenderemos que la página es estática, si por el contrario es necesaria la intervención en el lado del servidor y/o en el lado del cliente entenderemos que la página es dinámica.

Ofrecer páginas estáticas es simple, puesto que solamente se necesita que el servidor web disponga de soporte html/xhtml/css o incluso solamente html/xhtml.

En cuanto a configuración y administración del servidor es el caso más simple: solamente se necesita un soporte mínimo base de instalación del servidor Apache, no se necesita por ejemplo soporte PHP.

En cuanto a rendimiento del servidor, es el caso más beneficioso: no necesita de ejecución de código en el lado del servidor para visionar la página y tampoco necesita ejecución de código en el lado del cliente. Lo que significa menos coste de CPU y memoria en el servidor y en el cliente, y por lo tanto una mayor rapidez en el acceso a la información de la página.

Para poder ofrecer páginas estáticas mediante el servidor Apache simplemente copias la página en la ruta correspondiente donde quieres que se visiona la página.

## Rutas de interés en la instalación de Apache (apache2)

- /etc/apache2/
  - apache2.conf
  - conf.d
  - envvars
  - httpd.conf
  - magic
  - mods-available
  - mods-enabled
  - ports.conf
  - sites-available
  - sites-enabled

- /etc/apache2/sites-available/

- default
  - default-ssl

- /var/www/html/

- index.html

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

- /etc/apache2/apache2.conf

# 1.1 – Servicio de Ficheros Estáticos.

En la instalación de Apache se crea una página web en **/var/www/html/index.html** referenciada a través del archivo **000-default.conf** (/etc/apache/sites-available/000-default.conf), éste contiene la configuración por defecto, generada en la instalación de Apache, para esa página.

Si solamente quieres servir una página web la forma más fácil de hacerlo sería sustituyendo la página **index.html**, referenciada en **default**, por la página que quieres servir, por ejemplo **empresa.html**. Puedes comprobarlo siguiendo el procedimiento:

1. Abres el navegador en la página por defecto creada en la instalación de Apache: index.html
2. Sustituyes los archivos en el servidor. Ten en cuenta que la página a servir debe siempre poseer el nombre index.html
3. Pulsas F5 en el navegador para actualizar la página y la página que verás será la tuya.

Si lo que quieres es servir otra página, por ejemplo **empresa.html**, simplemente no le cambies como antes el nombre, deja el que posee la página. Ahora podrás ver dos páginas en el servidor: la página **index.html** y la página **empresa.html**.

Si lo que quieres es servir más páginas pues, como antes, simplemente vas subiendo al servidor las páginas e incluso podrías organizarlas en carpetas.

## 1.2– Contenido Dinámico.

Muchas veces seguro que te encuentras visitando una página web y la información te parece tan interesante que procedes y guardas en **Favoritos** la dirección URL para una posterior visión, pero cuando de nuevo deseas ver la página resulta que lo que estás viendo no tiene nada que ver o es distinto de lo que esperabas,

¿qué ha ocurrido? Pues puede que la página haya cambiado su contenido o que la página que visitas posee contenido no estático, dinámico, dependiente del código ejecutado en el servidor o en el cliente al acceder a la página.

Imagínate que accedes a una página web y dependiendo si posees una cuenta de usuario u otra el contenido es distinto, o que presionas en una imagen de la página y se produce un efecto en la misma, o que el contenido cambia dependiendo del navegador. De cualquier forma la página ha sido modificada mediante una interacción con el usuario y/o el navegador, por lo tanto nos encontramos con una página dinámica.

Una página dinámica necesita más recursos del servidor web que una página estática, ya que consume más tiempo de CPU y más memoria que una página estática.

Además la configuración y administración del servidor web será más compleja: cuántos más módulos tengamos que soportar, más tendremos que configurar y actualizar.

Esto también tendrá una gran repercusión en la seguridad del servidor web: cuántos más módulos más posibilidades de problemas de seguridad.

Algunos módulos con los que trabaja el servidor web Apache para poder soportar páginas dinámicas son: **mod\_actions, mod\_cgi, mod\_cgid, mod\_ext\_filter, mod\_include, mod\_ldap, mod\_perl, mod\_php5, mod\_python.**

En el siguiente enlace a la página de Apache puedes ampliar la información que te proporcionamos sobre los módulos.  
**<http://httpd.apache.org/docs/2.2/es/mod>**

**Tarea 0:** Busca en la web dada de Apache los módulos mencionados en la diapositiva y redacta un documento con la traducción al Español de la descripción de cada módulo. Evita el uso de traductores on-line.

# 1.3– Protocolo HTTP y HTTPS.

¿Quieres conservar la información de forma confidencial? ¿Quieres transferir información de forma segura? Si estás pensando en este tipo de preguntas necesariamente estás pensando en el protocolo HTTPS y no en el protocolo HTTP.

El **protocolo HTTPS** permite que la información viaje de forma segura entre el cliente y el servidor, por la contra el **protocolo HTTP** envía la información en texto claro, es decir, cualquiera que accediese a la información transferida entre el cliente y el servidor puede ver el contenido exacto y textual de la información.

Para asegurar la información, el protocolo HTTPS requiere de certificados y siempre y cuando sean validados, la información será transferida cifrada. Pero cifrar la información requiere un tiempo de computación, por lo que se verá perjudicado el rendimiento del servidor web.

**¿Es necesario que toda la información sea transferida entre el cliente y servidor de forma cifrada?** A lo mejor solamente es necesario que sea cifrada la autenticación a dicha información, por eso en algunas páginas web puede que el servidor esté configurado para que en todo el dominio esté cifrada su información o simplemente el intento de acceso a la misma.

Un servidor web, como Apache, puede emitir certificados, pero puede que en algún navegador sea interpretado como peligroso, esto suele ser debido a que los navegadores poseen en su configuración una lista de Entidades Certificadoras que verifican, autentican y dan validez a los certificados.

¿Tú, confiarías en un DNI que no fuese certificado por una entidad de confianza como el Ministerio del Interior? Pues, lo mismo le pasa a los navegadores, solamente confían en quien confían. Eso no quiere decir que no puedes crear tus certificados en un servidor web, de hecho muchas empresas lo hacen, sobre todo para sitios internos o externos en los que solamente puede acceder personal autorizado por la propia empresa.

**Tarea 1:** Elabora un documento académico explicando los Protocolos HTTP y HTTPS,

## 1.3– Protocolo HTTP y HTTPS.

El protocolo HTTPS utiliza cifrado sobre SSL/TLS (*protocolos criptográficos que proporcionan comunicaciones seguras por una red, comúnmente Internet*) *que proporcionan autenticación y privacidad. Entonces, si necesitas que la información viaje cifrada debes emplear el protocolo HTTPS, en caso contrario el protocolo HTTP.*

Hay que dejar claro que la utilización del protocolo HTTPS no excluye ni impide el protocolo HTTP, los dos pueden convivir en un mismo dominio.

Bien, pero, ¿cómo funcionan?

**En el protocolo HTTP** cuando escribes una dirección URL en el navegador, por ejemplo **`http://www.debian.org/index.es.html`**, antes de ver la página en el navegador existe todo un juego de protocolos, sin profundizar en todos ellos básicamente lo que ocurre es lo siguiente:

Se traduce el dominio DNS *por una IP, una vez obtenida la IP se busca en ella si un servidor web aloja la página solicitada en el puerto 80 , puerto TCP asignado por defecto al protocolo HTTP. Si el servidor web aloja la página ésta será transferida a tu navegador.*

Sin embargo cuando escribes en el navegador una dirección URL **con llamada al protocolo HTTPS**, el procedimiento es similar al anterior pero un poco más complejo:

Se traduce el dominio DNS por una IP, con la IP se busca el servidor web que aloja la página solicitada en el puerto 443, puerto TCP asignado por defecto al protocolo HTTPS, pero ahora antes de transferir la página a tu navegador se inicia una negociación SSL, en la que entre otras cosas el servidor envía su certificado -el navegador aunque es poco habitual también puede enviar el suyo-. Si el certificado es firmado por un Entidad Certificadora de confianza se acepta el certificado y se cifra la comunicación con él, transfiriendo así la página web de forma cifrada.



## 1.4– Tipos MIME.

¿Cómo se transmite un vídeo por Internet, con qué codificación? ¿Cómo sabe un navegador que al seguir un enlace de vídeo el programa que debe utilizar para reproducirlo?

El estándar Extensiones Multipropósito de Correo de Internet o MIME (Multipurpose Internet Mail Extensions), especifica como un programa debe transferir archivos de texto, imagen, audio, vídeo o cualquier archivo que no esté codificado en US-ASCII.

**MIME** está especificado en seis RFC (***Request for Comments***. *Serie de documentos en los que se detalla prácticamente todo lo relacionado con la tecnología de la que se sirve Internet: protocolos, recomendaciones, comunicaciones...*) :

RFC2045	<a href="http://tools.ietf.org/html/rfc2045">http://tools.ietf.org/html/rfc2045</a>
RFC 2046	<a href="http://tools.ietf.org/html/rfc2046">http://tools.ietf.org/html/rfc2046</a>
RFC 2047	<a href="http://tools.ietf.org/html/rfc2047">http://tools.ietf.org/html/rfc2047</a>
RFC 4288	<a href="http://tools.ietf.org/html/rfc4288">http://tools.ietf.org/html/rfc4288</a>
RFC4289	<a href="http://tools.ietf.org/html/rfc4289">http://tools.ietf.org/html/rfc4289</a>
RFC2077	<a href="http://tools.ietf.org/html/rfc2077">http://tools.ietf.org/html/rfc2077</a>

¿Cómo funciona? Imagínate el siguiente ejemplo: Transferencia de una página web.

Cuando un navegador intenta abrir un archivo, el estándar MIME le permite saber con qué tipo de archivo está trabajando para que el programa asociado pueda abrirlo correctamente. Si el archivo no tiene un tipo MIME especificado, el programa asociado puede suponer el tipo de archivo mediante la extensión del mismo, por ejemplo: un archivo con extensión **.txt** supone contener un archivo de texto.

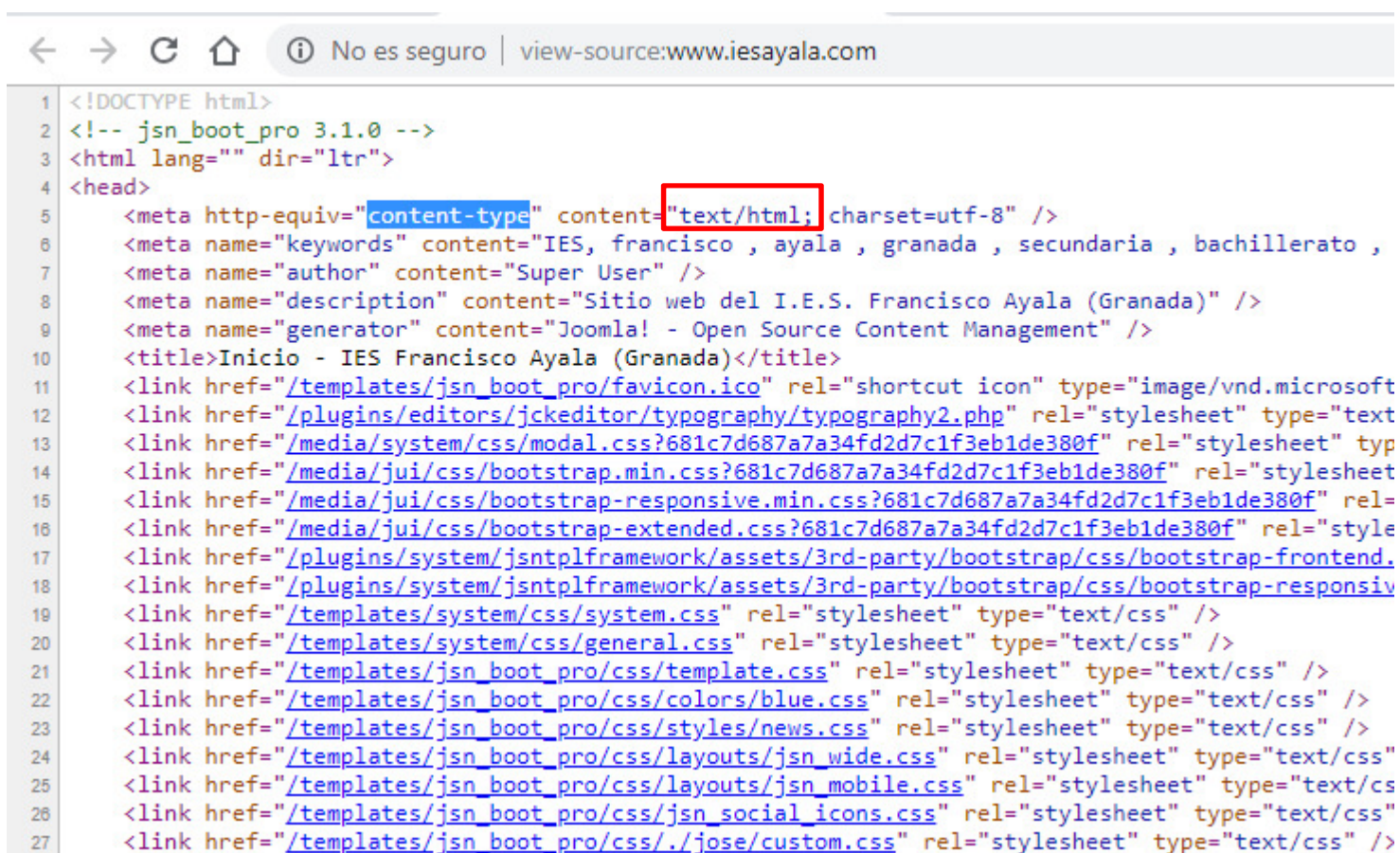
Bien, pero ¿cómo lo hace?

El navegador solicita la página web y el servidor, antes de transferirla, confirma que la petición requerida existe y el tipo de datos que contiene. Esto último, mediante referencia al tipo MIME al que corresponde. Este diálogo, oculto al usuario, es parte de las cabeceras HTTP

## 1.4– Tipos MIME.

En ese diálogo, en las cabeceras respuestas del servidor existe el campo **Content-Type**, donde el servidor avisa del tipo MIME de la página. Con esta información, el navegador sabe cómo debe presentar los datos que recibe.

Por ejemplo cuando visitas <http://www.iesayala.com>, puedes ver como respuesta en la cabecera del servidor el campo **Content-Type: text/html**, indicando que el contenido de la página web es tipo texto/html:



```
1 <!DOCTYPE html>
2 <!-- jsn_boot_pro 3.1.0 -->
3 <html lang="" dir="ltr">
4 <head>
5     <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6     <meta name="keywords" content="IES, francisco , ayala , granada , secundaria , bachillerato ,
7     <meta name="author" content="Super User" />
8     <meta name="description" content="Sitio web del I.E.S. Francisco Ayala (Granada)" />
9     <meta name="generator" content="Joomla! - Open Source Content Management" />
10    <title>Inicio - IES Francisco Ayala (Granada)</title>
11    <link href="/templates/jsn_boot_pro/favicon.ico" rel="shortcut icon" type="image/vnd.microsoft
12    <link href="/plugins/editors/jckeditor/typography/typography2.php" rel="stylesheet" type="text
13    <link href="/media/system/css/modal.css?681c7d687a7a34fd2d7c1f3eb1de380f" rel="stylesheet" typ
14    <link href="/media/jui/css/bootstrap.min.css?681c7d687a7a34fd2d7c1f3eb1de380f" rel="stylesheet
15    <link href="/media/jui/css/bootstrap-responsive.min.css?681c7d687a7a34fd2d7c1f3eb1de380f" rel=
16    <link href="/media/jui/css/bootstrap-extended.css?681c7d687a7a34fd2d7c1f3eb1de380f" rel="style
17    <link href="/plugins/system/jsn_tplframework/assets/3rd-party/bootstrap/css/bootstrap-frontend.
18    <link href="/plugins/system/jsn_tplframework/assets/3rd-party/bootstrap/css/bootstrap-responsiv
19    <link href="/templates/system/css/system.css" rel="stylesheet" type="text/css" />
20    <link href="/templates/system/css/general.css" rel="stylesheet" type="text/css" />
21    <link href="/templates/jsn_boot_pro/css/template.css" rel="stylesheet" type="text/css" />
22    <link href="/templates/jsn_boot_pro/css/colors/blue.css" rel="stylesheet" type="text/css" />
23    <link href="/templates/jsn_boot_pro/css/styles/news.css" rel="stylesheet" type="text/css" />
24    <link href="/templates/jsn_boot_pro/css/layouts/jsn_wide.css" rel="stylesheet" type="text/css" />
25    <link href="/templates/jsn_boot_pro/css/layouts/jsn_mobile.css" rel="stylesheet" type="text/cs
26    <link href="/templates/jsn_boot_pro/css/jsn_social_icons.css" rel="stylesheet" type="text/css" />
27    <link href="/templates/jsn_boot_pro/css/./jose/custom.css" rel="stylesheet" type="text/css" />
```

## 1.4– Tipos MIME.

Cada identificador de tipo MIME consta de dos partes. La primera parte indica la categoría general a la que pertenece el archivo como, por ejemplo, **"text"**. La segunda parte del identificador detalla el tipo de archivo específico como, por ejemplo, **"html"**. Un identificador de tipo **MIME "text/html"**, por ejemplo, indica que el archivo es una página web estándar.

Los tipos MIME pueden indicarse en tres lugares distintos: el servidor web, la propia página web y el navegador.

- El servidor debe estar capacitado y habilitado para manejar diversos tipos MIME.
- En el código de la página web se referencia tipos MIME constantemente en etiquetas link, script, object, form, meta, así por ejemplo:
  - El enlace a un archivo hoja de estilo CSS: `<link href="/miarchivo.css" rel="stylesheet" type="text/css">`
  - El enlace a un archivo código javascript: `<script language="JavaScript" type="text/javascript" src="scripts/mijavascript.js">`
- Con las etiquetas meta podemos hacer que la página participe en el diálogo servidor-cliente, especificando datos MIME: `<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">`
- El navegador del cliente también participa, además de estar capacitado para interpretar el concreto tipo MIME que el servidor le envía, también puede, en el diálogo previo al envío de datos, informar que tipos MIME puede aceptar la cabecera `http_accept`, así por ejemplo una cabecera `http_accept` tipo de un navegador sería:

`text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8`

(El valor `/*` significa que el navegador aceptará cualquier tipo MIME)

## 1.4.1 – Configurar el Servidor para enviar los tipos MIME correctos.

En un servidor web podemos especificar el tipo MIME por defecto para aquellos archivos que el servidor no pueda identificar automáticamente como pertenecientes a un tipo concreto, esto es, para aquellos los cuales no se resuelven según su extensión.

Para el servidor web Apache se utilizan dos directivas: **DefaultType** y **ForceType**.

-**DefaultType** asigna la cabecera Content-Type a cualquier archivo cuya MIME no pueda determinarse desde la extensión del archivo.

-**ForceType** hace que todos los ficheros cuyos nombres tengan una equivalencia con lo que se especifique sean servidos como contenido del tipo MIME que se establezca.

### Ejemplos:

**DefaultType text/plain** : Esto significa que cuando el navegador web solicita y recibe ese archivo como respuesta, desplegará el contenido como un archivo de texto.

**DefaultType text/html** : Desplegará el contenido como un archivo HTML.

**ForceType image/gif** : Desplegará el contenido como un archivo de imagen gif.

**ForceType video/mp4** : Desplegará el contenido como un archivo de vídeo mp4.

En el servidor web Apache existe el archivo **/etc/apache2/mods-available/mime.conf** donde encontrarás una referencia al archivo **/etc/mime.types**, el cual contiene la lista de tipos MIME reconocidos por el servidor.

En el siguiente enlace encontrarás la lista oficial de los tipos MIME:

<http://www.iana.org/assignments/media-types/>

## 2.- Host Virtuales. Configuración, creación y funcionamiento.

Anteriormente hemos visto como poder alojar múltiples páginas web en el servidor web Apache, pero todas pertenecientes al mismo sitio/dominio, es decir, todas pertenecientes a **empresa.com**, entonces, ¿no se puede alojar páginas de distintos dominios en el mismo servidor web?

La respuesta es que si, si se puede, ¿cómo?, mediante la configuración de hostsvirtuales o virtualhosts.

Éstos lo que hacen es permitir que un mismo servidor web pueda alojar múltiples dominios, así configurando hosts virtuales podemos alojar: **empresa1.com** , **empresa2.com**, ..., **empresaN.com** en el mismo servidor web.

Cada empresa tendrá su virtualhost único e independiente de las demás.

Aunque como se ha comentado anteriormente cada virtualhost es único e independiente de los demás, todo aquello que no esté incluido en la definición de cada virtualhost se heredarán de la configuración principal: **apache2.conf** (**/etc/apache2/apache2.conf**).

Si se quiere definir una directiva común en todos los virtualhost no se debe modificar cada uno de los virtualhost introduciendo esa directiva sino que se debe definir esa directiva en la configuración principal del servidor web Apache, de tal forma que todos los virtualhost heredarán esa directiva, por ejemplo en **apache2.conf** puedes encontrar la directiva **Timeout 300**, que establece en 300 el número de segundos antes de que se cancele una conexión por falta de respuesta.

Existen tres tipos de virtualhost: **Basados en Nombre**, **Basados en IP** y **Basados en Varios Servidores Principales**.

## 2.- Host Virtuales. Configuración, creación y funcionamiento.

Al no tener configurado un servidor DNS con las entradas de dominio necesarias, lo primero que configuraremos será el archivo **hosts** de nuestro SO, **/etc/hosts**, añadiéndolas al final del mismo:

```
#EMPRESA 1
127.0.0.1 empresa1.com www.empresa1.com

#EMPRESA 2
127.0.0.1 empresa2.com www.empresa2.com
```

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

Este archivo lo que hace es asignar direcciones IP a los nombres de host, para que al colocar por ejemplo 127.0.0.1 o localhost en nuestro navegador nos redirija al mismo contenido.

El carácter “#” nos permite ingresar un texto a manera de comentario.

Para confirmar su funcionamiento abrimos el navegador y escribimos cualquiera de los nombres de host que se ha agregado.

Si al ingresar el nombre del host le muestra el contenido de <http://localhost> significa que la asignación de IP para el nombre de host a tenido efecto.

## 2.1.– Virtualhosts Basados en Nombre.

La IP que debemos poner siempre en la definición de la directiva Virtualhost es la IP del servidor web, en nuestro escenario:

IP\_Servidor\_Web=127.0.0.1

¿Cómo lo haces? Sigues el procedimiento:

1. En la configuración de Apache2 existe un directorio /etc/apache2/sites-available donde se definen los virtualhosts, cada virtualhost en un fichero de texto de configuración distinto, así crea los dos ficheros siguientes en la ruta **/etc/apache2/sites-available**

2. Fichero configuración virtualhost: **empresa1.conf**

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

3. Fichero configuración virtualhost: **empresa2.conf**

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

**Explicación fichero virtualhost:**

**<VirtualHost IP\_Servidor\_Web:80>** : Inicio etiqueta virtualhost, define la IP del servidor web donde se aloja la página de la empresa.

**DocumentRoot /var/www/empresa1/** : Definición de la ruta donde está alojada la página web en el servidor, en este caso: /var/www/html/empresa1/ mediante la directiva DocumentRoot.

**ServerName www.empresa1.com** : Definición del nombre DNS que buscará la página alojada en la ruta anterior del servidor mediante la directiva ServerName. Es el nombre que escribes en el navegador para visitar la página.

**ServerAlias empresa1.com** : La directiva ServerAlias permite definir otros nombres DNS para la misma página.

## 2.1.– Virtualhosts Basados en Nombre.

Creamos los directorios que alojaran los archivos de las páginas web empresa1 y empresa 2 en **/var/www/html/**

Para poder acceder a las páginas correctamente, conviene dar otros permisos a la carpeta raíz del servidor:

```
sudo chmod -R 755 /var/www/html
```

### Habilitar los nuevos Virtual Host

Para habilitar los nuevos Virtual Host utilizaremos la herramienta a2ensite de Apache:

```
sudo a2ensite empresa1.conf
```

```
sudo a2ensite empresa2.conf
```

Ahora mismo, si accediésemos a **empresa1.com** y **empresa2.com** desde el navegador y los DNS estuvieran bien configurados, veríamos un directorio vacío, por lo que vamos a crear dos archivos **index.html**, uno en cada carpeta para ver si todo funciona correctamente. El archivo de **empresa1.com** será el siguiente:

```
<html>
  <head>
    <title>Empresa 1</title>
  </head>
  <body>
    <h1>Empresa 1 Configurado Correctamente</h1>
  </body>
</html>
```

Y para **empresa2.com**:

```
<html>
  <head>
    <title>Empresa 2</title>
  </head>
  <body>
    <h1>Empresa 2 Configurado Correctamente</h1>
  </body>
</html>
```

Al acceder, ahora, a **empresa1.com** y **empresa2.com** debemos ver las páginas **index.html** creadas.

**Tarea 2:** Crear un Virtualhost basado en nombre para empresa3.com y empresa4.com. Puedes ayudarte de los siguientes enlaces:

<https://www.youtube.com/watch?v=ZDgdSLRNtXY>  
<https://www.youtube.com/watch?v=qPQtD5Ht1Rc>  
<https://geekytheory.com/como-configurar-un-virtual-host-de-apache-en-linux>



## 3.– Módulos.

La importancia de un servidor web radica en su: estabilidad, disponibilidad y escalabilidad. Es muy importante poder dotar al servidor web de nuevas funcionalidades de forma sencilla, así como del mismo modo quitárselas.

Es por esto que la posibilidad que nos otorga el servidor web Apache mediante sus módulos sea uno de los servidores web más manejables y potentes que existen.

Existen dos comandos fundamentales para el funcionamiento de los módulos en el servidor web Apache: **a2enmod** y **a2dismod**.

- **a2enmod**: Utilizado para habilitar un módulo de apache. Sin ningún parámetro preguntará que módulo se desea habilitar. Los ficheros de configuración de los módulos disponibles están en **/etc/apache2/mods-available/** y al habilitarlos se crea un enlace simbólico desde **/etc/apache2/mods-enabled/**

- **a2dismod**: Utilizado para deshabilitar un módulo de Apache. Sin ningún parámetro preguntará que módulo se desea deshabilitar. Los ficheros de configuración de los módulos disponibles están en **/etc/apache2/mods-available/** y al deshabilitarlos se elimina el enlace simbólico desde **/etc/apache2/mods-enabled/**

**a2ensite** es un comando para habilitar configuraciones de "sitios web" en Apache2. Los ficheros de configuración de los "sitios web" disponibles (normalmente son configuraciones de hosts virtuales) están en **/etc/apache2/sites-available/** y al habilitarlos se crea un enlace simbólico desde **/etc/apache2/sites-enabled/**

## 3.1 – Operaciones sobre Módulos.

Los módulos de Apache puedes instalarlos, desinstalarlos, habilitarlos o deshabilitarlos, así, puedes tener un módulo instalado pero no habilitado.

En la tabla siguiente encontrarás un resumen de operaciones, ejemplos y comandos necesarios que se le pueden realizar a los módulos:

Operaciones sobre módulos Apache	
<b>Instalar un módulo</b>	<b>Ejemplo: Instalar el módulo ssl</b>
apt-get install nombre-modulo	apt-get install libapache2-mod-gnutls
<b>Desinstalar un módulo</b>	<b>Ejemplo: Desinstalar el módulo ssl</b>
apt-get remove nombre-modulo	apt-get remove libapache2-mod-gnutls
<b>Habilitar un módulo</b>	<b>Ejemplo: Habilitar el módulo ssl</b>
a2enmod nombre-modulo-apache	a2enmod ssl
<b>Deshabilitar un módulo</b>	<b>Ejemplo: Deshabilitar el módulo ssl</b>
a2dismod nombre-modulo-apache	a2dismod ssl

## 4.– Acceso a Carpetas Seguras.

¿Todas las páginas web que están alojadas en un sitio deben ser accesibles por cualquier usuario? ¿Todas las accesibles deben enviar la información sin cifrar, en texto claro? ¿Es necesario que todo el trasiego de información navegador-servidor viaje cifrado?

Existe la posibilidad de asegurar la información sensible que viaja entre el navegador y el servidor, pero esto repercutirá en un mayor consumo de recursos del servidor, puesto que asegurar la información implica en que ésta debe ser cifrada, lo que significa computación algorítmica.

El cifrado al que nos referimos es el cifrado de clave pública o asimétrico: **clave pública (kpub)** y **clave privada (kpriv)**. La **kpub** interesa publicarla para que llegue a ser conocida por cualquiera, la **kpriv** no interesa que nadie la posea, solo el propietario de la misma. Ambas son necesarias para que la comunicación sea posible, una sin la otra no tiene sentido, así una información cifrada mediante la **kpub** solamente puede ser descifrada mediante la **kpriv** y una información cifrada mediante la **kpriv** solo puede ser descifrada mediante la **kpub**.

En el cifrado asimétrico podemos estar hablando de individuos o de máquinas, en nuestro caso hablamos de máquinas y de flujo de información entre el **navegador (A)** y el **servidor web (B)**.

## 4.- Acceso a Carpetas Seguras.

Funcionamiento del cifrado asimétrico:



$A(\text{inf}) \rightarrow \text{inf cifrada} \rightarrow B [\text{descifrar inf}] \rightarrow B(\text{inf}) = A(\text{inf})$

$A(\text{inf}) \rightarrow \text{inf cifrada} = [(\text{inf})]k_{\text{pub}B} \rightarrow B [\text{inf. cifrada}]k_{\text{priv}B} \rightarrow B(\text{inf}) = A(\text{inf})$

Identificación	
A	Navegador web.
$\text{inf cifrada} = [(\text{inf})]k_{\text{pub}B}$	Información cifrada mediante la clave pública de B obtenida a través de un certificado digital.
$[\text{inf. cifrada}]k_{\text{priv}B}$	Información descifrada mediante la clave privada de B.
B	Servidor web.

## 4.– Acceso a Carpetas Seguras.

Funcionamiento del cifrado asimétrico:

**A** envía la información cifrada mediante la **kpubB** y **B** la descifra mediante su clave privada(**kprivB**), por lo que se garantiza la confidencialidad de la información.

Pero, ¿estás seguro que B es quién dice que es? ¿Es quién debe ser? ¿Cómo garantizas la autenticidad de B?

Pues ya que suponemos que B es quien dice ser mediante un certificado digital, debes confiar en ese certificado, así ¿quién emite certificados digitales de confianza?

Igual que el DNI es emitido por un entidad certificadora de confianza, el Ministerio del Interior, en Internet existen autoridades de certificación (CA ó AC) que aseguran la autenticidad del certificado digital, y así la autenticidad de B, como: [Verisign](#) y [Thawte](#).

Como ya hemos comentado el Servidor Web Apache permite ser CA, por lo que tienes la posibilidad de crear tus propios certificados digitales, ahora bien, ¿el navegador web(A) confiará en estos certificados?

Pues, en principio no, por lo que los navegadores avisarán que la página a la cuál intentas acceder en el servidor web representa un peligro de seguridad, ya que no existe en su lista de autoridades certificadoras de confianza. En determinados casos, puede ser un problema, pero si la empresa posee una entidad de importancia reconocida o el sitio es privado y no público en Internet o sabes el riesgo que corres puedes aceptar la comunicación y el flujo de información viajará cifrado.

## 4.1 – Certificados Digitales, AC Y PKI.

Un certificado digital es un documento electrónico que asocia una clave pública con la identidad de su propietario, individuo o máquina, por ejemplo un servidor web, y es emitido por autoridades en las que pueden confiar los usuarios. Éstas certifican el documento de asociación entre clave pública e identidad de un individuo o máquina (servidor web) firmando dicho documento con su clave privada, mediante firma digital.

La idea consiste en que los **dos extremos de una comunicación, por ejemplo cliente (navegador web) y servidor (servidor web Apache) puedan confiar directamente entre sí, si ambos tienen relación con una tercera parte, que da fe de la fiabilidad de los dos, aunque en la práctica te suele interesar solamente la fiabilidad del servidor, para saber que te conectas con el servidor que quieres y no con otro servidor -supuestamente cuando tú te conectas con el navegador al servidor eres tú y no otra persona la que establece la conexión-**.

Así la necesidad de una **Tercera Parte Confiable (TPC ó TTP, Trusted Third Party)** es fundamental en cualquier entorno de clave pública. La forma en que esa tercera parte avalará que el certificado es de fiar es mediante su firma digital sobre el certificado. Por tanto, podremos confiar en cualquier certificado digital firmado por una tercera parte en la que confiamos. La **TPC** que se encarga de la firma digital de los certificados de los usuarios de un entorno de clave pública se conoce con el nombre de **Autoridad de Certificación (AC)**.

El modelo de confianza basado en **Terceras Partes Confiables** es la base de la definición de las **Infraestructuras de Clave Pública (ICP o PKIs, Public Key Infrastructures)**. Una Infraestructura de Clave Pública es un conjunto de protocolos, servicios y estándares que soportan aplicaciones basadas en criptografía de clave pública.

Algunos de los servicios ofrecidos por una **ICP (PKI)** son los siguientes:

- **Registro de claves:** emisión de un nuevo certificado para una clave pública.
- **Revocación de certificados:** cancelación de un certificado previamente emitido.
- **Evaluación de la confianza:** determinación sobre si un certificado es válido y qué operaciones están permitidas para dicho certificado.
- **Recuperación de claves:** posibilidad de recuperar las claves de un usuario.

## 4.1 – Certificados Digitales, AC Y PKI.

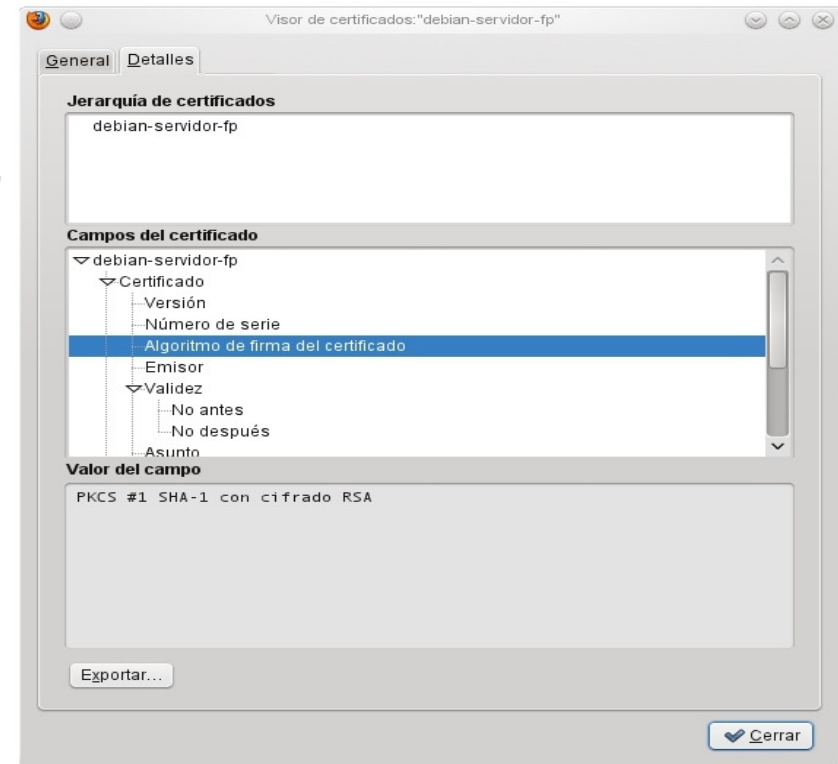
Las **ICP (PKI)** están compuestas por:

-**Autoridad de Certificación (AC)**: realiza la firma de los certificados con su clave privada y gestiona la lista de certificados revocados.

-**Autoridad de Registro (AR)**: es la interfaz hacia el mundo exterior. Recibe las solicitudes de los certificados y revocaciones, comprueba los datos de los sujetos que hacen las peticiones y traslada los certificados y revocaciones a la **AC** para que los firme.

Existen varios formatos para certificados digitales, pero los más comúnmente empleados se rigen por el estándar **UIT-T** (*es la organización de las Naciones Unidas para las tecnologías de la información y la comunicación. En su calidad de coordinador mundial de gobiernos y sector privado, la función de la UIT abarca tres sectores fundamentales, a saber: radiocomunicaciones, normalización y desarrollo*) **X.509**.

El certificado **X.509** contiene los siguientes campos: versión, nº de serie del certificado, identificador del algoritmo de firmado, nombre del emisor, periodo de validez, nombre del sujeto, información de clave pública del sujeto, identificador único del emisor, identificador único del sujeto y extensiones.



## 4.2– Módulo ssl para Apache.



### Reflexiona

Todos los días los bancos efectúan transferencias bancarias, así como también aceptan conexiones a sus páginas web para ofrecer su servicio online. ¿Qué pasaría si cualquiera pudiese interceptar una comunicación bancaria de ese tipo? ¿Sería interesante cifrar la información efectuada antes y durante la conexión bancaria?

El método de cifrado SSL/TLS utiliza un método de cifrado de clave pública (cifrado asimétrico) para la autenticación del servidor.

El **módulo ssl** es quien permite cifrar la información entre navegador y servidor web. En la instalación por defecto éste módulo no viene activado, así que debes ejecutar el siguiente comando para poder activarlo: **sudo a2enmod ssl**

Este módulo proporciona SSL v2/v3 y TLS v1 para el Servidor Apache HTTP; y se basa en Open SSL (*Paquete de herramientas de administración y bibliotecas relacionadas con la criptografía, que suministran funciones criptográficas, entre otros, a navegadores web para acceso seguro a sitios mediante el protocolo HTTPS*) para proporcionar el motor de la criptografía.

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

[http://httpd.apache.org/docs/2.2/es/mod/mod\\_ssl.html](http://httpd.apache.org/docs/2.2/es/mod/mod_ssl.html)

**Tarea 3.** Activar el módulo ssl y crear un servidor seguro en Apache.



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

Apache posee por defecto en su instalación el fichero `/etc/apache2/sites-available/default-ssl.conf`, que contiene la configuración por defecto de SSL.

```
<IfModule mod_ssl.c>
  <VirtualHost _default_:443>
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
    SSLEngine on
    SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem
    SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
    <FilesMatch "\.(cgi|shtml|phtml|php)$">
      SSLOptions +StdEnvVars
    </FilesMatch>
    <Directory /usr/lib/cgi-bin>
      SSLOptions +StdEnvVars
    </Directory>
    BrowserMatch "MSIE [2-6]" \
      nokeepalive ssl-unclean-shutdown \
      downgrade-1.0 force-response-1.0
    BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown
  </VirtualHost>
</IfModule>
```

En su contenido podemos ver las siguientes líneas:

**SSLEngine on** : Activa o desactiva SSL

**SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem** : Certificado digital del propio servidor Apache

**SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key** : Clave privada del servidor Apache.

Esas líneas lo que quieren decir es que Apache **permite conexiones SSL** y posee un certificado digital autofirmado por sí mismo -ya que **Apache actúa como entidad certificadora**-.

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

Ya tenemos configurado nuestro virtualhost para SSL, ahora lo activamos:

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

Y reiniciamos el Servidor Apache:

```
sudo service apache2 restart
```

Cuando se activa el módulo ssl, mediante el comando `a2enmod ssl` permitiste que Apache atienda el protocolo SSL. Así, si ahora lanzas el navegador **Firefox** con la dirección de tu servidor web Apache mediante el protocolo HTTPS, verás una imagen similar a la siguiente:

**https://127.0.0.1**



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

Lo que indica que el certificado digital del servidor no viene firmado por una **AC** contenida en la lista que posee el navegador, sino por el mismo Apache.

Si lo compruebas haciendo clic en **Detalles Técnicos** verás algo similar a:

**127.0.0.1 usa un certificado de seguridad no válido.**

**No se confía en el certificado porque está autofirmado.  
El certificado sólo es válido para debian-servidor-fp.**

**(Código de error: sec\_error\_untrusted\_issuer)**

Ahora tienes dos opciones: Confiar en el certificado o no.

- Si confías haces clic en **Entiendo los riesgos y Añadir excepción...**

Una vez que confías puedes, antes de **Confirmar excepción de seguridad**, ver el contenido del certificado. Si estás de acuerdo la comunicación se establece y la información viaja cifrada.

- Si no confías haces clic en **!Sácame de aquí!**

¿Pero...? Como eres AC puedes firmar certificados e incluso puedes generar también tu propio certificado autofirmado similar al que viene por defecto en Apache.

Hay que tener en cuenta que la negociación SSL es dependiente totalmente de la IP, no del nombre del sitio web, así no puedes servir distintos certificados en una misma IP.

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

El procedimiento para **crear un certificado SSL** Autofirmado para [Apache](#) en un **servidor de Ubuntu** que le permitirá encriptar el tráfico a su servidor lo vamos a ver en el siguiente video:

<https://www.youtube.com/watch?v=E5WFZWj070Y>

1. Instalación del paquete openssl

**sudo apt-get install openssl**

2. Vamos a empezar por la creación de un subdirectorio dentro de las carpetas de configuración de Apache para colocar los archivos de certificado que vamos a estar creando:

**mkdir /etc/apache2/ssl/**

3. Ahora que tenemos un lugar para colocar la llave y el certificado, podemos crear los dos en un solo paso escribiendo lo siguiente en la consola:

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

Los detalles de la configuración son los siguientes:

**openssl** : Se trata de la herramienta de línea de comandos básica proporcionada por [OpenSSL](#) para crear y administrar certificados, llaves, solicitudes de firma , etc.

**req** : Esto especifica un sub comando para la solicitud de [certificate signing request X.509](#) (CSR ). X.509 es un estándar de infraestructura de clave pública que SSL se añade por su clave y certificado administrado. Dado que estamos queriendo crear un nuevo certificado X.509.

**x509**: Esta opción especifica que queremos hacer un archivo de certificado auto firmado en lugar de generar una solicitud de certificado.

**nodes**: Esta opción le dice a OpenSSL que no queremos asegurar nuestro archivo de clave con una contraseña. Tener un archivo con clave protegida por contraseña haría que **Apache se inicie automáticamente**, ya que habría que introducir la contraseña cada vez que se reinicia el servicio.

**days 365**: Esto especifica que el certificado que estamos creando será válida por un año.

**newkey rsa:2048**: Esta opción creará la solicitud de certificado y una clave privada nueva, al mismo tiempo. Esto es necesario ya que nosotros no creamos una clave privada con antelación. El **rsa: 2048** le dice a OpenSSL que genere una clave **RSA** que es de **2048 bits de longitud**.

**keyout**: los nombres de este parámetro, es del archivo de salida para el archivo de clave privada que se está creando.

**out**: Esta opción da nombre al archivo de salida para el certificado que estamos generando.

Cuando se pulse “Enter”, se le pedirá una serie de preguntas.

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

El punto más importante que se solicita es la línea que dice “Nombre común (por ejemplo, el [FQDN](#) del servidor ó su nombre)”. Debe introducir el nombre de dominio que desea asociar con el certificado, o la dirección IP pública del servidor, si no cuenta con un nombre de dominio.

Lo anterior se veria algo como lo siguiente:

**Country Name (2 letter code) [AU]:US**

**State or Province Name (full name) [Some-State]:New York**

**Locality Name (eg, city) []:New York City**

**Organization Name (eg, company) [Internet Widgits Pty Ltd]:Your Company**

**Organizational Unit Name (eg, section) []:Department of Kittens**

**Common Name (e.g. server FQDN or YOUR name) []:your\_domain.com**

**Email Address []:your\_email@domain.com**

4. Ahora que ya tenemos nuestro certificado y la clave disponibles, podemos configurar Apache para que pueda utilizar estos archivos en un archivo de hostvirtual.

## 4.4– Comprobar el acceso seguro al servidor.

A continuación una serie de actuaciones que te servirán para comprobar que el acceso seguro que estableces con el servidor es el esperado:

- Siempre que te conectes mediante SSL a una página web y el certificado no sea admitido, debes ver los campos descriptivos del certificado antes de generar la excepción que te permita visitar la página.

- Debes comprobar en el certificado si la página a la que intentas acceder es la misma que dice el certificado.

- Típicamente en los navegadores, si no está configurado lo contrario, cuando accedes mediante cifrado SSL a una página web puedes ver en algún lugar del mismo un icono: un candado, por lo cual debes verificar su existencia para asegurarte que estás accediendo por https.

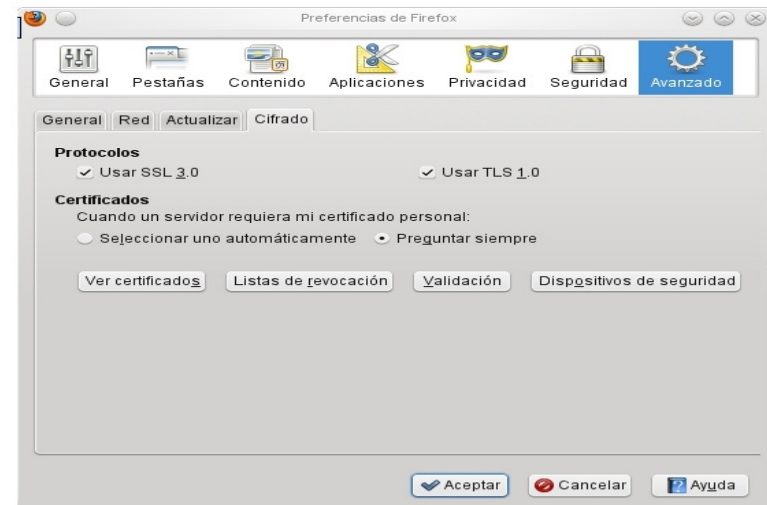
Incluso si el certificado pertenece a alguna AC que el navegador posee en su lista de AC puedes ver en la barra de direcciones indicaciones del tipo de certificado con el que se cifra la comunicación.

- Revisar la lista de certificados admitidos que posee tu navegador. En **Firefox, versión > 3.x**, donde x es el número de revisión de la versión 3, puedes verlas dirigiéndote por las pestañas a:

Editar → Preferencias → Avanzado → Cifrado → Ver certificados

Revisar la lista de revocaciones que posee tu navegador. En **Firefox, versión > 3.x**, donde x es el número de revisión de la versión 3, puedes verlas dirigiéndote por las pestañas a:

Editar → Preferencias → Avanzado → Cifrado → Listas de revocación



Puedes **Importar/Exportar** certificados en los navegadores, con lo cual los puedes llevar a cualquier máquina. Esto es muy útil cuando necesitas un certificado personal en máquinas distintas.

## 4.4– Comprobar el acceso seguro al servidor.

En el siguiente enlace encontrarás unos videos con información muy interesante sobre el protocolo SSL:

**Introducción al Protocolo SSL:**

<https://youtu.be/pOeWmStBOYY>

**Tarea 4:** Resumen del vídeo del **Protocolo SSL**.

**Ataques al Protocolo SSL:**

<https://youtu.be/pSVNnShpCOM>

**Tarea 5:** Resumen del vídeo de **Ataques al Protocolo SSL**.