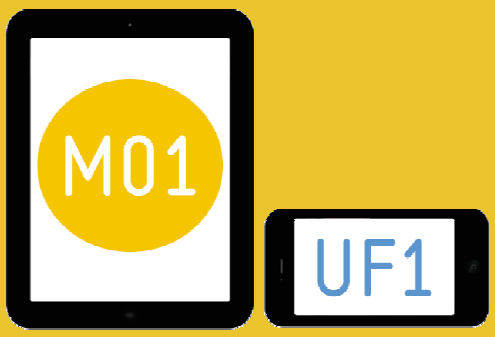
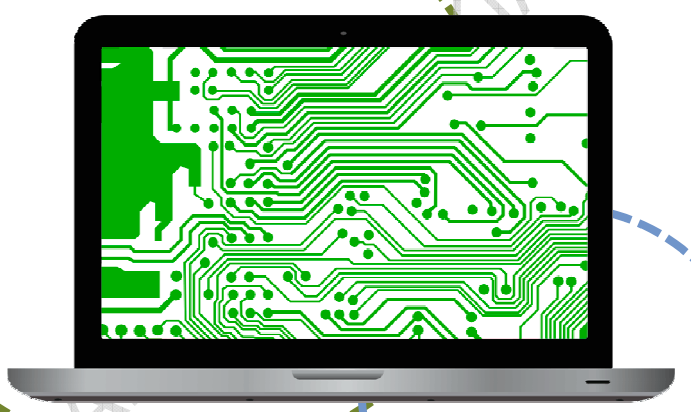
DESARROLLO DE APLICACIONES WEB





Módulo 08. Despliegue de Aplicaciones Web

Silvia Macho
Lidia Porcel

VERSIÓN IMPRIMIBLE A VIAFP

Unidad Formativa 1.
Instalación, configuración y explotación del sistema informático

Linkia FP
Formación Profesional Oficial a Distancia

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP

DESARROLLO DE APLICACIONES WEB

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP

Módulo 08. Despliegue de Aplicaciones Web

Silvia Macho
Lidia Porcel

Unidad Formativa 1.
Servidores web y de transferencia de ficheros

Linkia FP
Formación Profesional Oficial a Distancia

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP

© Macho y Porcel, 2018
© Ediciones Linkia FP, 2018
Calle Pelayo 7, principal 1ª. 08001 Barcelona.
1ª edición: septiembre 2017
Reservados todos los derechos. El material es de uso exclusivo para alumnos de LinkiaFP. No se permite la reproducción total o parcial de esta obra, ni su incorporación a un sistema informático, ni su transmisión en cualquier forma o por cualquier medio (electrónico, mecánico, fotocopia, grabación u otros) sin autorización previa y por escrito de los titulares del copyright. La infracción de dichos derechos puede constituir un delito contra la propiedad intelectual

T1 Tema 1: Arquitectura Web

¿Qué aprenderás?

- Instalar y configurar un servidor web en Windows
- Instalar y configurar un servidor web en Linux

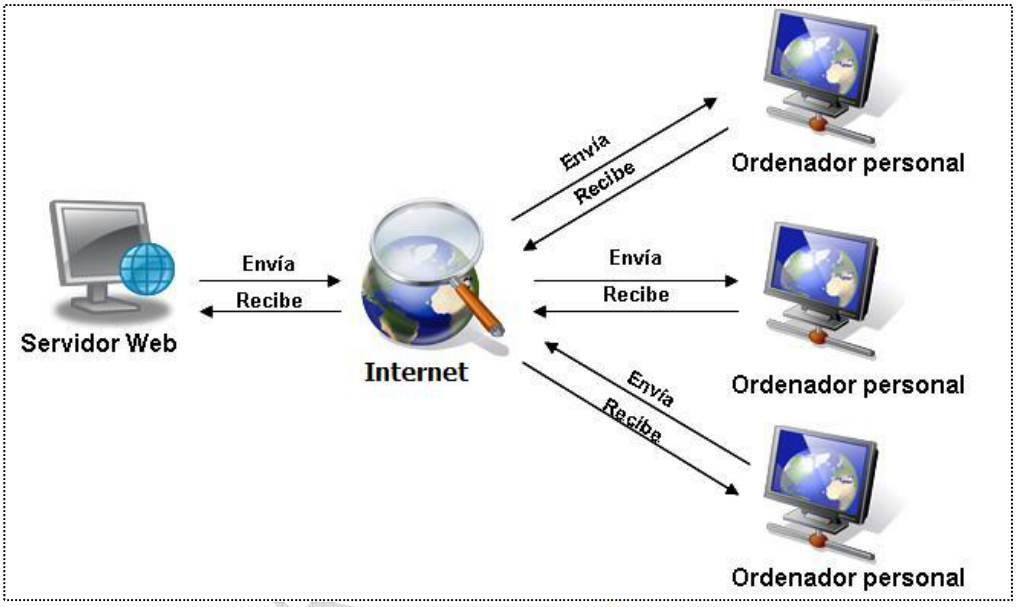
¿Sabías que…?

- W3G es el acrónimo de World Wide Web Consortium.
- El servidor HTTP Apache es un servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.).

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP

1. Introducción

Las páginas web a las que se accede desde los navegadores web, son documentos digitales, típicamente en formato html, que están albergados (almacenados) en diferentes servidores web. Los clientes (ordenadores, dispositivos móviles, etc…) envían peticiones a través de la red, que llegan a los servidores web, los cuales reciben esas peticiones y responden a ellas enviando el contenido solicitado.



Los servidores web eran, en un principio, máquinas tan sólo con un software especializado en enviar dicho contenido web a cada cliente que se lo solicitara, y a través del protocolo http. Hoy en día, sin embargo, las páginas web y su contenido han evolucionado mucho, y el papel del servidor web es más complejo.

1.1.1. Conversión binario-decimal

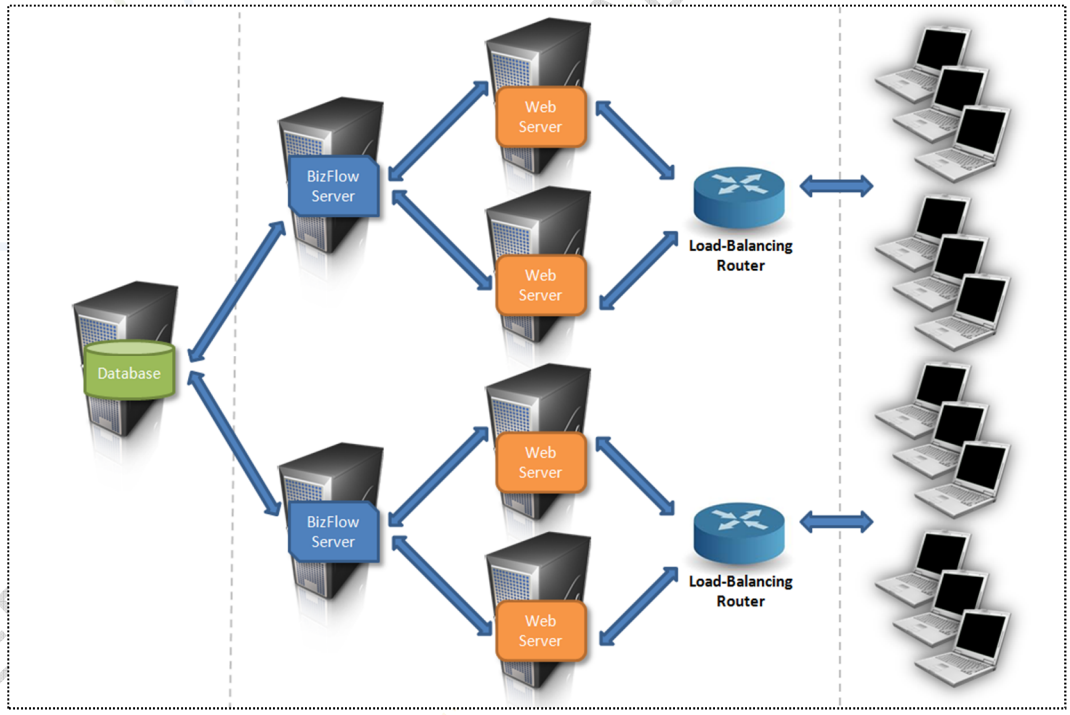
1.1. Servidor de aplicaciones web

El servidor web ya no es tan sólo una máquina encargada de enviar contenido html al cliente a través de http, también debe ejecutar otras acciones tales como: acceder a una base de datos para recoger valores y mostrarlos en una página web, autentificar a los clientes que

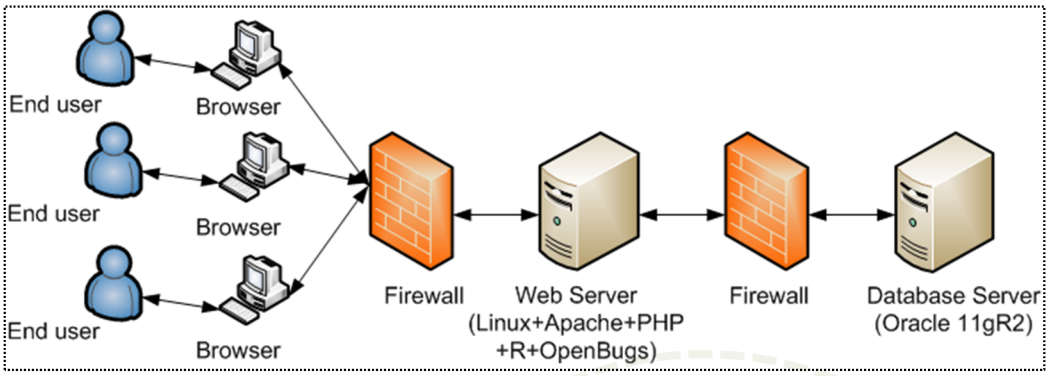
acceden, ejecutar código de diferentes tipos y tecnologías, repartirse las peticiones entrantes entre varios servidores, etc.

Los sitios web de tamaño medio o grande tienen detrás de sí numerosos servidores, ya sean físicos o virtuales, que se reparten las diferentes funciones a realizar. A su vez, una sola función, puede estar repartida entre diferentes servidores, con el objetivo de distribuir la carga de peticiones (soportar más clientes conectados), y de proporcionar alta disponibilidad al servicio (estar siempre operativo aunque un servidor falle). Al estar las diferentes funciones repartidas entre varios servidores se mejora también la escalabilidad del sistema. Esto significa que las ampliaciones del sistema son más sencillas de llevar a cabo si el servicio está distribuido por varios servidores. En ese caso la ampliación consiste principalmente en aumentar el número de servidores.

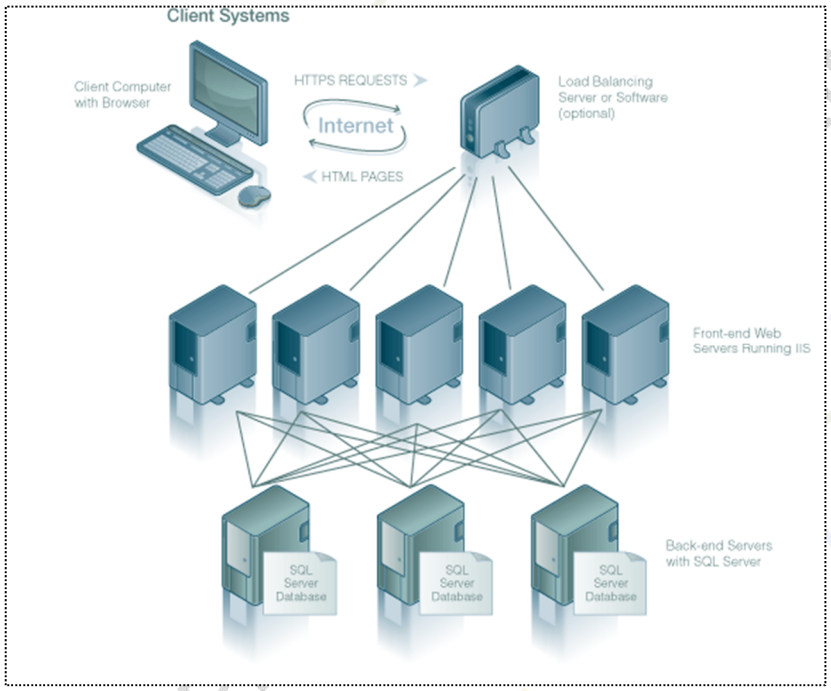
La arquitectura de sistemas resultante puede variar según las necesidades específicas, pero en general, detrás de un sitio web existirán varios servidores interconectados entre sí a través de redes TCP/IP locales de alta velocidad. A continuación se muestran ejemplos de diferentes arquitecturas posibles:



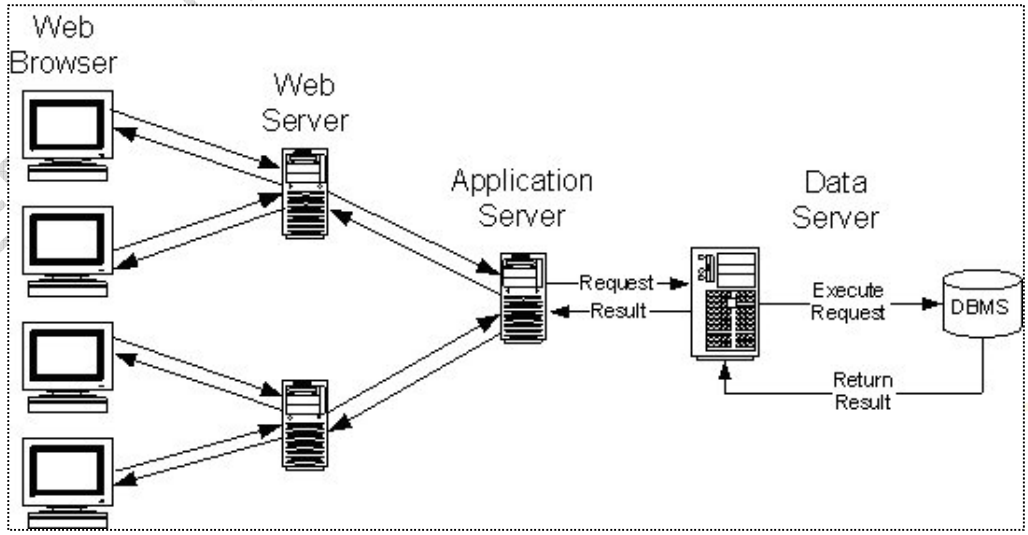
http://bizflow.com/



https://biostatistics.csmc.edu/



http://www.coop-systems.com/



http://weblogs.foxite.com/andykramek/

Como se puede observar en los esquemas anteriores, las posibilidades son muchas y variadas. Dependiendo de las necesidades se deberán implantar diferentes servicios adicionales. En cualquier caso, antes de poner en marcha un sitio web se debe diseñar la arquitectura web que lo va a soportar. Tan importante es hacer un buen diseño y programación del sitio web, como después implantarlo y configurarlo en unos sistemas rápidos, escalables y fiables.

En definitiva, no sólo se debe hablar de un sitio web, sino de un concepto más amplio como el de aplicación web, que implica muchos más componentes.

Un caso típico de implantación en Internet de aplicaciones web se lleva a cabo sobre plataformas LAMP (Linux Apache MySQL PHP), que es una pila (un conjunto) de tecnologías de uso mayoritario. Con el tiempo se ha demostrado que estas tecnologías y sistemas trabajan juntos de forma rápida y fiable, y es por ello que tienen una presencia mayoritaria en Internet.

Para entornos de test, existen paquetes integrados que juntan todo el software necesario para implantar una aplicación web en un solo paquete de software. Ejemplos de este software son los paquetes XAMPP y WAMP. Estos paquetes son útiles para que el desarrollador de la aplicación pueda hacer pruebas ágilmente, incluso en su propio ordenador, pero no sirven como plataforma de funcionamiento de la aplicación. Para desplegar una aplicación en el llamado entorno de "producción" (es decir, atendiendo a los clientes reales), se debe hacer el estudio de la arquitectura necesaria y configurarla con los elementos elegidos.

1.2. Hosting

Para hacer público (o pasar a "producción") un sitio web, o de forma más general, una aplicación web, es necesaria cierta infraestructura fiable, que conste de diversos elementos tales como: una conexión a red rápida, capacidad de proceso suficiente, almacenamiento suficiente, etc.

Uno mismo puede proveerse de todos los elementos de infraestructura necesarios para poner en producción una aplicación. Ello supone comprar el hardware necesario y colocarlo en salas especialmente condicionadas para que se pueda servir la aplicación durante las 24h del día.

Aunque siempre es posible hacerlo todo uno mismo, en la mayoría de casos es más inteligente recurrir a empresas especialistas que ofrecen estos servicios. Estas empresas ofrecen diferentes tipos de servicios que permiten a los clientes alojar sus aplicaciones web

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP

en sus sistemas. Estas empresas disponen de grandes salas de servidores, CPD (Centros de procesos de datos) que después alquilan a clientes de diferentes formas.

Según el grado de control que el cliente tiene sobre sus aplicaciones web, existen diferentes tipos de servicios:

- Hosting compartido: suele ser el más barato. Se ofrece un servidor a varios clientes que lo comparten. Los clientes acceden cada uno a una carpeta de disco donde suben su contenido web. No tienen control sobre el servidor ni sobre su configuración.
- VPS: se dedica una máquina virtual de las características contratadas a cada cliente. El cliente así puede configurar todo el sistema operativo virtual y cualquier software que incorpore.
- Servidor dedicado: se dedica un servidor físico completo de las características contratadas para cada cliente.
- Housing o colocation: se alquila al cliente un espacio físico. Se le proporciona sólo los servicios básicos (electricidad, seguridad física, conectividad). El cliente coloca su propio hardware y lo maneja él mismo a todos los niveles (hardware y software).

En todos los tipos de servicios, estas empresas proveedoras ofrecen servicios adicionales de soporte, para asistir al cliente en las operaciones de mantenimiento e instalación que pueda necesitar.

1.3. Cloud hosting

El éxito rotundo que hoy en día tiene el paradigma del Cloud Computing ha hecho surgir también nuevos tipos de servicios ofrecidos.

Entre ellos, el conocido como Cloud Hosting, en el que las empresas proveedoras ofrecen a los clientes plataformas online donde pueden manejar los recursos contratados con mucha flexibilidad y control.

Las grandes empresas en Internet como Google, Amazon, Microsoft y muchas otras ofrecen plataformas Cloud con muchas posibilidades de configuración por parte de los clientes. Ellas proporcionan, con los grandes CPDs de que disponen distribuidos por el mundo, grandes capacidades de proceso.

El modelo más común de estos servicios suele ser el del pago por uso. Los clientes pagan por el tiempo en que sus máquinas están en ejecución. Cuanto más tiempo estén en producción y mayores sean las características, mayor es el coste.

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP

Una de las grandes ventajas de estas plataformas es su gran flexibilidad. Los clientes pueden preestablecer reglas para que los servicios utilizados se amplíen automáticamente frente a una demanda alta de peticiones, mientras que se puede también automáticamente en caso de que la carga baje.

Uno de los principales ejemplos de estas plataformas Cloud es Amazon Web Services, que es uno de los pioneros. Sin embargo hoy en día existen muchos otros:

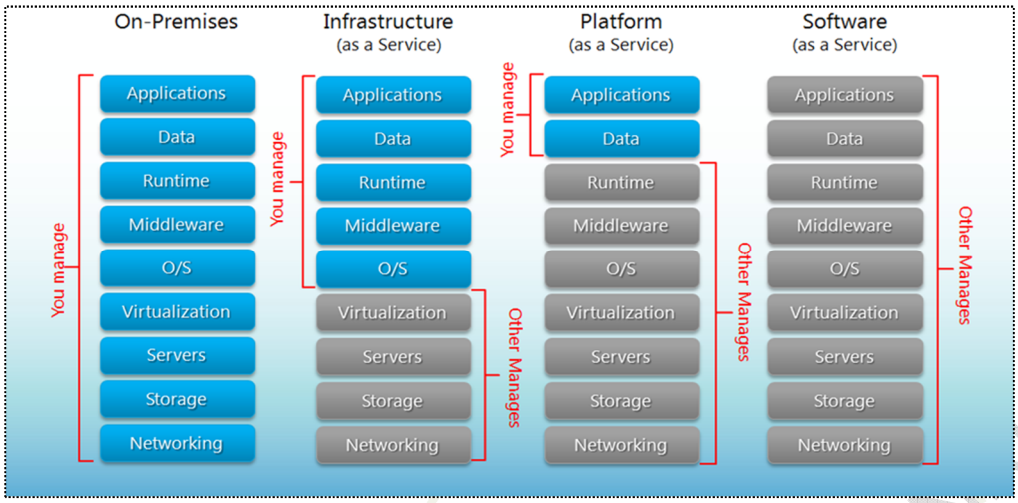
forbes.com

Cada proveedor se centra en diferentes niveles de servicio ofreciendo diferentes posibilidades a los clientes. Los modelos IaaS, PaaS y SaaS acumulan la mayor parte de la oferta de servicios Cloud.

Aunque existen muchas maneras de ofrecer servicios en la nube, existe una clasificación de tres tipos: IaaS, PaaS y SaaS. Esta clasificación diferencia estos servicios de la instalación local de servicios que se haría de forma tradicional, instalando todo el hardware y manejando todos los niveles con recursos propios (On Premises).

En el siguiente gráfico se observan las diferencias:

VERSIÓN IMPRIMIBLE AFP



http://programacion.jias.es/

Los diferentes modelos se diferencian en las capas que se permite controlar a los clientes, y las capas de las que se ocupa el proveedor. El modelo On Premises es el modelo tradicional de una empresa que se compra todo el hardware y dispone de un departamento para implantar todos los servicios en sus propias instalaciones.

Por ejemplo, según el modelo IaaS, el proveedor se encarga del hardware y los sistemas de virtualización y almacenamiento. A partir de ahí, el cliente puede manejar sus sistemas operativos y todas las aplicaciones y servicios que se quiera instalar en ellos. AWS es un ejemplo de este modelo, donde el cliente se puede crear sus máquinas virtuales con muchos sistemas operativos ofertados, se puede conectar a ellas y configurarlas con mucha flexibilidad.

En los modelos PaaS y SaaS el proveedor ofrece servicios más prefabricados y el cliente no tiene control sobre el sistema operativo y sobre las capas superiores.

Empieza a ser necesario por parte de los profesionales del sector conocer el funcionamiento de estas plataformas Cloud, y saber desplegar servicios en ellas, además de todos los conocimientos tradicionales sobre sistemas para soportar sus aplicaciones web

En este dosier se estudiarán las aplicaciones web, poniendo el foco en su implantación y configuración desde el punto de vista de la arquitectura de sistemas. Se estudiarán los sistemas necesarios para implantar aplicaciones web.

VERSIÓN IMPRIMIBLE ALUMNO LINUXIAFP

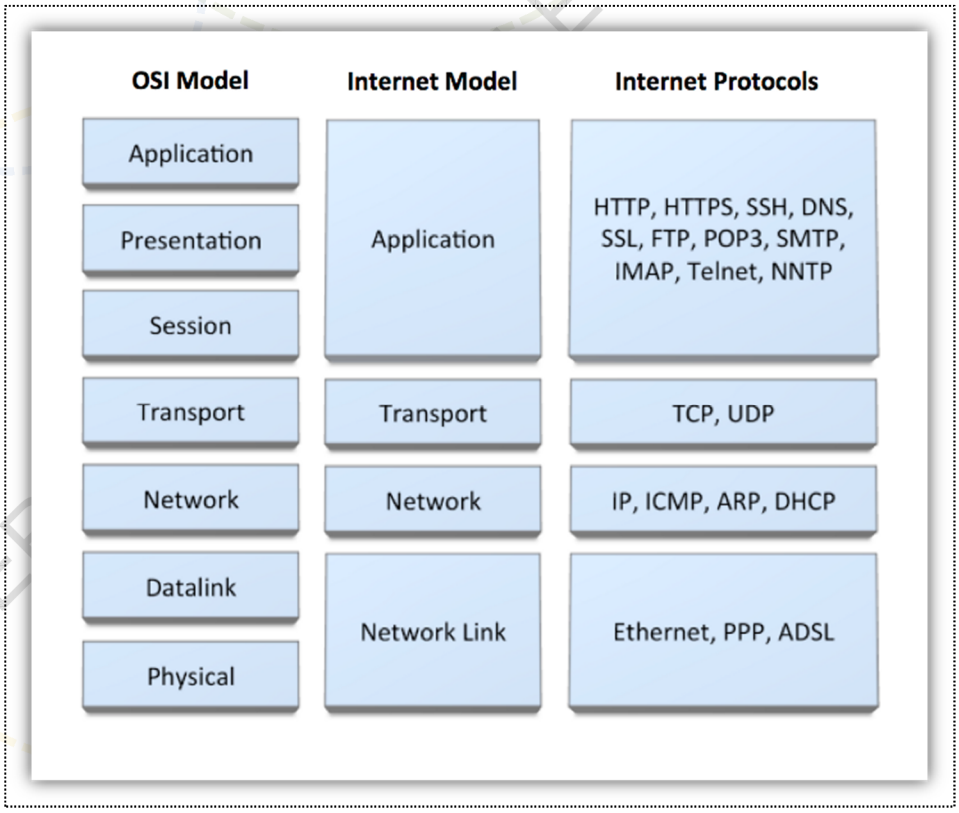
2. Servicios web

La función de servidor web es la principal de entre los servicios involucrados en hacer llegar el contenido web a los clientes. En concreto, se encarga de recepcionar las peticiones HTTP de los clientes, recopilar el contenido solicitado (habitualmente almacenado de forma local) y entregarlo al cliente también mediante el protocolo HTTP.

2.1. Principios básicos de HTTP

HTTP (Hypertext Transfer Protocol) es el protocolo de capa de Aplicación (tanto según el modelo OSI, como según el modelo Internet) encargado de la transmisión de datos de contenido web. HTTP utiliza TCP como protocolo de la capa de transporte para establecer, mantener y finalizar las conexiones entre el servidor web y el cliente (navegador web).

A continuación se ven los dos modelos de capas de protocolos TCP/IP, OSI e Internet, con ejemplos de protocolos asociados a cada capa. Se puede observar como HTTP y HTTPS (http seguro) se encuentran en lo más alto de la pila.



http://vichargrave.com/network-programming-design-patterns-in-c/

HTTP define métodos (son en total menos de diez) para establecer las diferentes acciones a realizar. Ente los métodos más representativos están los siguientes

- GET: Pide una representación (contenido web) del recurso especificado (servidor web). La petición puede ser simple, es decir en una línea o compuesta.
- POST: Envía los datos para que sean procesados por el recurso identificado (servidor web). Los datos se incluirán en el cuerpo de la petición. Esto puede resultar en la creación de un nuevo recurso o de las actualizaciones de los recursos existentes o ambas cosas.
- PUT: Sube, carga o realiza un upload de un recurso especificado (archivo), es el camino más eficiente para subir archivos a un servidor, esto es porque POST utiliza un mensaje multiparte y el mensaje es decodificado por el servidor. En contraste, el método PUT te permite escribir un archivo en una conexión socket establecida con el servidor. La desventaja del método PUT es que los servidores de hosting compartido no lo tienen habilitado.
- OPTIONS: Devuelve los métodos HTTP que el servidor soporta para un URL específico.

A continuación se muestra un ejemplo de comunicación HTTP entre cliente y servidor. En primer lugar, la petición desde el cliente solicitando un contenido.

```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: nombre-cliente
[Línea en blanco]
```

wikipedia.org

Y a continuación la respuesta del servidor, con un encabezado y después el contenido web solicitado.

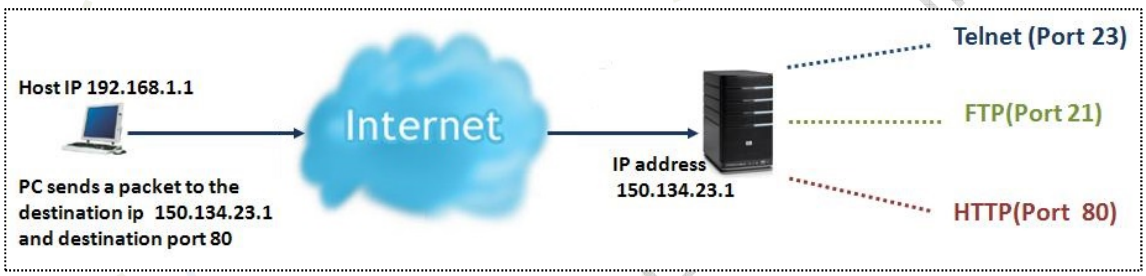
```
HTTP/1.1 200 OK
Date: Fri, 31 Dec 2003 23:59:59 GMT
Content-Type: text/html
Content-Length: 1221

<html>
<body>
<h1>Página principal de tuHost</h1>
(Contenido)
  .
  .
  .
</body>
</html>
```

wikipedia.org

VERSIÓN IMPRIMIBLE ALUMNO LINKAFP

Respecto a la capa de transporte, el servicio web está habitualmente escuchando peticiones en su puerto 80 TCP, que es al que acceden comúnmente todos los navegadores por defecto.

Los puertos de la capa de transporte (protocolos TCP, UDP) son útiles para establecer diferentes conexiones con diferentes propósitos entre clientes y servidores. De alguna manera, un puerto es una puerta de entrada a un servidor al que se llega gracias a su dirección IP. Un mismo servidor, con una sola IP, puede ofrecer varios servicios de forma simultánea gracias al establecimiento de diferentes puertos de escucha. Por ejemplo, por el puerto 80 se puede escuchar a peticiones Web (HTTP), mientras que en la misma máquina, por el puerto 21, se pueden estar esperando peticiones del servicio FTP para la transferencia de ficheros.

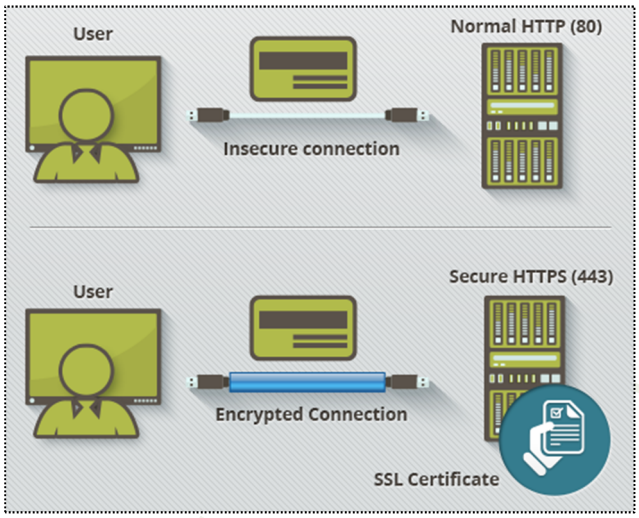


http://ipcisco.com

En la imagen anterior se observa como el cliente, desde su IP de origen (192.168.1.1) lanza una petición http (port 80) a la IP de destino (150.134.23.1) que es la IP del servidor. Cuando el servidor recibe dicha petición, la analiza y detecta que va dirigida al puerto 80, que corresponde al servicio web HTTP. A su vez el servidor es capaz de ofrecer otros servicios (FTP y telnet) asociados a otros puertos diferentes.

Las transacciones http son inseguras ya que alguien con acceso al tráfico existente entre cliente y servidor, podría inspeccionar el contenido sin ninguna traba. Para asegurar la integridad y privacidad de la información transmitida se hace uso del protocolo HTTPS (HTTP sobre TLS), que encripta la información y la firma para que llegue de forma segura a su destinatario. El servicio https escucha de forma estándar en todos los servidores en el puerto 443 TCP.

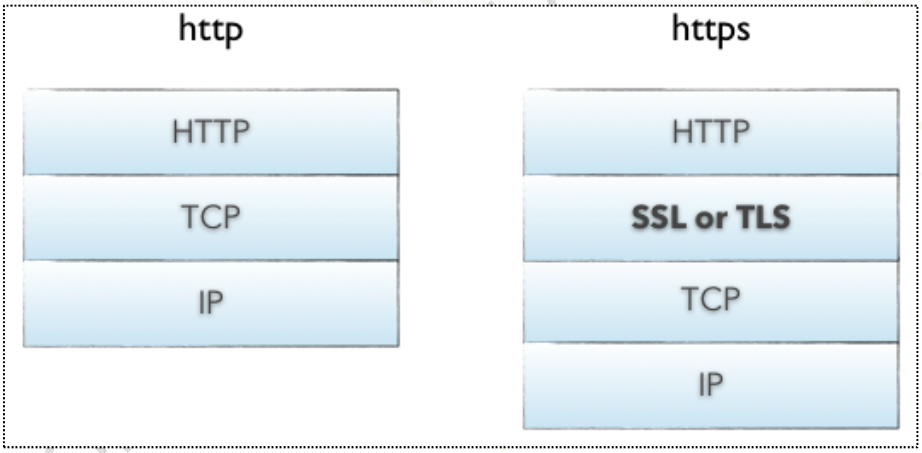
VERSIÓN IMPRIMIBLE AL INKIAP



https://www.instantssl.com/ssl-certificate-products/https.html

El uso del protocolo https implica también el uso de certificados digitales.

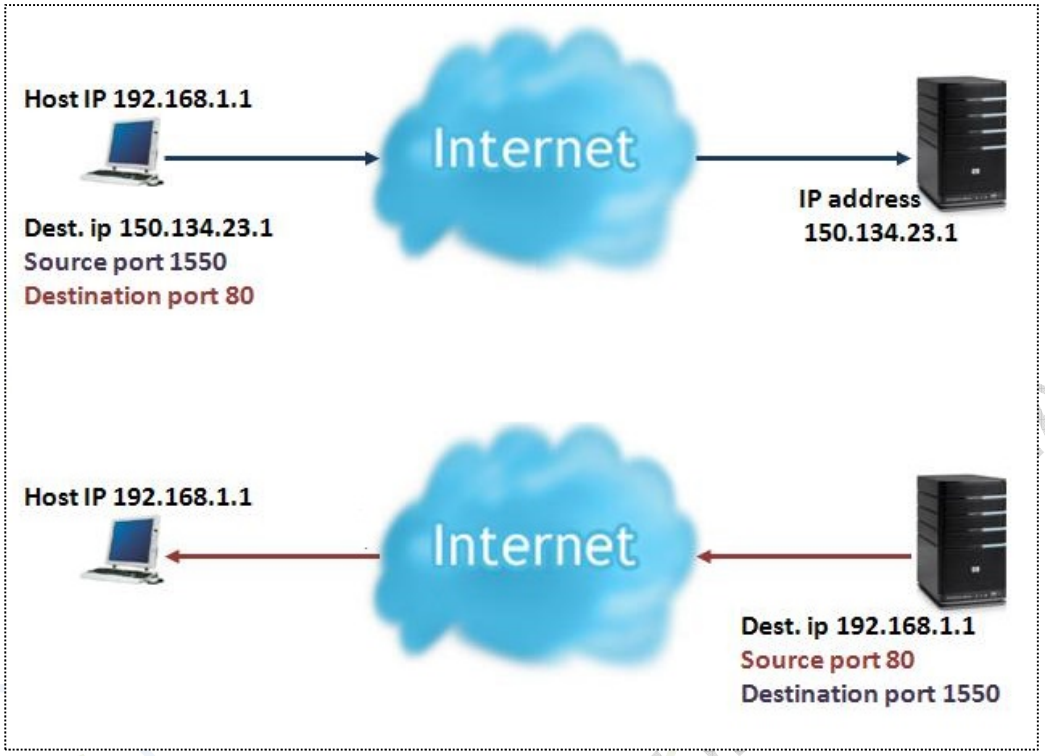
Para asegurar la información se introduce una capa extra en el conocido modelo de capas de protocolos TCP/IP. Si tenemos en cuenta las tres capas de aplicación (http), transporte (tcp) y red (ip), la encriptación sucedería entre las dos primeras. Actualmente el protocolo criptográfico utilizado es TLS, sucesor de SSL.



http://www.ujjwal.com/technical-optimization/https-seo-662/

Para que la comunicación TCP sea totalmente bidireccional y full-duplex (que se pueda transmitir y recibir información de forma simultánea), será necesario que cada host (tanto el cliente como el servidor) dispongan de un puerto de origen (para recibir). En el servidor web, el puerto es el 80 TCP, ya que es un estándar acordado para los servicios web. En el cliente, sin embargo, se abre un puerto de número aleatorio (>1024) para poder recibir las respuestas del servidor por dicho puerto. Ese puerto sólo estará abierto mientras la conexión TCP siga activa.

VERSIÓN ALUMNO LINKIAFP



http://ipcisco.com

En la imagen anterior se pueden observar en primer lugar la petición HTTP, y en segundo lugar la respuesta. La petición va dirigida al puerto de destino 80, que es donde el servidor escucha peticiones HTTP. La petición se hace desde el puerto de origen 1550 (generado aleatoriamente en el cliente).

Cuando el servidor recibe la petición, la procesa, y al responder, utiliza el puerto abierto en el cliente (el 1550) para responder, será en este caso el puerto de destino de la respuesta.

Aunque el estudio a fondo de los protocolos de red implicados en las transacciones de información vía web queda fuera del alcance de estos apuntes, es aconsejable un conocimiento detallado de tales protocolos, ya que resulta de mucha ayuda para la gestión y administración de los servicios web.

2.2. Software de servicios web

Existen múltiples paquetes de software que ejercen la función de servidor web. Entre los más relevantes están:

- *Apache HTTP Server*: es el más utilizado en internet. Su estructura modular le da una gran flexibilidad y la posibilidad de realizar configuraciones avanzadas en entornos complejos. Es un software libre y open-source, actualmente soportado por la Apache Foundation. Es mayoritariamente utilizado en sistemas GNU/Linux, aunque también corre en otros sistemas operativos como Windows.

VERSIÓN IMPRIMIBLE-AEU AFP

- nginx: es un servidor web especializado en el alto rendimiento con un uso de memoria muy bajo. Se suele configurar también como proxy inverso, como balanceador de carga y como cache HTTP. Nginx es libre y open-source, bajo una licencia tipo BSD.
- lighthttp: servidor web optimizado para entornos donde la velocidad de respuesta es crítica. Servidores que deben soportar grandes cargas de peticiones simultáneas suelen funcionar con este software que es open-source.
- Internet Information Service (IIS): servidor web integrado en el sistema operativo Windows Server

2.2.1. Apache HTTP Server

El servidor web HTTP de la Apache Software Foundation es uno de los proyectos de más relevancia de dicha fundación, la cual está muy implicada en el desarrollo de software open-source.

Toda la información técnica relativa al proyecto http está disponible en http://httpd.apache.org/
Aunque Apache http Server puede instalarse en diversos sistemas operativos, en estos apuntes sólo se describirán los procedimientos de instalación y configuración para sistema operativo GNU/Linux, y en concreto en una distribución Ubuntu.

En las siguientes páginas de este documento se hará referencia al "Apache http server" tan sólo con el nombre de "Apache".

2.2.1.1. Instalación y configuración básica

La instalación de Apache en sistemas Ubuntu 18.04 se puede realizar mediante el gestor de paquetes *apt:*

```
linkia@linkia:~$ sudo apt install apache2
```

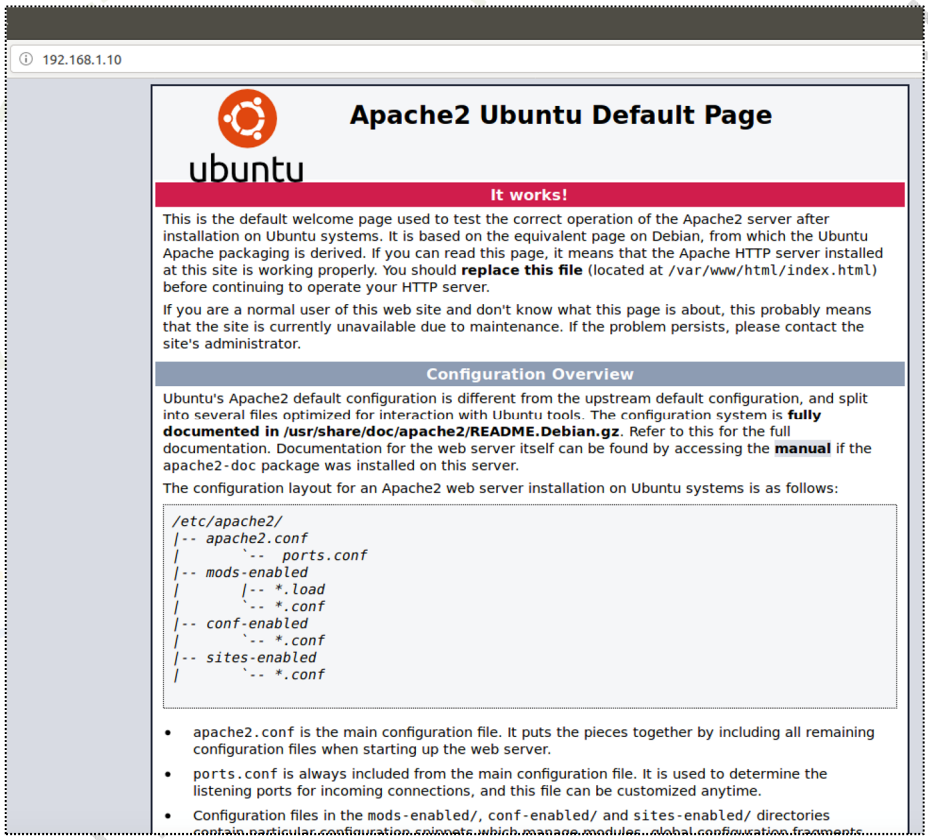
Este comando instalará todos los paquetes necesarios, y realizará una configuración básica inicial y operativa completamente. De hecho, una vez acabado el proceso, el servidor web ya está funcionando y sirviendo el contenido por defecto.

VERSIÓN IMPRIMIBLE - LINKIAFP

Habitualmente es necesario ejecutar previamente el comando *apt update* para actualizar el estado de los repositorios.

```
linkia@linkia:~$ sudo apt update
```

SI la instalación finaliza correctamente se puede comprobar que está todo en marcha desde un cliente con un navegador web, accediendo a la IP del servidor web. Se deberá mostrar la página web de bienvenida instalada por defecto:



La versión que se muestra instalada por apt en esta documentación es la 2.4.29 Se crea una cuenta de usuario, de nombre www-data la cual se asocia al servicio de apache.

La configuración de Apache instalado mediante apt en Ubuntu es específica y no es la misma que si se instala Apache manualmente. Dicha configuración se explica claramente en la página web mostrada por defecto por el servidor. En ella constan los parámetros básicos de configuración del servidor tal como queda instalado. Los puntos principales a tener en cuenta para empezar a configurar el servidor son:
La ubicación del contenido web por defecto está en el disco del servidor, en la ruta */var/www/html* , y en concreto se muestra el archivo de nombre *index.html*.

- La configuración está documentada en

  /usr/share/doc/apache2/README.Debian.gz

- Los archivos de configuración principales están ubicados en */etc/apache2/*

Efectivamente, si se analiza el contenido del disco en la carpeta indicada en el DocumentRoot se encuentra la página de inicio mostrada con el fichero *index.html*

```
linkia@linkia:/etc/apache2/sites-enabled$ cd
linkia@linkia:~$ cd /var/www/html/
linkia@linkia:/var/www/html$ ls
index.html
```

Los ficheros y directorios a tener en cuenta para la configuración del servicio son (algunos ficheros pueden ya no estar presentes según la versión):

- *apache2.conf*: es el archivo principal de configuración. Desde éste se hace referencia a todos los demás.
- *httpd.conf*: en otras distribuciones y con otros métodos de instalación puede ser el archivo principal de configuración. En esta situación (Apache en Ubuntu con instalación mediante *apt*) no tiene relevancia. De hecho ya no está presente en las últimas versiones instaladas en Ubuntu.
- *envvars*: variables de entorno de Apache.
- *ports.conf*: definición de los puertos de escucha del servidor.
- Directorios *mods-available y mods-enabled*: módulos disponibles para la activación en Apache. En el directorio -available están los ficheros de los módulos disponibles, mientras que en el directorio -enabled están los que están activados, y que tan sólo consisten en links que apuntan a los ficheros del directorio –available.

Los llamados sites en apache no son más que ficheros de configuración. Son una manera de separar diferentes configuraciones en diferentes ficheros. A menudo llevan a confusión con el concepto de virtual host que se presenta más adelante en estos apuntes, pero no son lo mismo.
En estos apuntes se trabajará con un solo site, el site por defecto que existe ya una vez instalado apache, su fichero de configuración está en el directorio */etc/apache2/sites-enabled* y el nombre del fichero es *000-default.conf.*

```
linkia@linkia:~$ cd /etc/apache2/sites-enabled/
linkia@linkia:/etc/apache2/sites-enabled$ ls
000-default.conf
```

El contenido de este fichero es el que proporciona la configuración por defecto de Apache, y el que hace visualizar la página web cuando en el servidor está recién instalado el servicio. Editando el fichero del *site default* podemos analizarlo con más detalle. A continuación se muestra dicho contenido:

```
  GNU nano 2.9.3                          000-default.conf

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

Dentro del fichero mostrado se pueden ver diferentes directivas de configuración del servicio, además de comentarios de ayuda para su uso. Las líneas que comienzan con el símbolo "#" son comentarios y no tienen efecto en la configuración.

En el fichero se pueden ver operativas (no comentadas con #) algunas de las directivas de configuración principales, que son:

- VirtualHost: para la creación de un Virtual Host en Apache. Los *Virtual Host* son divisiones que se pueden hacer en un solo servidor Apache, para mostrar diferentes contenidos web con diferentes configuraciones cada uno.
- ServerAdmin: establece la dirección que el servidor incluirá como dirección de contacto en cualquier mensaje de error que genere. Normalmente se configura con la dirección del administrador del sistema.
- DocumentRoot: establece el directorio base a partir del cual se ubicarán los ficheros a mostrar vía web.
- ErrorLog y CustomLog: establecen la ubicación y formato de los ficheros de registro que se generarán con toda la actividad del servidor

VERSIÓN IMPRIMIBLE

Si se desea habilitar *sites* adicionales para disponer de más archivos de configuración separados del fichero de *site default*, se debe utilizar el script a2ensite. Este script habilita el fichero de site que se halle en el directorio *sites-available*, y crea el link correspondiente en el directorio *sites-enabled*.

Existe también un script de configuración general del servicio que se llama *apachectl*. Si se ejecuta con el modificador *-h*, aparecen las diferentes opciones del script:

```
linkia@linkia:/etc/apache2/sites-enabled$ apachectl -h
Usage: /usr/sbin/apache2 [-D name] [-d directory] [-f file]
                         [-C "directive"] [-c "directive"]
                         [-k start|restart|graceful|graceful-stop|stop]
                         [-v] [-V] [-h] [-l] [-L] [-t] [-T] [-S] [-X]
Options:
  -D name            : define a name for use in <IfDefine name> directives
  -d directory       : specify an alternate initial ServerRoot
  -f file            : specify an alternate ServerConfigFile
  -C "directive"     : process directive before reading config files
  -c "directive"     : process directive after reading config files
  -e level           : show startup errors of level (see LogLevel)
  -E file            : log startup errors to file
  -v                 : show version number
  -V                 : show compile settings
  -h                 : list available command line options (this page)
  -l                 : list compiled in modules
  -L                 : list available configuration directives
  -t -D DUMP_VHOSTS  : show parsed vhost settings
  -t -D DUMP_RUN_CFG : show parsed run settings
  -S                 : a synonym for -t -D DUMP_VHOSTS -D DUMP_RUN_CFG
  -t -D DUMP_MODULES : show all loaded modules
  -M                 : a synonym for -t -D DUMP_MODULES
  -t -D DUMP_INCLUDES: show all included configuration files
  -t                 : run syntax check for config files
  -T                 : start without DocumentRoot(s) check
  -X                 : debug mode (only one worker, do not detach)
Action '-h' failed.
The Apache error log may have more information.
linkia@linkia:/etc/apache2/sites-enabled$ _
```

Por ejemplo, con "*apachectl -v*", se puede comprobar la versión de apache instalada:

```
linkia@linkia:/etc/apache2/sites-enabled$ apachectl -v
Server version: Apache/2.4.29 (Ubuntu)
Server built:   2018-10-03T14:41:08
linkia@linkia:/etc/apache2/sites-enabled$ _
```

Cuando se realizan cambios en los ficheros de configuración el servicio se debe reiniciar. Se puede realizar el reinicio de diferentes maneras, con el propio script *apachectl* o con herramientas propias del sistema:

- *apachectl -k restart*
- *service apache2 restart (/etc/init.d/apache2 restart)*

En ambos casos se requieren permisos para ejecutar el reinicio del servicio.

2.2.1.2. Directivas

Apache se configura mediante directivas incluidas en sus ficheros de configuración. Las directivas son palabras clave que configuran diferentes aspectos del servidor web.

La lista de directivas es extensa, y está disponible en la página web de Apache: http://httpd.apache.org/docs/2.4/mod/directives.html.

Es importante entender como está especificada una Directiva en la documentación de Apache. Con la información que se proporciona es suficiente para entender cómo funciona y cómo y dónde se debe escribir. A continuación se observa la ayuda correspondiente a una Directiva de ejemplo (ServerAdmin):

ServerAdmin Directiva

Descripción:	Email address that the server includes in error messages sent to the client
Sintaxis:	`ServerAdmin email-address|URL`
Contexto:	server config, virtual host
Estado:	Core
Módulo:	core

The `ServerAdmin` sets the contact address that the server includes in any error messages it returns to the client. If the `httpd` doesn't recognize the supplied argument as an URL, it assumes, that it's an *email-address* and prepends it with `mailto:` in hyperlink targets. However, it's recommended to actually use an email address, since there are a lot of CGI scripts that make that assumption. If you want to use an URL, it should point to another server under your control. Otherwise users may not be able to contact you in case of errors.

It may be worth setting up a dedicated address for this, e.g.

```
ServerAdmin www-admin@foo.example.com
```

as users do not always mention that they are talking about the server!

De cada directiva se especifica:

- Descripción: breve explicación de lo que hace.
- Sintaxis: cómo se escribe la directiva.
- Contexto: en qué zonas de los ficheros de configuración puede utilizarse.
- Estado: cómo de integrada está la directiva en Apache.
- Módulo: indica el módulo en el que está incluida la directiva

- Por último se hace una explicación más detallada de la directiva, habitualmente con ejemplos incluidos.

En el Anexo I se incluyen más detalles de algunas directivas de uso común, extraídos de la página web de Apache.

2.2.1.3. Virtual Hosts

Los Virtual Hosts de Apache son una herramienta fundamental del servidor, que le proporcionan una gran flexibilidad para realizar configuraciones diferentes.
Es muy común que se quiera aprovechar un mismo servidor web Apache para servir diferentes contenidos web (diferentes páginas web). Sería poco eficiente utilizar un apache diferente para cada página web que queramos servir. A menudo se utiliza un mismo servidor web Apache para servir múltiples contenidos.

El objetivo es poder aprovechar al máximo el servidor, alojando más de un sitio web de forma simultánea. Los Virtual Hosts permiten dar una configuración diferente a cada uno de esos sitios web. Dicho de otra forma, los Virtual Hosts permiten disponer de varios servidores Apache dentro de uno solo, o particionar Apache web Server en varios servidores web.

Como se observa en la imagen anterior, un servidor Apache puede alojar esos dos sitios web con un solo servicio Apache en funcionamiento. Dentro de ese Apache se configurarán dos Virtual Hosts diferentes, cada uno con la configuración específica para cada sitio web, por ejemplo, dentro de cada Virtual Host, habrá un *DocumentRoot* diferente, direccionando cada petición a una carpeta diferente cada uno.

En la imagen anterior, se observa cómo llegan las peticiones de dos sitios diferentes al mismo servidor web Apache. Cabe destacar que este proceso es totalmente externo al servidor web. Un servidor DNS externo, al cual consultan los clientes, es quien dirige a los clientes hacia la IP del servidor Apache.

Existen dos tipos de Virtual Hosts en Apache:

- Virtual Hosts basados en IP: los diferentes contenidos web se sirven a través de diferentes direcciones IP de las que dispone el servidor. Es decir, el servidor Apache dispone de varias direcciones IP, y a través de cada una de ellas sirve contenidos web diferentes.
- Virtual Hosts basados en nombre: los diferentes contenidos web se sirven a

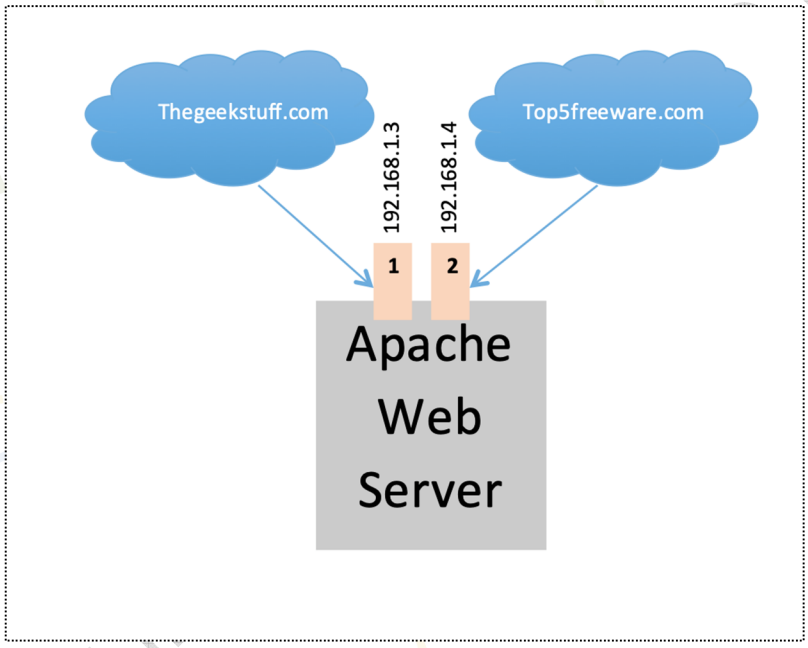
VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP

través de una misma IP del servidor. Para servir diferentes contenidos será necesario que se lea de la petición del cliente qué contenido se requiere, y que además, el servidor sepa identificar cada contenido con el nombre de la petición.

A continuación se profundiza más en estos dos tipos de Virtual Hosts.

Virtual Hosts basados en IP

Si el servidor dispone de varias IPs, se puede configurar Apache para que sirva contenidos web diferentes por cada IP disponible:



En el ejemplo de la imagen anterior, el servidor dispone de dos adaptadores de red y dos IPs. Las peticiones de clientes para el contenido "*thegeekstuff.com*" llegaran a la IP 192.168.1.3 gracias a un DNS externo al que el cliente habrá consultado (sin intervención de Apache). Apache se encargará de servir el contenido correcto cuando la petición llegue a esa IP, y el otro contenido cuando la petición llegue a la otra IP disponible.

En este caso, el trabajo de Apache es fácil, si las peticiones le llegan a una IP, entonces sirve un contenido (un *DocumentRoot*), y si le llega a la otra IP, sirve otro contenido diferente.

A continuación se muestra una configuración de Apache de ejemplo:

```
Listen 80

<VirtualHost 172.20.30.40>
   DocumentRoot /www/example1
   ServerName www.example1.com
</VirtualHost>

<VirtualHost 172.20.30.50>
   DocumentRoot /www/example2
   ServerName www.example2.org
</VirtualHost>
```

En esta configuración se muestran dos virtual hosts basados en IP. Cada virtual host es lo que existe entre las directivas <VirtualHost IP> y </VirtualHost>.

La directiva "Listen" habilita que el servidor escuche peticiones en el puerto especificado, en este caso el 80. La directiva Listen es obligatoria. En el caso de Apache instalado en Ubuntu mediante apt, esta directiva suele estar configurada dentro del archivo de configuración ports.conf.

En la configuración mostrada se puede ver como cada virtual host está asociado a una de las dos IPs del servidor. Así que, si la petición del cliente llega a la IP 172.20.30.40 (recuérdese que en este proceso no interviene Apache, esto lo consigue un servidor DNS externo, que es el que dirige al cliente hacia esa IP), entonces el servidor servirá el contenido presente en el DocumentRoot correspondiente, que para esa IP está en el disco en /www/example1.

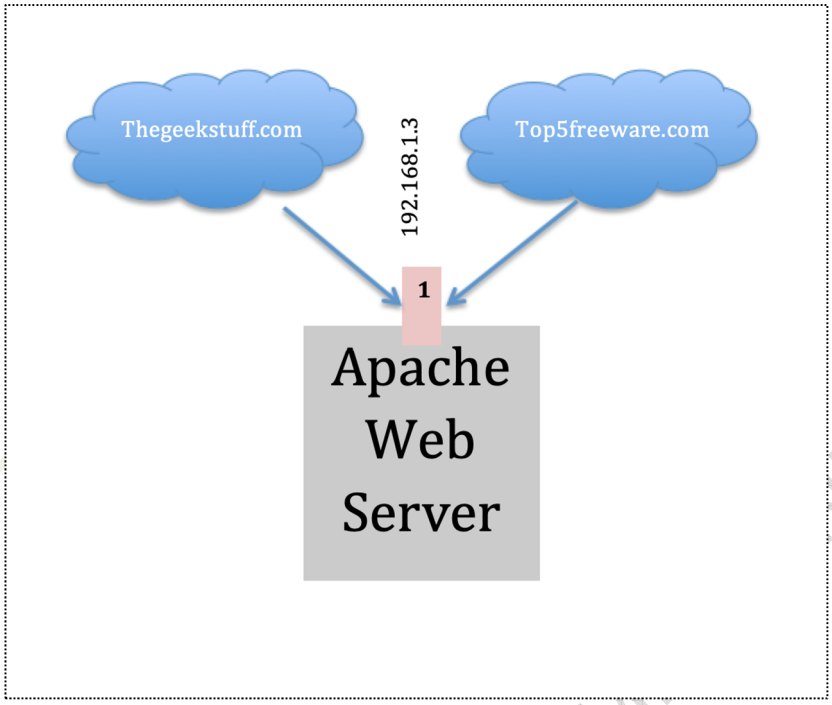
Si la petición del cliente llega a la IP 172.20.30.50 del servidor, entonces se le mostrará el contenido presente en el disco en /www/example2.

En este caso las directivas ServerName presentes no tienen utilidad específica, más allá de identificar las peticiones de los clientes con los Virtual Hosts.
Estas líneas de configuración comentadas deben estar ubicadas en los ficheros de configuración de apache, en este caso, en el de configuración principal de site comentado anteriormente (/etc/apache2/sites-enabled/000-default.conf).

<u>Virtual Hosts basados en nombre</u>

Cuando no se dispone de diferentes IPs en el servidor, aún es posible servir diferentes contenidos web por una misma IP. Para ello se deben configurar los Virtual Hosts basados en nombre:

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP



En este ejemplo el servidor sólo dispone de una IP, y se desean servir dos contenidos diferentes. Las peticiones llegaran todas a la misma IP gracias al servidor DNS externo, y el servidor Apache tendrá que distinguir los dos tipos de peticiones para ofrecerles a cada una el contenido web correspondiente.

A continuación se muestra una nueva configuración con dos Virtual hosts basados en nombre:

```
Listen 80
<VirtualHost 172.20.30.50>
   DocumentRoot /www/example1
   ServerName www.example1.com

   # Other directives here ...

</VirtualHost>

<VirtualHost 172.20.30.50>
   DocumentRoot /www/example2
   ServerName www.example2.org

   # Other directives here ...

</VirtualHost>
```

Se puede observar como la IP del servidor 172.20.30.50 está utilizada en dos virtual hosts diferentes, cada uno con un DocumentRoot diferente (diferente contenido). Éstos son entonces virtual hosts basados en nombre. Es imprescindible que dentro

de cada virtual host se declare el ServerName para diferenciar las peticiones de los clientes y mostrar a cada uno el contenido que ha pedido.

Gracias a que el nombre que ha pedido el cliente también está incluido en la petición HTTP, el servidor puede discernir qué contenido enviar a cada cliente. Si se le pide www.example1.com, se le ofrece el contenido que se halle en /www/example1 en el disco, y si se le pide por www.example2.org, se le ofrece el contenido de /www/example2 en el disco. Todo ello a través de una sola IP, la 172.20.30.50. En este caso la directiva ServerName es necesaria para diferenciar los virtual hosts basados en nombre.

Otras configuraciones con Virtual Hosts

Las configuraciones con virtual hosts pueden ser variadas y complejas. A continuación se comentan otros ejemplos para ofrecer un mejor entendimiento de estas configuraciones.

Ejemplo

```
Listen 80
<VirtualHost *:80>
   DocumentRoot /www/example1
   ServerName www.example1.com

   # Other directives here

</VirtualHost>

<VirtualHost *:80>
   DocumentRoot /www/example2
   ServerName www.example2.org

   # Other directives here

</VirtualHost>
```

La configuración anterior muestra dos Virtual Hosts basados en nombre. En este caso, el carácter "*" indica "todas las IPs del servidor". Así que, por cualquier IP que tenga el servidor web (ya sean una, dos o cinco, o las que sean), por todas ellas se servirán los dos contenidos web, dependiendo de cuál se solicite, se enviará uno o el otro.

En este caso se especifica el puerto 80 además de la IP * (*:80). En caso de que el puerto no se especifique, como sucedía en ejemplos anteriores, el puerto por defecto es siempre el 80, así que en este caso podría omitirse el 80. Se suele hacer por claridad, pero tan solo es necesario hacerlo cuando el puerto no es el 80.

VERSIÓN ALUMNO LINKIAFP

Ejemplo

```
<VirtualHost 192.168.1.1 172.20.30.40>
   DocumentRoot /www/server1
   ServerName server.example.com
   ServerAlias server
</VirtualHost>
```

En esta configuración se ofrece el mismo contenido por dos IPs diferentes

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP

Ejemplo

```
Listen 172.20.30.40:80
Listen 172.20.30.40:8080
Listen 172.20.30.50:80
Listen 172.20.30.50:8080

<VirtualHost 172.20.30.40:80>
   DocumentRoot /www/example1-80
   ServerName www.example1.com
</VirtualHost>

<VirtualHost 172.20.30.40:8080>
   DocumentRoot /www/example1-8080
   ServerName www.example1.com
</VirtualHost>

<VirtualHost 172.20.30.50:80>
   DocumentRoot /www/example2-80
   ServerName www.example1.org
</VirtualHost>

<VirtualHost 172.20.30.50:8080>
   DocumentRoot /www/example2-8080
   ServerName www.example2.org
</VirtualHost>
```

En esta configuración se ofrecen cuatro contenidos diferentes. Se dispone de dos IPs en el servidor, y de cada IP se utilizan dos puertos, así que disponemos de cuatro Virtual Hosts con cuatro contenidos diferentes.

Para utilizar IPs y puertos concretos tienen que estar habilitados mediante la directiva Listen.

VERSIÓN IMPRIMIBLE
LINKIAFP

Ejemplo

```
Listen 80

<VirtualHost 172.20.30.40>
   DocumentRoot /www/example1
   ServerName www.example1.com
</VirtualHost>

<VirtualHost 172.20.30.40>
   DocumentRoot /www/example2
   ServerName www.example2.org
</VirtualHost>

<VirtualHost 172.20.30.40>
   DocumentRoot /www/example3
   ServerName www.example3.net
</VirtualHost>

# IP-based
<VirtualHost 172.20.30.50>
   DocumentRoot /www/example4
   ServerName www.example4.edu
</VirtualHost>

<VirtualHost 172.20.30.60>
   DocumentRoot /www/example5
   ServerName www.example5.gov
</VirtualHost>
```

En esta configuración se mezclan virtual hosts basados en nombre y basados en IPs. Por la IP 172.20.30.40 se sirven tres virtual hosts diferentes (basados en nombre), y por la IP 172.20.30.50 un contenido diferente (basado en IP) y por la IP 172.20.30.60 un contenido más (basado en IP).

VERSIÓN IMPRIMIBLE PERTENECIENTE AL CURSO ONLINE DE CONTENIDOS ACREDITADOS PARA EL CICLO FORMATIVO DE GRADO SUPERIOR DE DESARROLLO DE APLICACIONES WEB DE LA FAMILIA PROFESIONAL DE INFORMÁTICA Y COMUNICACIONES, OFERTADO EN EL CENTRO DE FORMACIÓN PROFESIONAL ONLINE LINKIAFP

Ejemplo

```
Listen 80
Listen 8080

<VirtualHost 172.20.30.40:80>
   ServerName www.example1.com
   DocumentRoot /www/domain-80
</VirtualHost>

<VirtualHost 172.20.30.40:8080>
   ServerName www.example1.com
   DocumentRoot /www/domain-8080
</VirtualHost>

<VirtualHost 172.20.30.40:80>
   ServerName www.example2.org
   DocumentRoot /www/otherdomain-80
</VirtualHost>

<VirtualHost 172.20.30.40:8080>
   ServerName www.example2.org
   DocumentRoot /www/otherdomain-8080
</VirtualHost>
```

En esta configuración hay dos virtual hosts basados en nombre en la IP 172.20.30.40 puerto 80, y otros dos virtual hosts basados en nombre en la misma IP puerto 8080.

Ejemplo

```
Listen 80

# This is the "main" server running on 172.20.30.40
ServerName server.domain.com
DocumentRoot /www/mainserver

# This is the other address


<VirtualHost 172.20.30.50>
   DocumentRoot /www/example1
   ServerName www.example1.com

   # Other directives here ...

</VirtualHost>

<VirtualHost 172.20.30.50>
   DocumentRoot /www/example2
   ServerName www.example2.org

   # Other directives here ...

</VirtualHost>
```

En esta configuración se presenta el concepto del "main server" o servidor principal. El main server es la parte del fichero de configuración que no está dentro de ninguna etiqueta de virtual host.

En este ejemplo se encuentra en el main server un DocumentRoot con un contenido web concreto. Las peticiones que no vayan dirigidas a ningún virtual host serán recogidas por la configuración del main server.

Es este caso, la IP 172.20.30.50 tiene dos virtual hosts basados en nombre asociados. Si el servidor dispone de otras IPs (por ej la 172.20.30.40) las peticiones que vayan a esas otras IPs, como no tienen ningún virtual host asociado, las recogerá la configuración del main server.

También es importante destacar que, en caso de que llegue una petición a la IP 172.20.30.50, pero que no sea para ninguno de los nombres declarados en los ServerName correspondientes (esto puede provocarlo un DNS externo), la petición será recogida por el virtual host de esa IP que más arriba se encuentre en el fichero de configuración.

2.2.1.4. Autenticación

La autenticación para restringir el acceso a un sitio web es una funcionalidad muy útil en diferentes situaciones. Muchos sitios web requieren filtrar el acceso de los usuarios a cierto contenido.

Dicha autenticación se puede incrustar y programar en el contenido web de la página, pero también la puede ejercer el servidor web Apache. En estos apuntes se verá un ejemplo básico de configuración de autenticación por parte del servidor.

Existen diversos tipos de autenticación, unos más seguros y eficientes que otros. En estos apuntes se mostrará los dos modos de configurar autenticación de servidor. Se trata de la denominada autenticación basic y la digest.
La diferencia entre ambos es que en el modo Basic, las contraseñas se envían sin cifrar, mientras que en Digest se envían cifradas.

El módulo de Apache que controla la autenticación Basic es auth_basic, y está habilitado por defecto. En cambio, el módulo auth_digest está deshabilitado.

Se debe tener en cuenta que mediante la autenticación básica no se ofrece una solución completa de seguridad, ya que los nombres de usuario y contraseña que escribe el cliente desde su navegador viajan de forma visible hacia el servidor, y cualquier observador desde un punto intermedio de la red podría interceptarlos fácilmente.

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP

Creación de usuarios

En primer lugar se debe instalar el paquete *apache2-utils (apt install apache2-utils*). Para crear el fichero de usuarios utilizamos el comando htpasswd (si es autenticación Basic) ó htdigest (si es Digest).

La primera vez que utilicemos el comando, tanto *htpasswd* como *htdigest*, debemos ponerle la opción -c para que cree el archivo. Si creamos más usuarios debemos omitir ésta opción para no sobrreescribir el archivo de usuarios.

Para utilizar la autenticación de usuarios debe estar instalado el módulo authz_user, que por defecto lo está.

- Con autenticación Basic, haremos lo siguiente para la creación de los usuarios.

```
root@linkia:/home/linkia# htpasswd -c /etc/apache2/contr_basic alumno1
New password:
Re-type new password:
Adding password for user alumno1
root@linkia:/home/linkia# htpasswd /etc/apache2/contr_basic alumno2
New password:
Re-type new password:
Adding password for user alumno2
root@linkia:/home/linkia#
```

- Con autenticación Digest hay que indicar además el dominio (grupo) al que pertenece el usuario.

```
root@linkia:/etc/apache2# htdigest -c /etc/apache2/contr_digest linkia profe1
Adding password for profe1 in realm linkia.
New password:
Re-type new password:
root@linkia:/etc/apache2# htdigest /etc/apache2/contr_digest linkia profe2
Adding user profe2 in realm linkia
New password:
Re-type new password:
root@linkia:/etc/apache2#
```

En el ejemplo anterior se muestra el uso de la aplicación htpasswd como se ha explicado:

- La opción "-c" debe indicarse tan solo la primera vez para que se cree el fichero.
- A continuación se debe especificar el nombre del fichero que se desea generar. Aunque el nombre y la ubicación del fichero pueden ser cualesquiera, no se recomienda que esté dentro de los directorios disponibles vía web.

- Ene l caso de la autenticación Digest después de especificar el nombre del fichero es necesario especificar el dominio.clea
- A continuación el nombre de usuario que se desea conste en el archivo (en el ejemplo "alumno1" o “profe1”)

La aplicación pedirá que se introduzca el password deseado para cada uno de los usuarios.

Después de introducir el password dos veces, se genera el archivo con dos usuarios incluidos (alumno1 y alumno2 / profe1 y profe2).

Una vez creado el fichero de contraseñas y rellenado con todos los usuarios a los que se desea permitir el acceso, se debe configurar Apache con las siguientes directivas en su fichero de configuración:

En el caso de autenticación básica el fichero debería ser parecido al siguiente:

```
  GNU nano 2.9.3                    000-default.conf                    Modified

<VirtualHost *:80>
<Directory "/var/www/html/alumnos">
        AuthType basic
        AuthName "alumnos"
        AuthBasicProvider file
        AuthUserFile "/etc/apache2/contr_basic"
        Require user alumno1
</Directory>
```

En el caso de autenticación digest el fichero debería ser parecido al siguiente:

```
  GNU nano 2.9.3                    000-default.conf

<VirtualHost *:80>
<Directory "/var/www/html/profesores">
        AuthType digest
        AuthName "linkia"
        AuthBasicProvider file
        AuthUserFile "/etc/apache2/contr_digest"
        Require user profe1
</Directory>
```

Todas estas directivas habilitan la autenticación básica en el servidor.
A continuación una explicación breve de lo que configura cada directiva:

- AuthType: tipo de autenticación Basic (Digest un método más seguro).
- AuthName: Realm aparece este texto en la ventana de autenticación. Si en el mismo servidor se vuelve a configurar el mismo Realm y el usuario ya está validado previamente, no volverá a preguntar.
- AuthBasicProvider: por defecto file. Donde va a buscar los usuarios, a un archivo, a una base de datos (requiere módulos adicionales

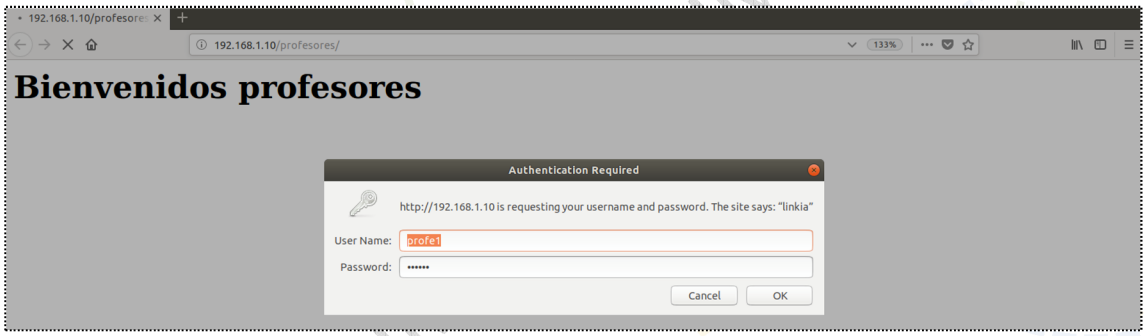
mod_authn_dbm).

- AuthUserFile: ruta del archivo de contraseñas.
- Require: permitir sólo el acceso al usuario alumno1 o profe1

En el ejemplo anterior, a pesar de que en el fichero de contraseñas hubiera muchos usuarios creados, tan solo se le permite el acceso al usuario "alumno1" en autenticación básico y “profe1” en autenticación avanzada.

Deberemos crear las rutas /var/www/html/alumno y /var/www/html/profesores y se deberá crear en su interior un fichero index.html que contenga el contenido deseado.

Cuando el servicio haya sido reiniciado, se podrá comprobar la funcionalidad desde un cliente y accediendo, en este caso, a http://192.168.1.10/alumnos o http://192.168.1.10/profes desde donde nos pedirá el usuario y el password del usuario desde el que queremos entrar.



Más opciones:

Si se desea dar acceso a todos los usuarios del fichero de contraseñas, se puede configurar la opción:

```
Require valid-user
```

También se puede controlar el acceso mediante el uso de grupos. Para ello se debe crear un fichero adicional donde consten los nombres de los grupos y de sus usuarios miembros (que deben constar en el archivo de contraseñas).
Por ejemplo, se podría crear un archivo con el siguiente contenido:

```
GroupName: rbowen dpitts sungo rshersey
```

Dicho archivo se debe guardar con el nombre que se desee y en una ubicación que no esté disponible vía web (fuera del DocumentRoot).

VERSIÓN IMPRIMIBLE ALUMNO LINKIA FP

En el ejemplo se puede ver un grupo de nombre "GroupName" con cuatro usuarios miembros.

Después se debería configurar Apache con las siguientes directivas:

```
AuthType Basic
AuthName "By Invitation Only"
# Optional line:
AuthBasicProvider file
AuthUserFile /usr/local/apache/passwd/passwords
AuthGroupFile /usr/local/apache/passwd/groups
Require group GroupName
```

La diferencia respecto al caso anterior es que se debe especificar también el fichero de grupos, y que la directiva "Require" hace referencia ahora al nombre del grupo creado, y da acceso a todos los usuarios miembros del grupo.

Para que estas directivas de autenticación en general y de grupos (AuthGroupFile) funcionen, deben estar cargados en Apache los módulos que vienen incluido en el directorio mods-available. Para cargarlos se debe ejecutar el script a2enmod:

a2enmod authz_groupfile
a2enmod authz_digest

Este script pone el módulo en el directorio mods-enabled, y a partir de entonces ya se pueden utilizar las directivas que incluye.

Todas las directivas de autenticación deben ir siempre agrupadas dentro de directivas "Directory". Las directivas directory permiten agrupar configuraciones y aplicarlas tan solo a un directorio del disco, es decir, a una parte del contenido web, no a toda la página web.

Por ejemplo, es común el caso en que el acceso a la página web principal es abierto, y en cuanto se hace click en una sección se desea filtrar el acceso con autenticación. En ese caso se deberían ubicar las directivas de autenticación dentro de la directiva Directory que identifique al directorio donde está el contenido web de la sección a la que se desea acceder.

El uso de la directiva Directory permite establecer diferentes tipos de autenticación para diferentes partes del contenido web:

VERSIÓN IMPRIMIBLE ALUMNO LI

```
<Directory "/www/docs">
    AuthType Basic
    AuthName Documents
    AuthBasicProvider file
    AuthUserFile "/usr/local/apache/passwd/passwords"
    Require valid-user
</Directory>

<Directory "/www/docs/public">
    AuthType None
    Require all granted
</Directory>
```

En el ejemplo mostrado anteriormente se observa como para la sección "/www/docs" se requiere uno de los usuarios presentes en el fichero de contraseñas indicado, mientras que para la sección "/www/docs/public" el acceso es libre, sin autenticación.

El sistema autenticación básica tiene muchas limitaciones respecto a la seguridad y el rendimiento que puede ofrecer:

- La seguridad es insuficiente ya que los nombres de usuario y contraseña viajan por la red sin encriptar. Se recomienda utilizar digest.
- El rendimiento de esta autenticación es malo ya que el usuario y password se deben evaluar cada vez que se pide un contenido al servidor. Eso supone acceder al archivo cada vez por parte del servidor, lo que puede ralentizar los accesos con cargas elevadas. Para superar esta limitación se pueden ubicar los usuarios en bases de datos organizadas, tipo SQL o LDAP, por ejemplo.

2.2.1.5. Control de acceso

Para filtrar quién puede acceder a un contenido, Apache dispone también de otros mecanismos además de la autenticación de usuarios. También es posible filtrar quien puede acceder basándose en la IP o nombre de origen de la máquina cliente que quiere acceder.

Para ello apache dispone de las directivas *Order, Allow y Deny*, las cuales permiten filtrar qué clientes pueden acceder y cuáles no. Aunque se puede hacer tanto por nombre del cliente como por IP del cliente, es más eficiente, si se puede, hacerlo por IP. Si se hace filtrado por nombre del cliente, el servidor se debe encargar también de resolver el nombre DNS del cliente, lo cual puede ser una tarea muy pesada para un servidor web.

VERSIÓN IMPRIMIBLE ALUMNO LIKIAFP

Se evaluarán todas las directivas Allow y Deny independientemente del orden en que estén escritas. La directiva Order es la que fijará el modo de funcionamiento de las otras dos, existen dos posibilidades:

- Order Allow, Deny
  - Se evalúan primero los Allow.
  - Si hay "match" se permite, excepto cuando haya "match" también después en los Deny.
  - Se evalúan los Deny.
  - Si hay "match" se deniega. Si no hay "match" se deniega.
- Order Deny, Allow
  - Se evalúan primero los Deny.
  - Si hay match, se deniega, excepto cuando haya match también después en los Allow.
  - Se evalúan los Allow.
  - Si hay match se permite. Si no hay match, se permite

La expresión "hacer match" se refiere al hecho que coincida la IP o nombre del cliente con alguna de las expresiones de las directivas Allow o Deny.

Las directivas Allow y Deny funcionan de forma idéntica entre sí. Establecen los patrones por los que se permitirá (Allow) o denegará (Deny) el acceso del cliente.
Por ejemplo, si se desea permitir el acceso de tres IPs de cliente en concreto se pueden identificar así:

```
Allow from 10.1.2.3
Allow from 192.168.1.104 192.168.1.205
```

Aunque está escrito en dos líneas se podría haber escrito también en una sola.

Si se desean identificar subredes completas también se puede escribir:

```
Allow from 10 172.20 192.168.2
```

Esta sintaxis permitiría todas las IPs del tipo 10.X.X.X , 172.20.X.X y 192.168.2.X.

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP

La combinación de las directivas Allow, Deny y order configura un filtro de acceso para los clientes. Un ejemplo de configuración completa sería la siguiente:

Order Deny,Allow
Deny from All
Allow from 81.97.62.218

En este caso, se evalúa primero los Deny, es decir, se deniega el acceso a todo el mundo (Deny from All). Pero también hay un Allow de una IP en concreto, y como los Allow se evalúan después, éstos tienen preferencia, así que esta IP en concreto sí que tendrá acceso, de hecho, será la única que lo tenga. Sólo un cliente con la IP 81.97.62.218 podrá acceder a la sección correspondiente de la configuración.
Este bloque de configuración se puede ubicar en diferentes sitios en una configuración de Apache, típicamente por ejemplo, dentro de una directiva <Directory> para restringir el acceso a una zona determinada de la web.
El orden en que se escriban las líneas es irrelevante, aunque por claridad es útil mantener un orden coherente, con la directiva Order en primer lugar, y después las Allow y Deny.

A modo de resumen, se presenta la siguiente tabla que explica qué sucede en cada posible caso de configuración de estas directivas:

Match	Allow,Deny result	Deny,Allow result
Match Allow only	Request allowed	Request allowed
Match Deny only	Request denied	Request denied
No match	Default to second directive: Denied	Default to second directive: Allowed
Match both Allow & Deny	Final match controls: Denied	Final match controls: Allowed

Por último, un ejemplo más de configuración completa:

```
<Directory /var/www/sub/payroll/>
Order allow,deny
Allow from 192.168.1.0/24
Allow from 127
</Directory>
```

¿Qué consigue esta configuración?

En primer lugar, se observa como el filtrado sólo aplica a un directorio en concreto (/var/www/sub/payroll/). Se puede observar que no existen directivas Deny, pero ya que el Order es "allow, deny", a cualquier cliente que no esté especificado en el allow, se le aplicará un deny.

En definitiva, la configuración mostrada, permite el acceso exclusivo de clientes de una subred completa (192.168.1.X) y de las IPs del tipo 127.X.X.X, que suelen ser las asociadas al propio servidor (localhost), los demás clientes no tendrán acceso a este directorio de la web.

2.2.1.6. Ficheros .htaccess

Apache dispone de los ficheros .htaccess para flexibilizar aún más su configuración. Estos ficheros pueden ser creados en el directorio que se desee para realizar una configuración adicional del servidor que aplique tan solo a ese directorio.
Existen entornos donde un solo servidor Apache puede querer ser administrado por varios administradores. Por ejemplo, en un hosting compartido, donde un Apache es compartido entre varios clientes, donde cada uno quiere configurar el servidor de forma diferente. Gracias a los ficheros .htaccess, cada uno puede crear su fichero .htaccess en su directorio asignado, y dentro del fichero, escribir directivas que cambiarán la configuración de Apache en ese directorio en concreto.

El administrador general del servidor tiene los ficheros generales de configuración del servidor, a los que no accederían los clientes. A cada cliente solo se le asigna un espacio de disco para subir su contenido web, pero además, si se quiere, se le puede dar permiso para que él mismo se cree los ficheros .htaccess y dentro de ellos haga la configuración extra que necesite, como si fueran una extensión de los ficheros de configuración general de Apache.

El administrador general puede controlar si da permiso para que los ficheros .htaccess sean efectivos o no. Los ficheros .htaccess son una forma de distribuir la configuración en diferentes ficheros. El fichero .htaccess se coloca en un directorio particular, y las directivas que contenga, aplicarán a ese directorio y todos sus subdirectorios. La sintaxis empleada en los ficheros .htaccess es idéntica a la de los archivos de configuración generales.
Se deberá consultar para cada directiva, si se puede utilizar dentro de un archivo .htaccess o no. En el apartado Context de la documentación de una directiva se indica dónde se puede escribir una directiva:

Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo

En el ejemplo mostrado, la directiva a la que pertenece esta documentación se podría utilizar tanto en el "main server", en un virtual host, en un directory o en un archivo .htaccess.

VERSIÓN IMPRIMIBLE AKIMOINKAFP

El administrador general habilita o no el uso de directivas en los ficheros .htaccess mediante la directiva *AllowOverride*. Con esta directiva, se permite el uso de directivas, agrupadas en grupos temáticos, en archivos .htaccess.

En la documentación de cada directiva, como se ve en la imagen anterior, además del Context, también se puede observar el campo "Override", que en el ejemplo tiene el valor de FileInfo. Eso significa que esa directiva pertenece al grupo Fileinfo.

Los usuarios, si se les ha permitido, tan solo tendrán que crear en el directorio que lo necesiten un fichero con el nombre .htaccess (fichero oculto al empezar por "."), y dentro de él escribir las directivas con la misma sintaxis que se haría en los ficheros de configuración general.

De la documentación de Apache, se presentan los grupos de directivas mencionados:

AuthConfig
Allow use of the authorization directives (`AuthDBMGroupFile`, `AuthDBMUserFile`, `AuthGroupFile`, `AuthName`, `AuthType`, `AuthUserFile`, `Require`, *etc.*).

FileInfo
Allow use of the directives controlling document types (`ErrorDocument`, `ForceType`, `LanguagePriority`, `SetHandler`, `SetInputFilter`, `SetOutputFilter`, and `mod_mime` Add* and Remove* directives), document meta data (`Header`, `RequestHeader`, `SetEnvIf`, `SetEnvIfNoCase`, `BrowserMatch`, `CookieExpires`, `CookieDomain`, `CookieStyle`, `CookieTracking`, `CookieName`), `mod_rewrite` directives `RewriteEngine`, `RewriteOptions`, `RewriteBase`, `RewriteCond`, `RewriteRule`) and `Action` from `mod_actions`.

Indexes
Allow use of the directives controlling directory indexing (`AddDescription`, `AddIcon`, `AddIconByEncoding`, `AddIconByType`, `DefaultIcon`, `DirectoryIndex`, `FancyIndexing`, `HeaderName`, `IndexIgnore`, `IndexOptions`, `ReadmeName`, *etc.*).

Limit
Allow use of the directives controlling host access (`Allow`, `Deny` and `Order`).

Options[=*Option*,...]
Allow use of the directives controlling specific directory features (`Options` and `XBitHack`). An equal sign may be given followed by a comma (but no spaces) separated lists of options that may be set using the `Options` command.

Esto significa que si el administrador general configura en los ficheros de configuración general lo siguiente:

AllowOverride AuthConfig

Entonces está habilitando que los subadministradores incluyan ficheros .htaccess en sus directorios y allí puedan configurar directivas como AuthType, AuthName, etc. pero no las de otro grupo.

Si el administrador configura:
AllowOverride All
Entonces está permitiendo el uso de todas las directivas que se puedan usar en ficheros .htaccess

Si el administrador no quiere que nadie utilice los ficheros .htaccess, configurará:
AllowOverride None

El uso de ficheros .htaccess de forma general no está recomendado por Apache, ya que puede afectar al rendimiento. Tan solo se impone su utilización cuando es necesario delegar a otros la configuración de ciertos directorios y permitirles que puedan añadir sus propias directivas, sin que tengan acceso a los ficheros de configuración general.

En cualquier otra situación, donde se quiera aplicar una configuración determinada a un directorio determinado, es preferible el uso de la directiva Directory, que tendrá el mismo efecto.
De la documentación oficial de Apache se extrae el siguiente aviso donde no se recomienda el uso de estos archivos:

> You should avoid using `.htaccess` files completely if you have access to httpd main server config file. Using `.htaccess` files slows down your Apache server. Any directive that you can include in a `.htaccess` file is better set in a `Directory` block, as it will have the same effect with better performance.

2.2.1.7. Alias, redireccionamiento y reescritura

Cuando una petición llega al servidor web, a veces es necesario tratarla para encontrar en otro sitio los contenidos solicitados, ya sea porque se encuentran en otra parte del disco del servidor, o incluso en otro servidor. Se dispone principalmente de tres directivas para efectuar estas operaciones:

- *Alias*: conectar una URL con un directorio local concreto
- *Redirect*: conectar una URL con otra
- Rewrite: alterar la URL

Alias y Redirect son directivas del módulo *mod-alias*, que por defecto se encuentra cargado en la instalación básica de Apache. Para comprobar que ese módulo está cargado se puede comprobar el directorio /etc/apache2/mods-enabled:

VERSIÓN IMPRIMIBLE UMNO LINKAPP

```
root@linkia:/etc/apache2/mods-enabled# ls
access_compat.load  authz_core.load       deflate.load  mpm_event.conf    setenvif.load
alias.conf          authz_groupfile.load  dir.conf      mpm_event.load    status.conf
alias.load          authz_host.load       dir.load      negotiation.conf  status.load
auth_basic.load     authz_user.load       env.load      negotiation.load
auth_digest.load    autoindex.conf        filter.load   reqtimeout.conf
authn_core.load     autoindex.load        mime.conf     reqtimeout.load
authn_file.load     deflate.conf          mime.load     setenvif.conf
root@linkia:/etc/apache2/mods-enabled# _
```

Como se observa en la imagen, los ficheros correspondientes al módulo alias están presentes en el directorio (alias.conf y alias.load), y por lo tanto están activados.

La directiva Rewrite forma parte del módulo mod-rewrite, que puede no estar activo. Si se desea activar el módulo se debe ejecutar el comando *a2enmod rewrite*:

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

serv
Se puede observar como después de activar el módulo el sistema obliga a reiniciar el servicio de apache.
Se puede comprobar que el módulo ha sido habilitado ya que estarán los archivos correspondientes en el directorio mods-enabled:

```
root@linkia:/etc/apache2/mods-enabled# ls
access_compat.load  authz_core.load       deflate.load  mpm_event.conf    setenvif.conf
alias.conf          authz_groupfile.load  dir.conf      mpm_event.load    setenvif.load
alias.load          authz_host.load       dir.load      negotiation.conf  status.conf
auth_basic.load     authz_user.load       env.load      negotiation.load  status.load
auth_digest.load    autoindex.conf        filter.load   reqtimeout.conf
authn_core.load     autoindex.load        mime.conf     reqtimeout.load
authn_file.load     deflate.conf          mime.load     rewrite.load
root@linkia:/etc/apache2/mods-enabled# _
```

Se observa cómo ha aparecido el fichero *rewrite.load*. A partir de este momento también se puede hacer uso de la directica *rewrite* incluida en el módulo.

<u>Alias</u>

La directiva Alias sirve para que el servidor pueda tener almacenados los contenidos fuera del directorio especificado en el DocumentRoot. Habitualmente se podrá tener todo el contenido web dentro de un solo directorio (y sus subdirectorios), ese directorio será configurado como Document Root, pero a veces es necesario que el servidor acuda a directorios fuera del DocumentRoot para buscar contenido web y enviarlo a los clientes. Cuando el contenido web puede estar almacenado fuera del DocumentRoot se puede hacer uso de la directiva Alias.

En una situación normal, cuando un cliente web solicita el siguiente contenido:
http://www.ejemplo.com/imagen1.jpg
El servidor irá a buscar el fichero "*imagen1.jpg*" en el directorio que tenga configurado como DocumentRoot.

Si la petición web es de la siguiente forma:
http://www.ejemplo.com/imagenes/imagen1.jpg
El servidor irá a buscar el fichero "*imagen1.jpg*" a un directorio llamado "*imagenes*" dentro del directorio que tenga configurado como DocumentRoot.
Hasta aquí el funcionamiento normal del sistema.

Si se desea almacenar contenido fuera del DocumentRoot, o con más flexibilidad, se puede configurar la directiva Alias.

Por ejemplo, si se configura la siguiente directiva en Apache:

```
Alias /image /ftp/pub/image
```

Se está indicando al servidor que cuando la petición web vaya al subdirectorio "/image" el contenido se debe ir a buscar al directorio del disco "/ftp/pub/image"

Es decir, si el cliente web solicita:

```
http://example.com/image/foo.gif
```

El servidor lo irá a buscar en el disco en la ubicación:

```
/ftp/pub/image/foo.gif
```

<u>Redirect</u>

Si lo que se desea es que cuando el servidor reciba una petición web se reenvíe al cliente a otra dirección URL, entonces se debe utilizar la directiva Redirect.

Por ejemplo, si el servidor tiene configurada la siguiente directiva:

```
Redirect /service http://foo2.example.com/service
```

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP

Se está indicando que cuando el cliente web solicite:

```
http://example.com/service/foo.txt,
```

La petición se redirigirá a:

```
http://foo2.example.com/service/foo.txt
```

Nótese que en este caso se ha modificado también la parte del *hostname* de la URL (se ha cambiado "example.com" por " foo2.example.com"), cosa que no es posible con la directiva Alias.

Rewrite y otras configuraciones

Las posibilidades de tratamiento de las URL que llegan al servidor como peticiones web se amplían con otras directivas del módulo mod-rewrite.

Sin embargo estas posibilidades quedan fuera del alcance de estos apuntes. Para dominar las técnicas de rewrite es necesario tener conocimiento previo de las llamadas "expresiones Regulares" para el establecimiento de patrones de texto.

Apache utiliza el vocabulario Perl Compatible Regular Expression (PCRE) para especificar patrones en diferentes directivas, entre ellas las relacionadas con las operaciones de Rewrite.

Aunque las expresiones regulares quedan también fuera del alcance de estos apuntes, se presenta a continuación una tabla de operadores básicos.

Character	Meaning	Example
.	Matches any single character	`c.t` will match `cat`, `cot`, `cut`, etc.
+	Repeats the previous match one or more times	`a+` matches `a`, `aa`, `aaa`, etc
*	Repeats the previous match zero or more times.	`a*` matches all the same things `a+` matches, but will also match an empty string.
?	Makes the match optional.	`colou?r` will match `color` and `colour`.
^	Called an anchor, matches the beginning of the string	`^a` matches a string that begins with `a`
$	The other anchor, this matches the end of the string.	`a$` matches a string that ends with `a`.
( )	Groups several characters into a single unit, and captures a match for use in a backreference.	`(ab)+` matches `ababab` - that is, the + applies to the group. For more on backreferences see below.
[ ]	A character class - matches one of the characters	`c[uoa]t` matches `cut`, `cot` or `cat`.
[^]	Negative character class - matches any character not specified	`c[^/]t` matches `cat` or `c=t` but not `c/t`

VERSIÓN ALUMNO LINKIAFP

Con estos operadores se pueden describir diferentes patrones que pueden cumplir las peticiones web, y que según las condiciones impuestas por las directivas, se puedan modificar las URLs especificadas por el cliente.

También las expresiones regulares se pueden utilizar en directivas tipo Alas y Redirect pero con identificación de patrones, como son las directivas AliasMatch, y RedirectMatch.

Por ejemplo, la siguiente directiva

```
RedirectMatch (.*)\.gif$ http://www.anotherserver.com$1.jpg
```

Redirige accesos a archivos .gif hacia un nuevo servidor con ficheros .jpg con los mismos nombres que los gifs originales.

Otro ejemplo, si el servidor tiene la siguiente directiva configurada:

```
RedirectMatch 301 /handler.php?cat=(.*)&article=(.*) http://www.devdaily.com/$1/$2
```

Cuando llega al servidor una petición del tipo:

```
http://www.devdaily.com/handler.php?cat=java&article=1001
```

Se le reenvía a la siguiente URL:

```
http://www.devdaily.com/java/1001
```

2.2.1.8. Registro de eventos y errores

Para administrar eficientemente un servidor web es necesario tener información de la actividad detallada del servidor, incluyendo los errores que puedan suceder.

Hay que diferenciar ente registrar los eventos calificados como errores (que un módulo no se cargue correctamente, por ejemplo) y registrar la actividad completa del servidor, donde se registra cualquier acceso al que dé servicio el servidor.

Para ello Apache ofrece diferentes directivas:

- ErrorLog: establece el fichero donde se registran los errores
- LogLevel: establece la severidad de los errores a registrar
- CustomLog: establece el fichero donde registrar los accesos al servidor
- LogFormat: establece el formato del registro de accesos al servidor

VERSIÓN IMPRIMIBLE ACIM - ALINEP

Mediante ErrorLog se establece el fichero para el registro de errores, por ejemplo, si se configura:

```
ErrorLog /var/log/httpd/error_log
```

dentro del fichero indicado, se podrían generar errores del tipo:

```
[Wed Oct 11 14:32:52 2000] [error] [client 127.0.0.1] client denied by server configuration: /export/home/live/ap/htdocs/test
```

Cada error se escribirá en una línea del fichero, con datos sobre el mismo, en este caso Fecha y hora, severidad del error, IP del cliente, descripción, ruta del documento solicitado.

Otro ejemplo de error generado:

```
[Fri Sep 09 10:42:29.902022 2011] [core:error] [pid 35708:tid 4328636416] [client 72.15.99.187] File
does not exist: /usr/local/apache2/htdocs/favicon.ico
```

Mediante la directiva LogLevel se establece qué nivel de severidad de errores quedarán registrados. A veces es necesario activar un nivel de severidad baja para que se registren todos los errores posibles y se pueda detectar un posible mal funcionamiento. Una vez detectado el fallo, es recomendable volver a un nivel de severidad más alto para que la tarea de registrar errores no suponga una carga para el servidor.

Los diferentes niveles de severidad son los siguientes:

Level	Description	Example
emerg	Emergencies - system is unusable.	"Child cannot open lock file. Exiting"
alert	Action must be taken immediately.	"getpwuid: couldn't determine user name from uid"
crit	Critical Conditions.	"socket: Failed to get a socket, exiting child"
error	Error conditions.	"Premature end of script headers"
warn	Warning conditions.	"child process 1234 did not exit, sending another SIGHUP"
notice	Normal but significant condition.	"httpd: caught SIGBUS, attempting to dump core in ..."
info	Informational.	"Server seems busy, (you may need to increase StartServers, or Min/MaxSpareServers)..."
debug	Debug-level messages	"Opening config file ..."

Un ejemplo de configuración de esta directiva sería:

LogLevel debug

Con esta configuración se registrarán muchos errores, por poco relevantes que sean. Se trata del modo "debug", que suele tener utilidad de detección de fallos, y no es válido para un servidor en producción.

Más allá de registrar errores, se pueden registrar todos las peticiones procesadas por un servidor apache. La ubicación y contenido del log de acceso los controla la directiva CustomLog. La directiva LogFormat establece el formato y contenido de los logs

Como es habitual, el trabajo de análisis de los logs generados puede ser una tarea muy tediosa y complicada. Para optimizarla existe software de terceros capaz de analizar y mostrar estadísticas de todos los logs generados por Apache.

AWStats es un ejemplo de software de terceros que se dedica a recoger el contenido de los ficheros de log de Apache y generar resúmenes visuales con gráficos incluidos.

Consulte https://help.ubuntu.com/community/AWStats para su instalación en sistemas Ubuntu.

A continuación se presenta un ejemplo típico de configuración con estas dos directivas:

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log common
```

Mediante Logformat se han definido los campos que formarán parte de cada registro de acceso, y a esa configuración se le ha atribuido el nombre "common" (también conocido como CLF, common log format)

A continuación, mediante CustomLog se establece la ubicación del fichero donde se almacenarán los registros de acceso y que se hará con el formato definido como "common".

El resultado de la configuración podría ser una línea como la siguiente incluida en el fichero indicado:

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326
```

Se observa que se registran los campos especificados en el LogFormat. En concreto, los campos disponibles son:

- %h -> IP del cliente
- %l -> remote logname (en el ejemplo no disponible)
- %u -> userid del usuario que hace la petición
- %t -> time and date
- \"%r\" -> la línea de request entre comillas
- %>s -> El código de estado que el servidor devuelve al cliente
- %b -> tamaño del contenido devuelto

Otro ejemplo de formato es el conocido como combined log format.

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\"" combined
CustomLog log/access_log combined
```

En este caso, el resultado de un acceso sería:

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326
"http://www.example.com/start.html" "Mozilla/4.08 [en] (Win98; I ;Nav)"
```

Como se puede observar, contiene más detalles sobre el acceso, entre ellos, el navegador utilizado por el cliente.

Si se desea se pueden generar tantos ficheros de registro cómo se desee. Por ejemplo quizás sea necesario generar un fichero de registro por cada virtual host de un servidor. En ese caso se puede configurar la directiva CustomLog para cada virtual host, apuntando cada una a un fichero diferente.

Además, también se puede hacer constar el nombre del virtual host correspondiente en la línea de registro. Para ello se debe incluir el parámetro %v en la directiva LogFormat:

```
LogFormat "%v %l %u %t \"%r\" %>s %b" comonvhost
CustomLog logs/access_log comonvhost
```

2.2.2. Internet Information Services (IIS)

IIS es un servidor web que está integrado en los sistemas operativos de Microsoft Windows. Aunque es actualizable por separado, por defecto viene una versión integrada en cada versión de Windows Server. Las versiones incluidas en las últimas versiones de sistema operativo Windows:

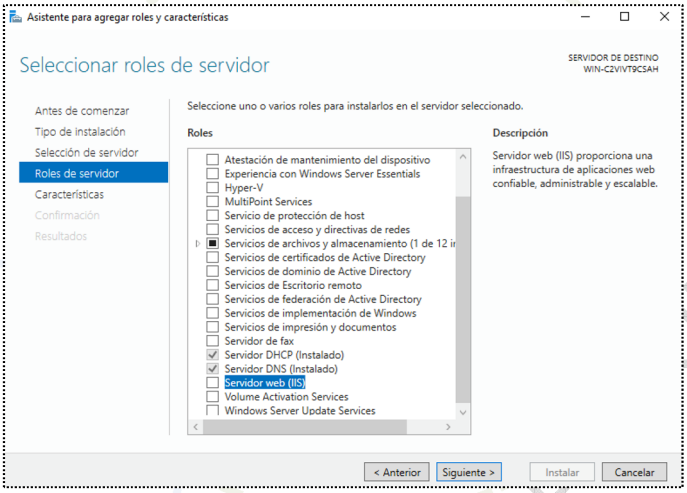
```
IIS 6.0          Windows Server 2003
IIS 6.0          Windows Server 2003 R2
IIS 6.0          Windows XP Professional x64
IIS 7.0          Windows Server 2008 / Windows Vista
IIS 7.5          Windows Server 2008 R2 / Windows 7
IIS 8.0          Windows Server 2012 / Windows 8
IIS 8.5          Windows Server 2012 R2 / Windows 8.1
IIS 10.0         Windows Server 2016 / Windows 10
```

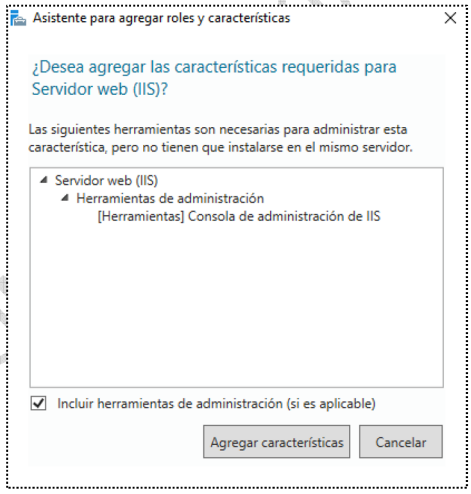
Además de servir http/https también incluye servicios FTP, SMTP y NNTP. Se basa en varios módulos que le dan capacidad para procesar distintos tipos de páginas. Por ejemplo, Microsoft incluye los de Active Server Pages (ASP) y ASP.NET. También pueden ser incluidos los de otros fabricantes, como PHP o Perl.

VERSIÓN IMPRIMIBLE - UMMO LINKIAFP

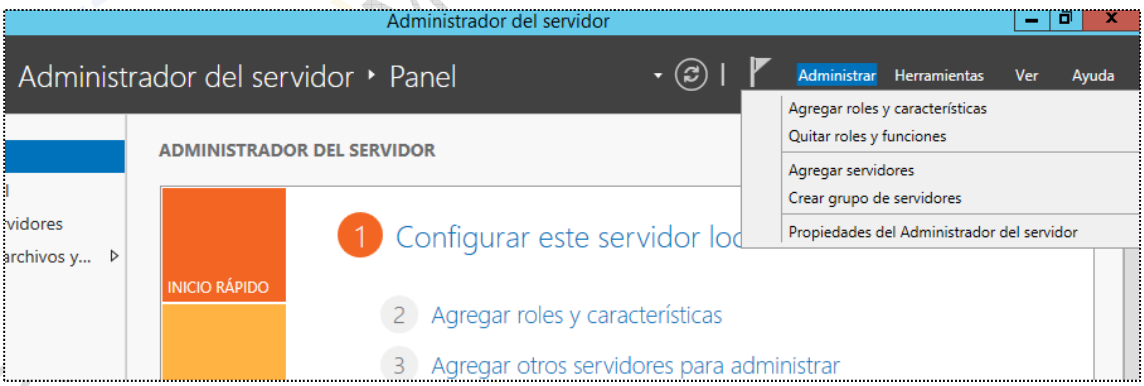
2.2.2.1. Instalación y configuración básica

IIS está integrado en los sistemas operativos Windows, pero normalmente hay que agregar el rol correspondiente para ponerlo en marcha.
En los siguientes procedimientos se mostrarán los pasos en un sistema operativo Windows Server 2016 con IIS 10.



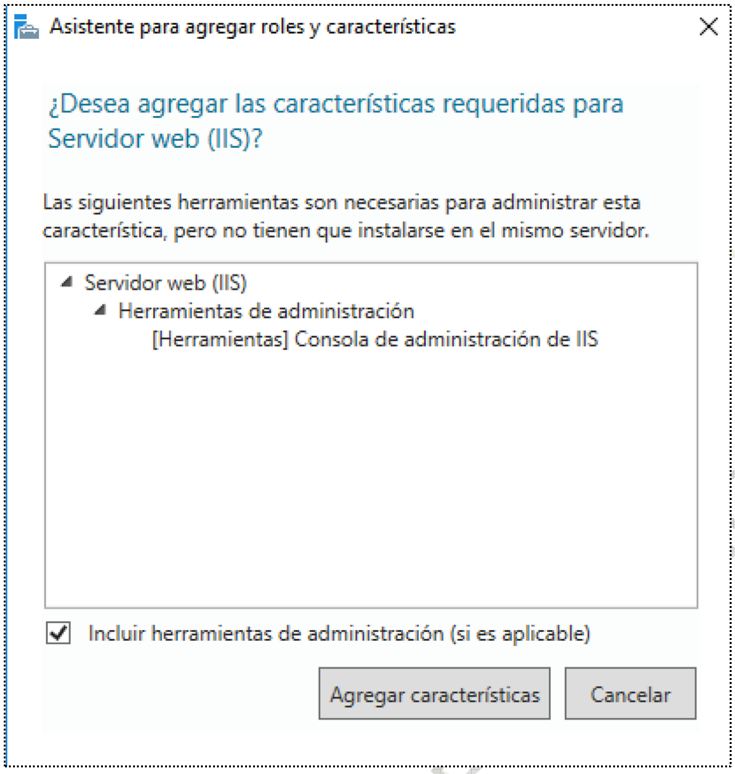


Para agregar el rol se puede hacer desde el "Administrador del Servidor" tal como se puede observar en la siguiente imagen:



Arriba a la derecha, en la opción Administrar se encuentra la posibilidad de agregar roles y características.
A continuación se selecciona el rol IIS (instalación basada en roles):

Al seleccionar el rol IIS también pide añadir ciertas características que se deberán aceptar:

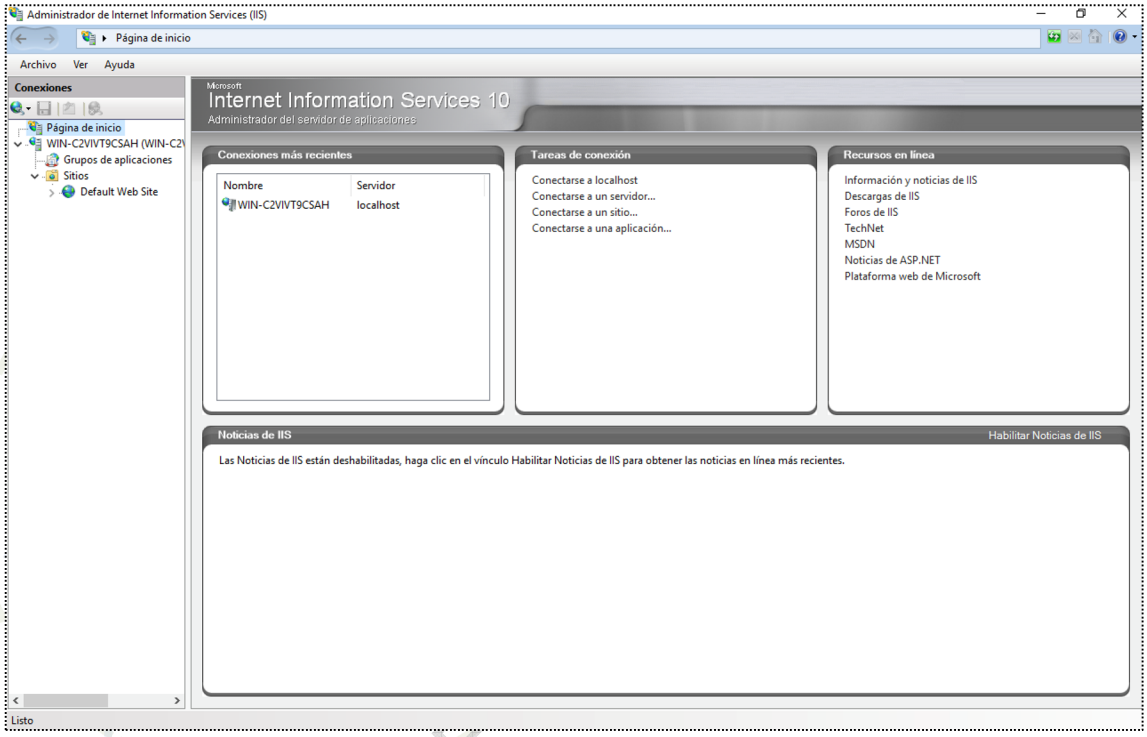


Después de pasar unas pantallas más donde se pueden agregar características adicionales, el proceso de instalación comienza.

Una vez la instalación finaliza se añade a las herramientas del administrador del servidor la herramienta del Administrador de IIS.

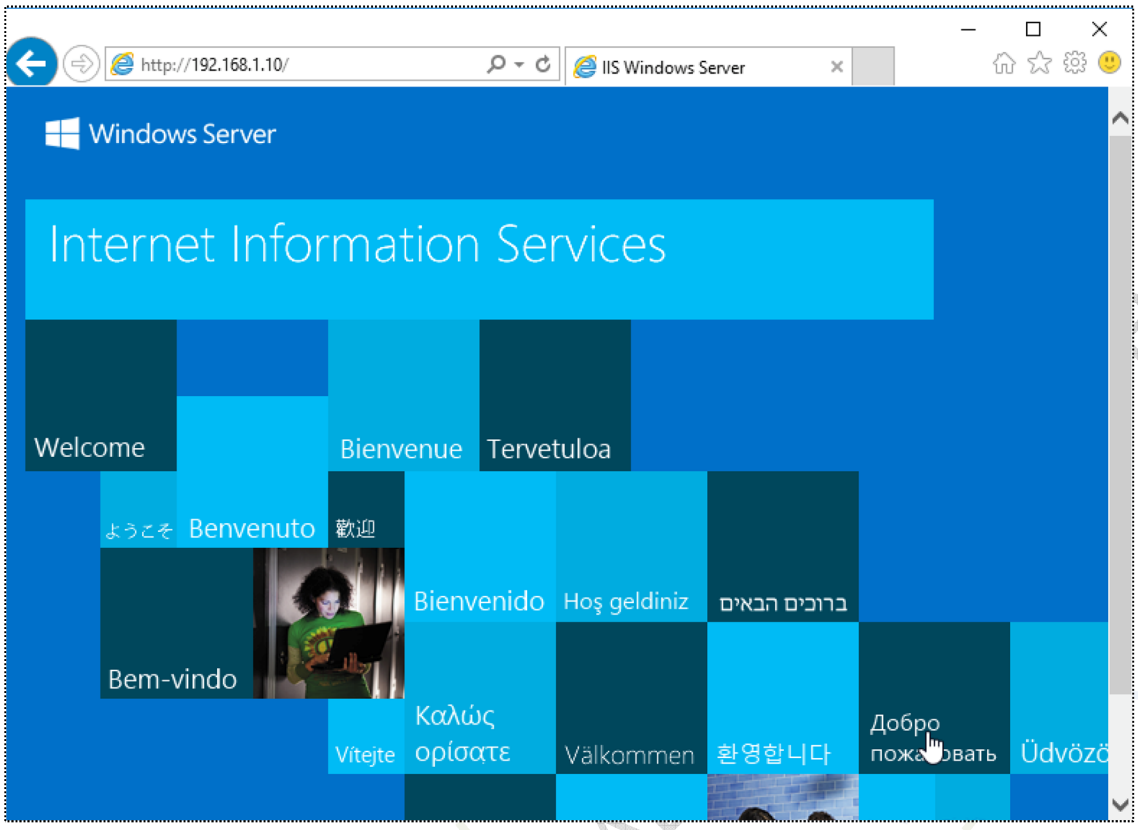
VERSIÓN IMPRIMIBLE
LINKIAFP

Dicha herramienta es el principal punto de configuración del servicio, donde por ejemplo, se puede observar que el servicio se instala con un sitio web llamado "Default Web Site".

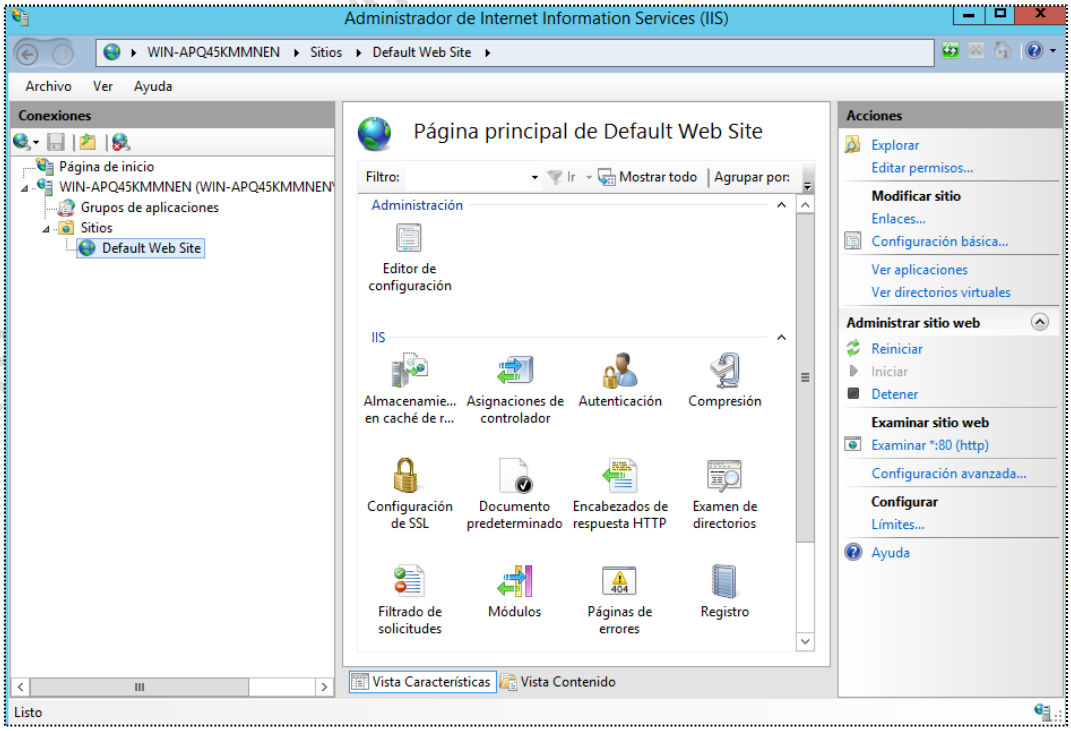


El Default Web Site está operativo desde el primer momento y podemos comprobar su correcto funcionamiento accediendo desde un cliente externo al servidor, con un navegador web y conociendo la IP del servidor.

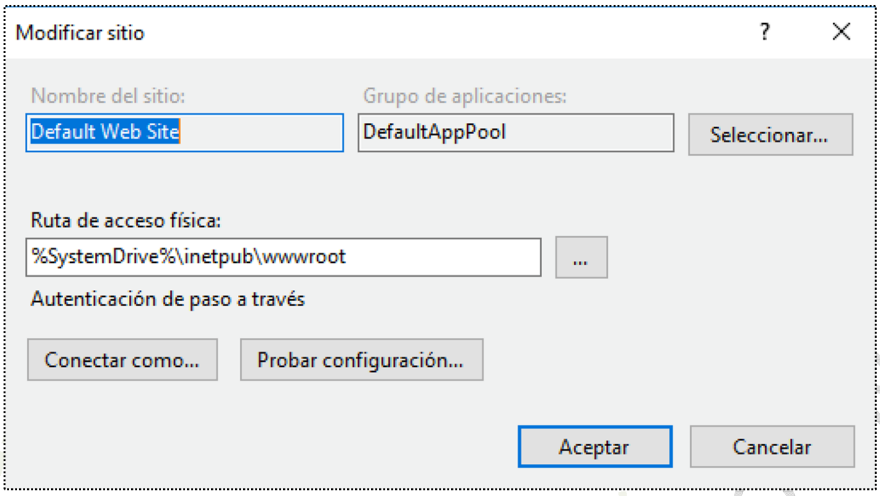
VERSIÓN IMPRIMIBLE



Se puede observar la página de inicio configurada en el Default Web Site.
Si se selecciona el Default Web Site aparecen las opciones de configuración del Site.

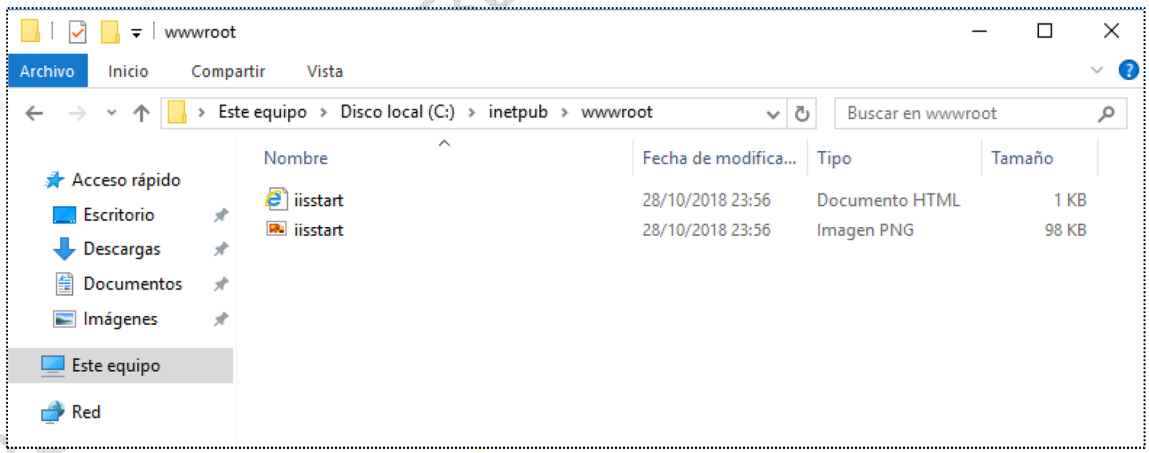


En la columna de la derecha (Acciones) aparece la opción "Configuración básica", donde se puede ver la ubicación de los archivos del sitio web:

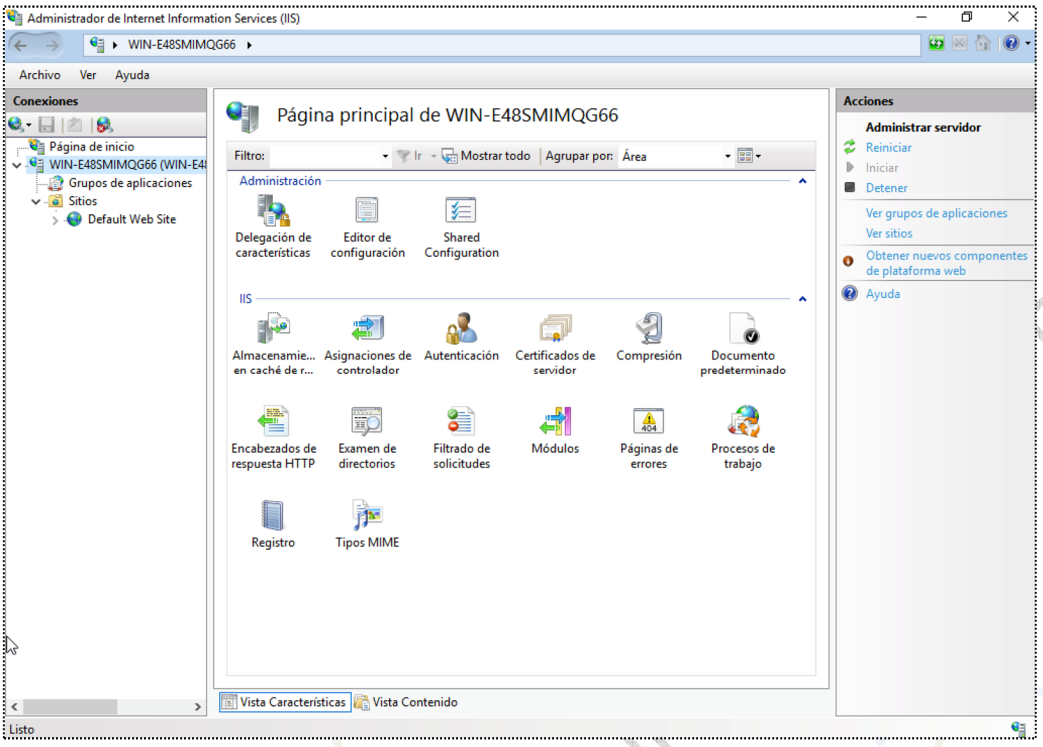


Como se puede ver la ubicación configurada de los archivos web del site es "C:\inetpub\wwwroot".
El equivalente a este directorio es configurado en Apache mediante la directiva DocumentRoot

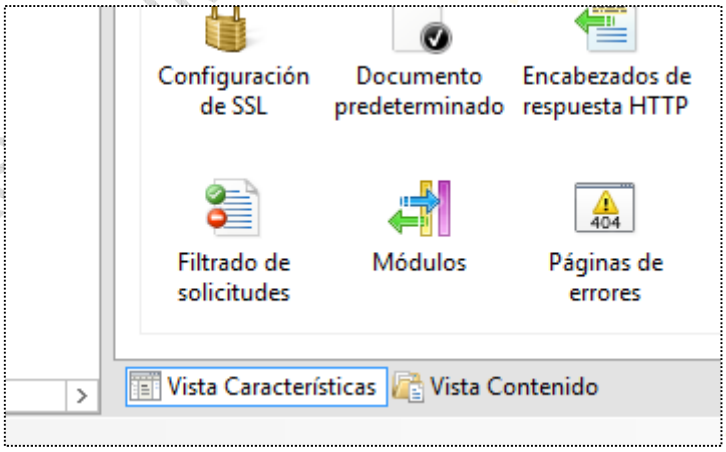
Para comprobar la existencia de los archivos es necesario navegar por el disco con el explorador de Windows.



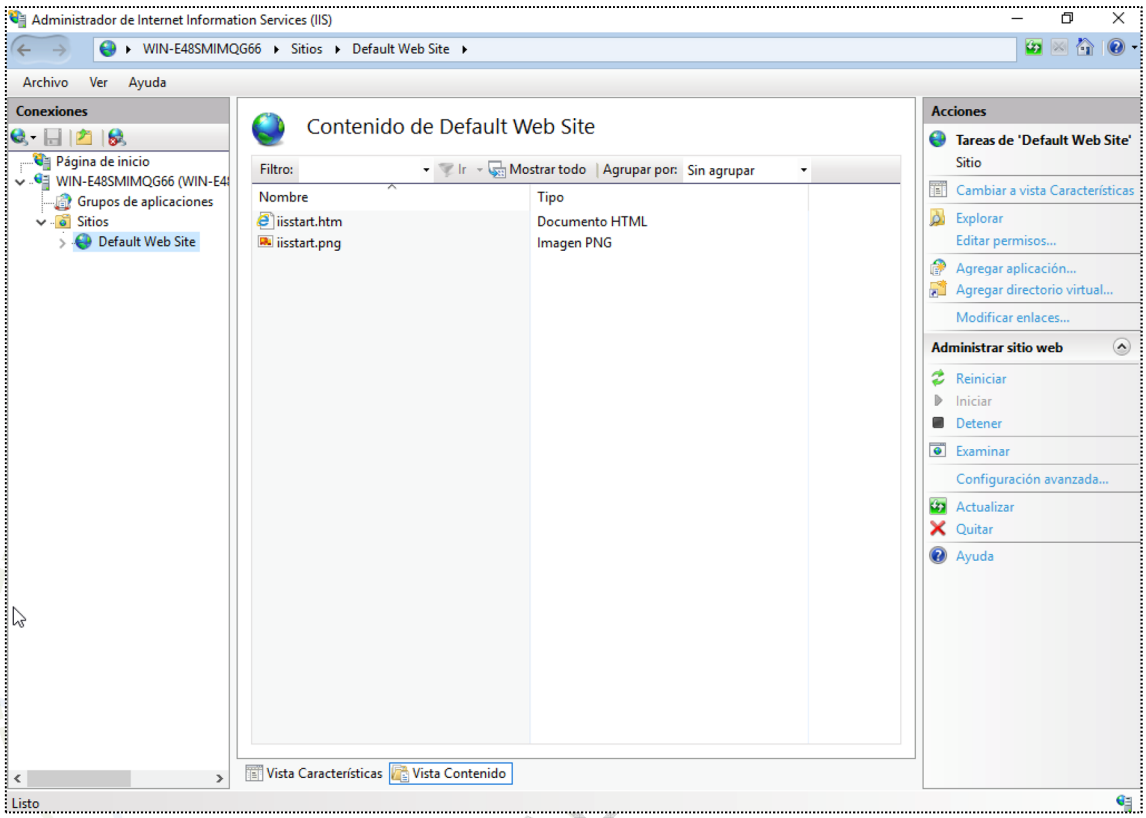
Seleccionando en la herramienta el nombre del servidor se pueden ver las opciones de configuración general del servidor:



En la columna de la derecha (Acciones) están los botones para parar, iniciar y reiniciar el servicio.
Desde la sección central se configuran aspectos como la autenticación, certificados o documentos predeterminados.
De cada web site se dispone de dos vistas, la vista de características y la vista de contenidos. Mediante la vista de características se accede a las opciones de configuración, mientras que con la vista de contenido se accede a los contenidos propios del site, navegando por los archivos web directamente desde la herramienta de IIS.
Para cambiar de una vista a otra existen unos botones en la parte inferior:



Con la Vista Contenido activada podemos ver directamente los archivos del site:



2.2.2.2. Sites

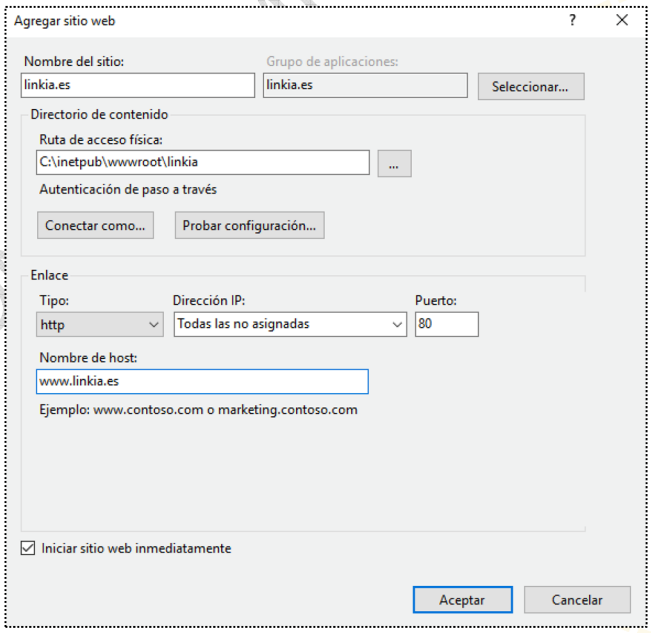
Los *sites* de IIS son los equivalentes a los Virtual Hosts de Apache. En cada *site* se puede servir un contenido diferente con una IP, y puerto específicos. También se le puede asignar un nombre DNS a cada *site* para diferenciarlos en caso de que utilicen la misma IP y puerto.

VERSIÓN IMPRIMIBLE AL

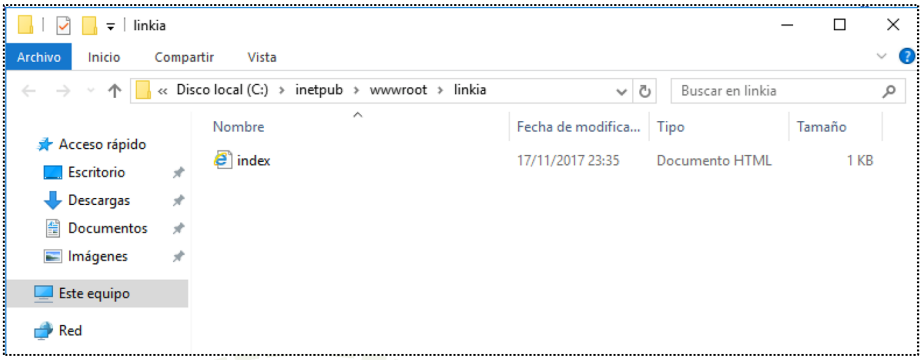
Para crear un nuevo site se puede elegir la opción de "Agregar nuevo site…" que surge con el botón derecho en el apartado Sites

Para agregar un site se tendrá que especificar:

- Un nombre descriptivo del site.
- Una ruta en el disco donde se encontrará el contenido web.
- Un enlace: que incluye la IP, el puerto y un nombre de host



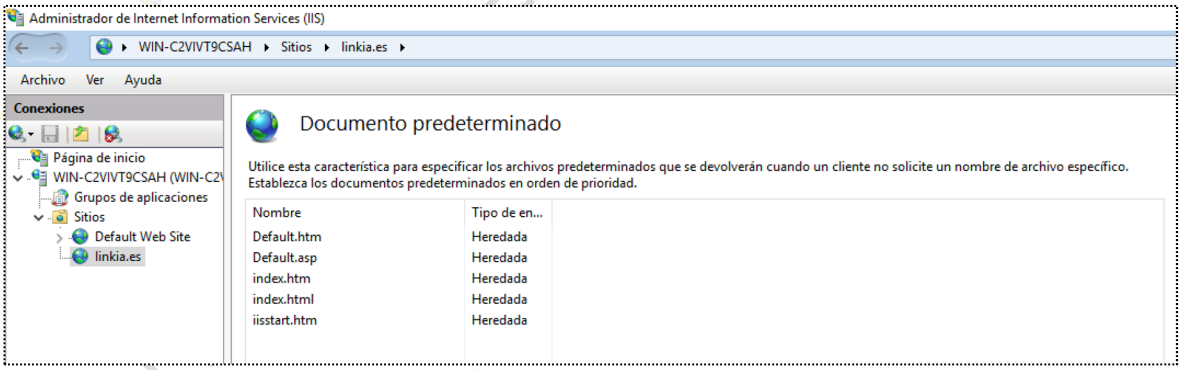
VERSIÓN ALUMNO LINKIAFP



Se debe tener en cuenta que el Default Site tiene asignadas todas las IPs del servidor por el puerto 80, y no tiene asignado un nombre de host. Si se crea un site que coincida con estas características el sistema no podrá arrancar los dos sites. Creará el site pero parará uno de los dos. Para tener varios sites en el servidor no pueden coincidir en la IP y puerto por el que se sirven, y si coinciden, no pueden tener el mismo nombre de host.

De todos modos, si se desea hacer una configuración completamente personalizada siempre se puede eliminar el Default Web Site, o cambiar su configuración.

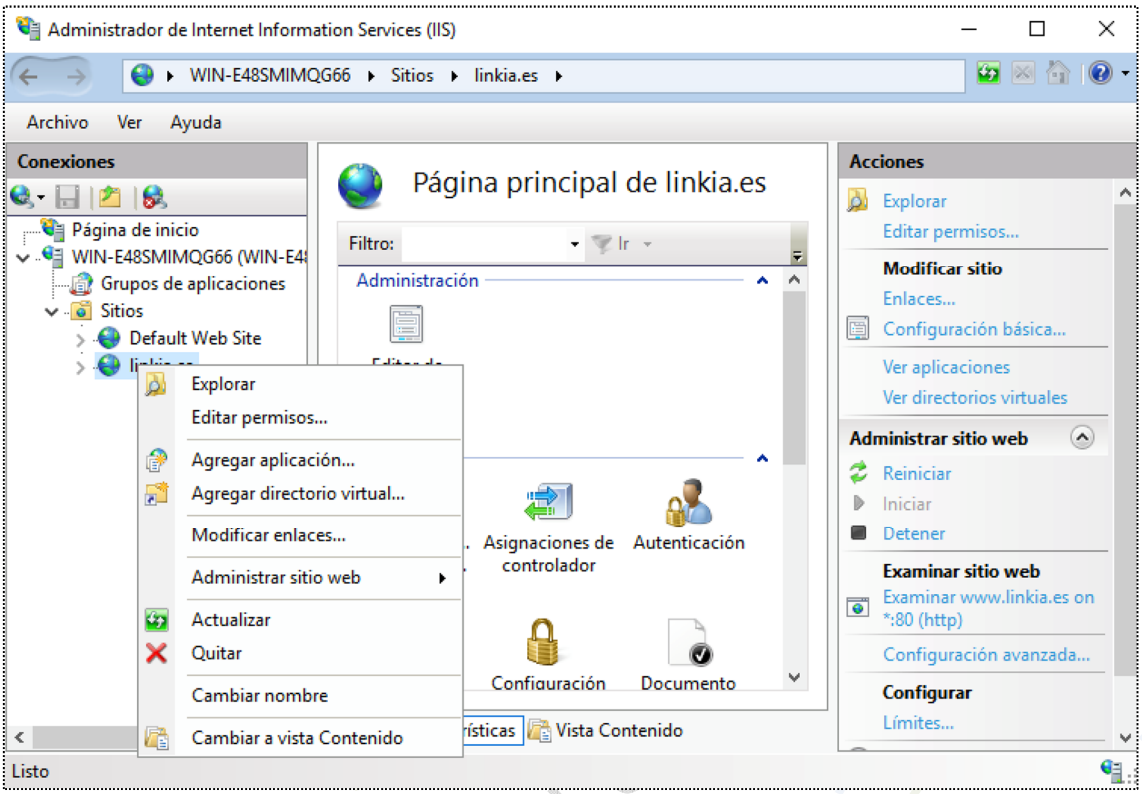
En la imagen siguiente se observa el segundo site de nombre linkia.es, ya creado.



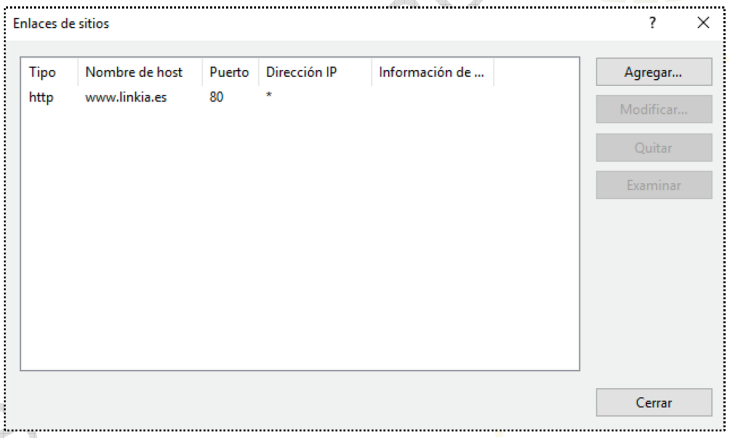
Nótese como los dos sites están siendo servidos por todas las IPs del servidor (*) y por el puerto 80. Esta coincidencia no es problema ya que el site1 (linkia.es) está diferenciado gracias a un nombre de host diferente asociado: www.linkia.es.
Cabe recordar que asignar un nombre de host al site no consigue por sí solo que los clientes lleguen al servidor a través de ese nombre. Eso sólo lo puede conseguir un DNS externo al que preguntan los clientes cuando buscan ese nombre de host y les dirige a la IP del servidor.

Una vez creado el site pueden modificarse sus características de enlace con la opción "Modificar Enlaces", accesible desde el botón derecho en el nombre de site:

VERSIÓN ALUMNO LINKIAFP



La ventana de modificar enlaces ofrece la posibilidad de asignar varios enlaces (IPs, puertos y nombres de host) a un mismo site.

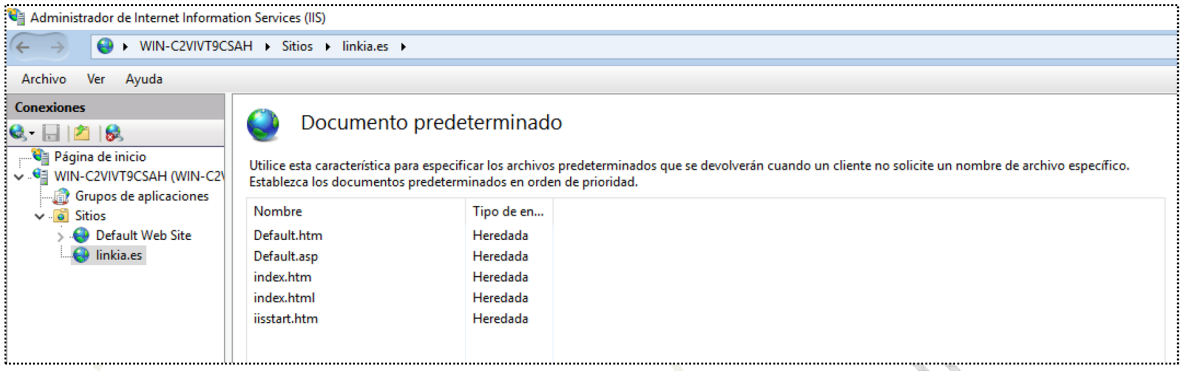


Mediante el botón "Agregar" se podrían asignar más enlaces al site. El enlace no es más que la configuración de la "puerta" que dará acceso al contenido web del site. Un site puede tener diversas "puertas" de entrada.

1.1.1.1. Documentos predeterminados

Cuando el cliente no especifica una ruta concreta y sólo escribe el nombre de host para acceder al servidor web, el servicio IIS busca documentos en el directorio configurado como ruta física del site, visto anteriormente.

Si el cliente no especifica un archivo concreto en su petición el servidor buscará los nombres de archivo configurados en la opción "Documentos Predeterminados", que se puede ver a continuación:

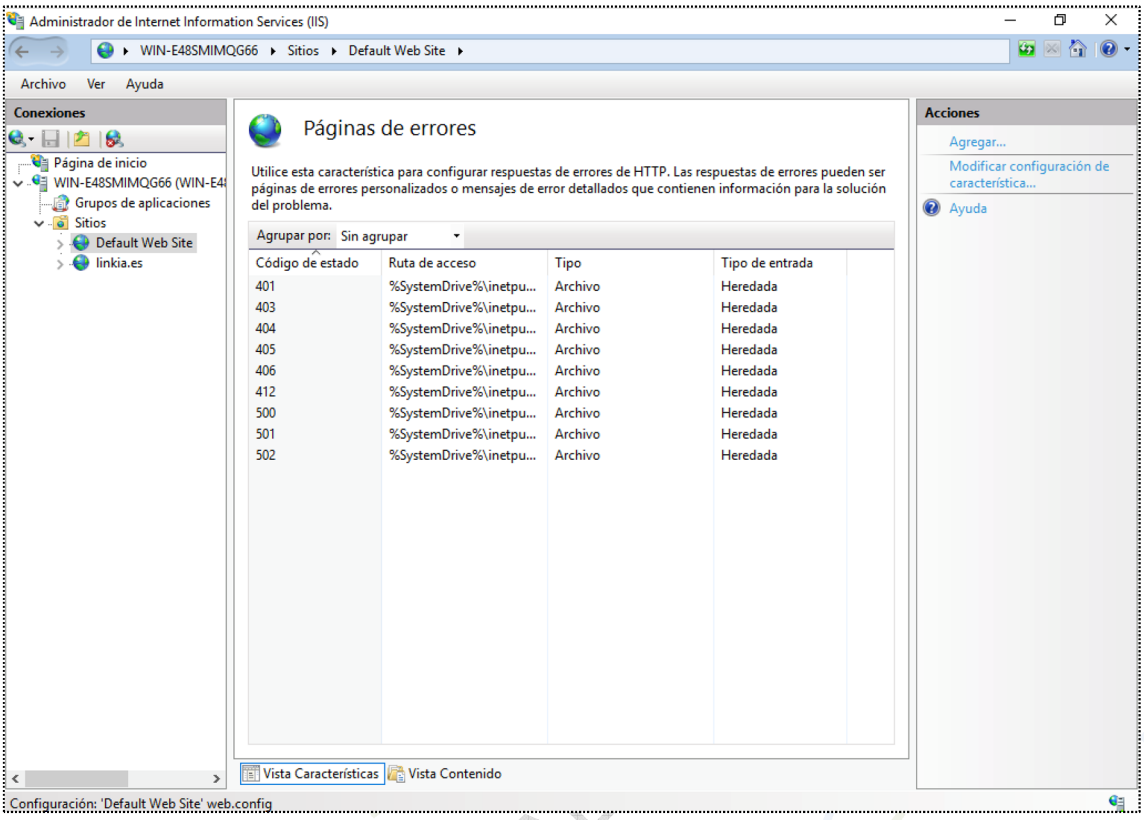


El servicio buscará los documentos con los nombres mostrados y en el orden mostrado (el orden es modificable). Los documentos predeterminados se pueden configurar a nivel general del servidor o a nivel concreto por cada site.
El fichero que existe para el default site se llama iisstart.htm. Gracias a la configuración que se puede observar en la imagen anterior, donde el nombre iisstart.htm está incluido como documento predeterminado, al poner tan solo la IP del servidor (sin especificar nombres de archivo ninguno) se puede acceder al contenido correctamente sin especificar el nombre del archivo iisstart.htm.

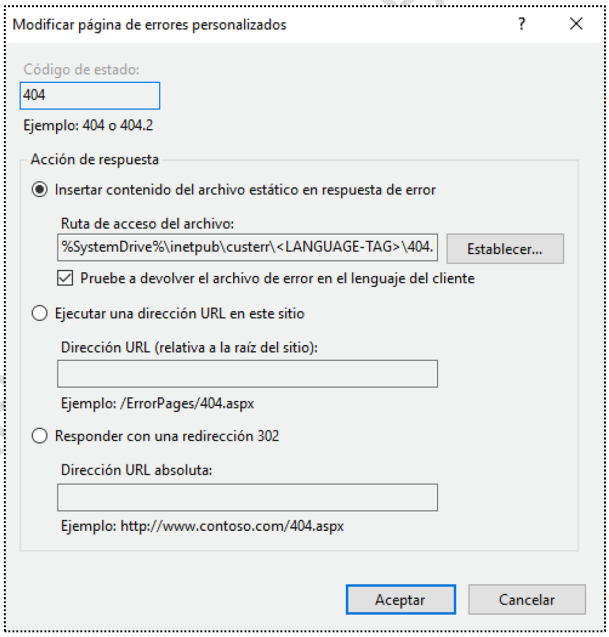
1.1.1.2. Páginas de errores

Mediante la opción "páginas de errores" presente a nivel de configuración del servidor o del site, se pueden personalizar las páginas de error, dependiendo de los códigos de error establecidos por el protocolo HTTP. Se puede establecer una página personalizada para cada tipo de error:

VERSIÓN IMPRIMIBLE ALUMNO LN



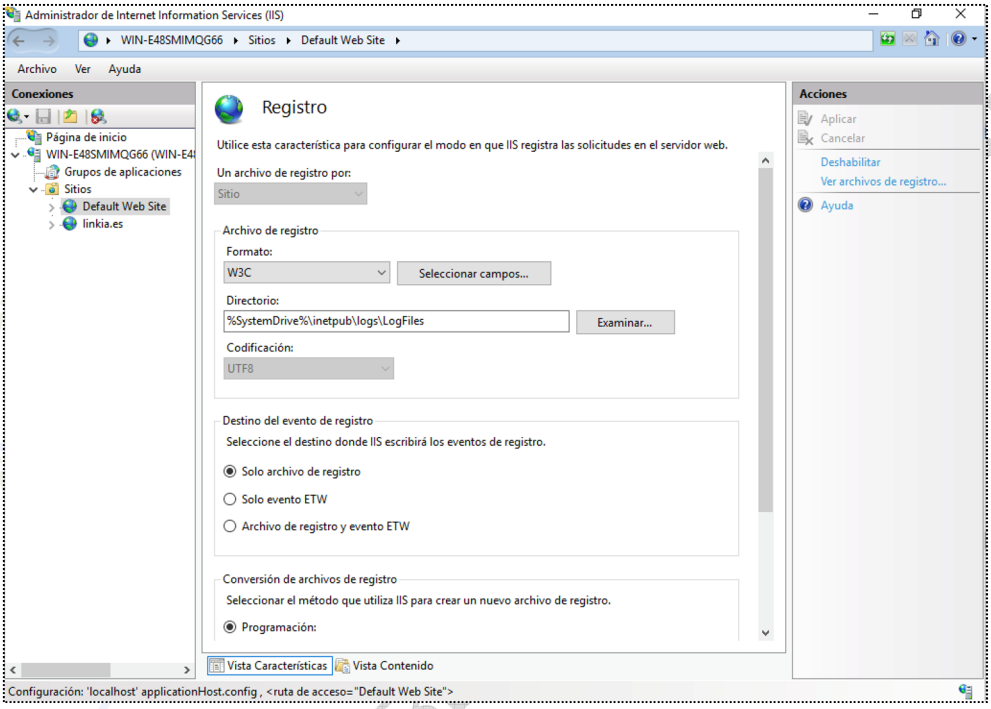
Aunque ya existen plantillas predeterminadas para los errores, se pueden personalizar las páginas para cada código de error:



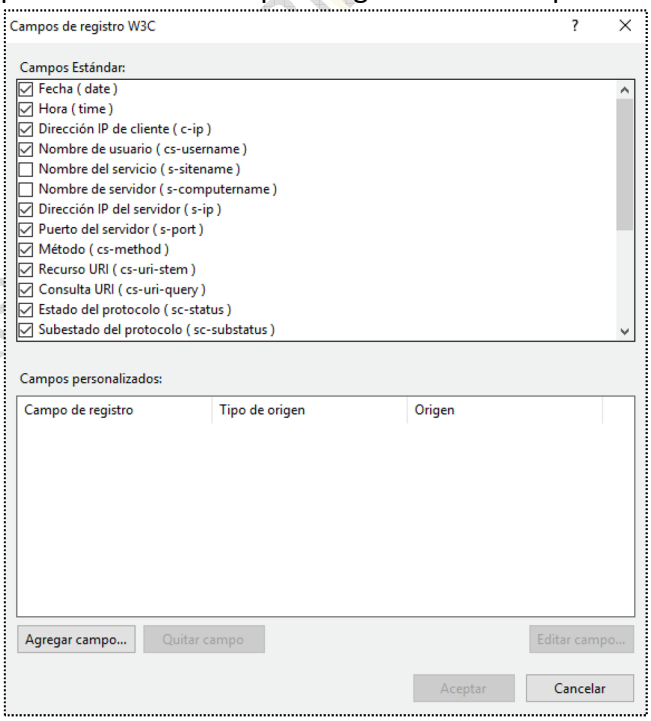
Se puede proporcionar un archivo html directamente para mostrarlo en caso de error, o se puede redirigir la petición a otra URL.

1.1.1.3. Registro

El registro de la actividad del servidor también es configurable tanto a nivel general de servidor o a nivel concreto de cada site.
Es configurable el formato del archivo de registro, su ubicación y los campos que constarán en la información registrada:



Con el botón "Seleccionar campos" se pueden ver los numerosos campos que se pueden seleccionar para registrar de cada petición y respuesta del servidor:



Se puede observar el listado de campos disponibles para el formato de registro W3C.

1.2. Seguridad

Si se quiere dotar a los servicios Web de seguridad es necesario, principalmente, asegurarse que las comunicaciones entre clientes y servidor son seguras. El tráfico generado entre un servidor web remoto y el cliente web correspondiente puede ser inspeccionado por cualquier persona con acceso a la red que los une. Se debe asegurar que aunque ese tráfico sea accesible, no sea legible para personas no autorizadas.

Se aplicarán técnicas criptográficas para encriptar y firmar digitalmente el tráfico entre cliente y servidor, de forma que se pueda asegurar la confidencialidad y autenticidad de las comunicaciones web. Tal como se ha comentado de forma introductoria en capítulos anteriores, los protocolos SSL/TLS serán los encargados de esta tarea. El uso de estos protocolos implicará también la instalación de certificados digitales.

VERSIÓN IMPRIMIBLE ALUMNO LINKAR

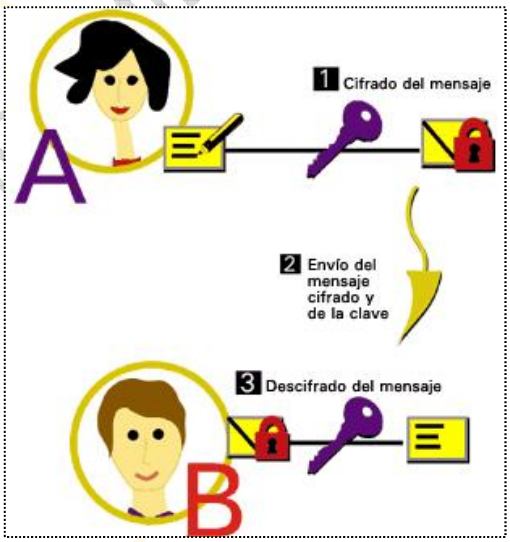
1.2.1. Principios de criptografía

La criptografía consiste en numerosas técnicas matemáticas complejas que pueden cumplir con diferentes necesidades específicas, las principales son:

- Confidencialidad: hacer que la información sólo sea inteligible por los usuarios autorizados y quede oculta para el resto.
- Autenticación: determinar la identidad de la persona o entidad con la que se está comunicando.
- Integridad: asegurar que el mensaje recibido es idéntico al enviado y no ha sido modificado en el camino.
- No repudio: en una comunicación, dotar al receptor de la capacidad de asegurar que el emisor ha sido el autor del mensaje recibido, y que el emisor no lo pueda negar.

1.2.1.1. Criptografía de clave simétrica

La forma más sencilla de encriptar información para proporcionar confidencialidad es aplicar al mensaje una clave simétrica. Una clave simétrica es aquella que se utiliza por igual tanto para encriptar la información (emisor) como para desencriptarla (receptor).



https://www.cert.fnmt.es

Este sistema de encriptación es simple y por lo tanto rápido para tratar grandes cantidades de información. Sin embargo, tiene el inconveniente de que además del mensaje encriptado, también es necesario hacer llegar al receptor la clave de encriptación, y cuando no se dispone de un canal seguro para realizar el envío de la clave, ello supone un riesgo.

VERSIÓN IMPRIMIBLE ALUMNO LINKAFT

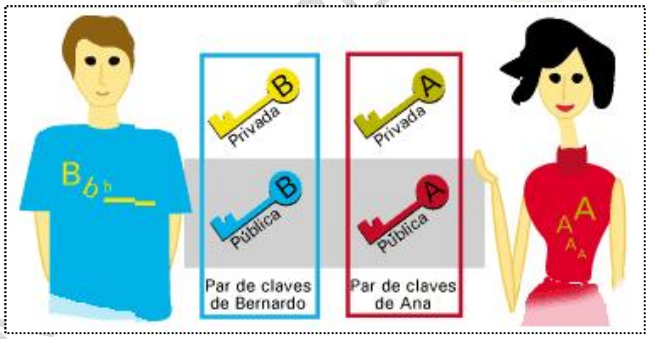
Para superar esta limitación se plantea el uso de claves asimétricas.

1.2.1.2. Criptografía de claves asimétricas

Cuando no se dispone de un canal seguro y fiable, el sistema de claves simétricas no es suficiente. Se plantea entonces el uso de claves asimétricas, pares de claves de forma que lo que encripte una clave, sólo lo desencripte la otra.
Con criptografía asimétrica a cada usuario se le asignan un par de claves: una clave pública y una clave privada. Las claves cumplen con las siguientes condiciones:

- Lo que encripta una clave sólo lo desencripta la otra.
- La clave privada es privada, y debe ser custodiada por el usuario, sin que ningún otro usuario nunca la pueda llegar a conocer.
- La clave pública debe ser conocida por todos los usuarios que se quieran comunicar con el propietario de la clave. Cada usuario debe publicar su clave pública para que hagan uso de ella los demás.
- A partir de una clave no se puede deducir la otra

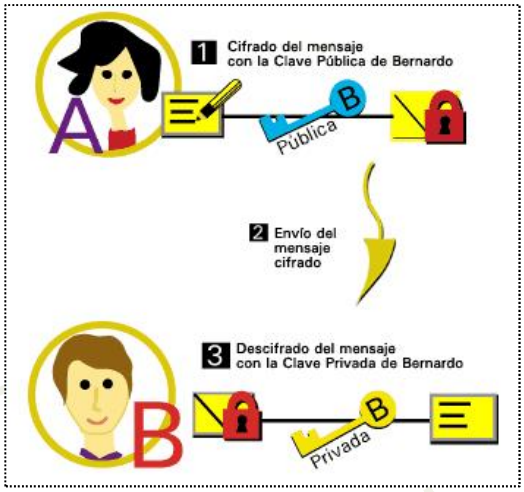
En la imagen se observan dos usuarios, cada uno con su par de claves:



https://www.cert.fnmt.es

La clave privada nunca se comparte con nadie, mientras que la pública se comparte con todos.

Cuando A quiere enviar un mensaje a B de forma privada (encriptado), A utiliza la clave pública de B (que es pública y puede acceder a ella) para encriptar el mensaje. En ese momento la única clave capaz de desencriptar ese mensaje es la clave privada de B, y B es el único que la tiene (porque es privada y él mismo la custodia), así que se puede asegurar la confidencialidad del mensaje enviado a B.



https://www.cert.fnmt.es

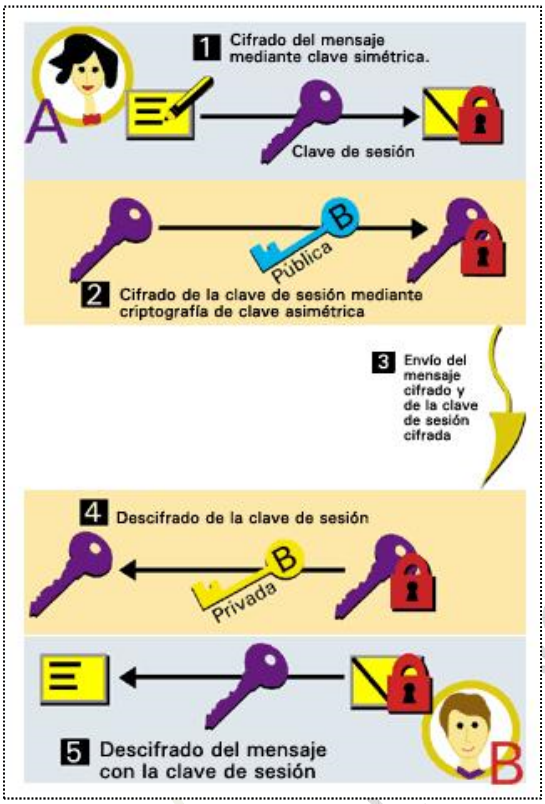
La ventaja de este sistema es que ya no se necesita un canal seguro para enviar la clave. La clave pública es compartida sin problemas a nivel público y por cualquier canal.

El inconveniente es que a nivel de proceso los mecanismos de encriptado son más complejos y por lo tanto se requiere más tiempo y capacidad de proceso para tratar grandes cantidades de información.

Habitualmente se mezclan los dos sistemas de encriptado, simétrico y asimétrico, para conseguir la solución más eficiente. Se encripta el mensaje mediante clave simétrica, que es rápido y eficiente. La clave simétrica utilizada es conocida como la clave de sesión y es generada aleatoriamente. A continuación se encripta la clave simétrica con la clave pública del receptor del mensaje. Esto es más eficiente ya que no se tiene que encriptar con clave asimétrica todo el mensaje, tan sólo se encripta la clave simétrica, que en tamaño es pequeña y por lo tanto se hace de forma rápida. Después se puede enviar la clave simétrica encriptada con la clave pública a través de un canal no seguro. El receptor recibe el mensaje y la clave. Desencripta la clave simétrica con su clave privada, y dicha clave simétrica la aplica al mensaje para desencriptarlo.

En definitiva, la combinación de los dos métodos es la mejor opción. DES, 3DES y AES son ejemplos de algoritmos de encriptación simétrica. RSA es un ejemplo de algoritmo de encriptación asimétrica.

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP



https://www.cert.fnmt.es

1.2.1.3. Firma digital

La criptografía de clave pública y privada (asimétrica) ofrece la posibilidad de generar firmas digitales, que tienen el mismo propósito que las firmas tradicionales, pero que aportan seguridad extra en el entorno digital.

Si el usuario A encripta (firma) un mensaje con su propia clave privada, ese mensaje sólo será visible con la clave pública del usuario A. Asumiendo que el usuario A es el único que posee su clave privada, cualquiera puede comprobar, con la clave pública de A, que el mensaje es efectivamente de A. Es decir, cualquiera puede certificar que el emisor del mensaje ha sido A, y además, A no puede negar que ha sido el autor/emisor del mensaje. A su vez, también se garantiza que el mensaje no ha sido modificado desde su firma.

En definitiva, gracias a la firma digital se pueden garantizar las necesidades de autenticación, integridad y no repudio de un mensaje.

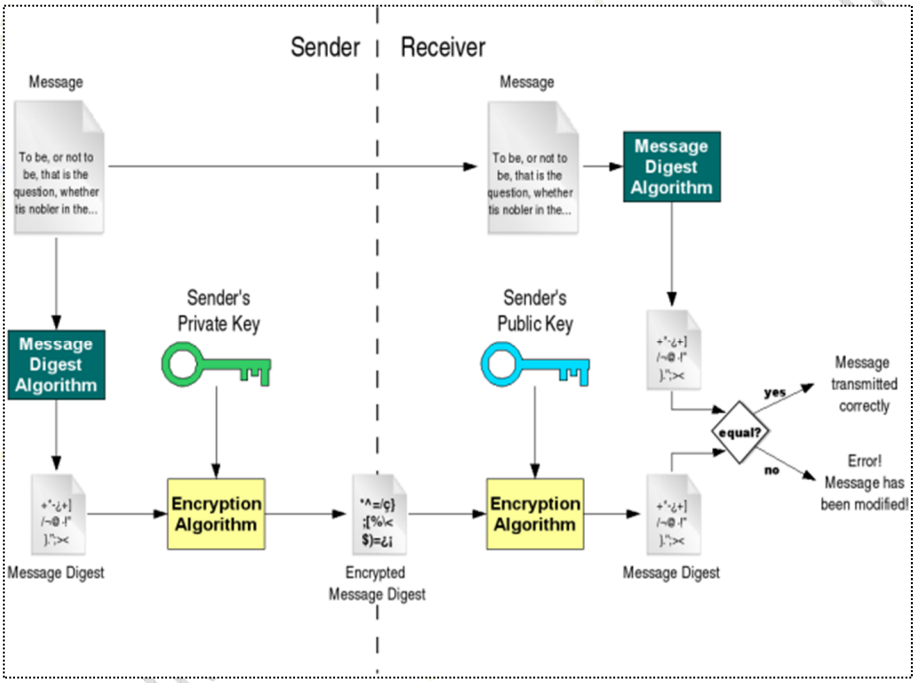
Sin embargo, tal como se ha comentado previamente, los algoritmos de encriptación asimétrica son lentos y costosos en cuanto a proceso y tratamiento de datos. Es por eso que no se "firma" digitalmente todo el mensaje, en su lugar, se genera primero un "resumen" del mensaje llama *hash* o *digest.*
El "resumen" o *hash* del mensaje se genera mediante funciones de *hash*, que son estándar y bien conocidas. MD5 o SHA son ejemplos de algoritmos de *hashing.* El

VERSIÓN IMPRIMIBLE AL CURSO LINKIAFP

hash resultante de un mensaje es de longitud fija (sea cual sea la longitud del mensaje) y se debe cumplir que:

- A partir del *hash* no se pueda deducir el mensaje.
- Un mensaje concreto siempre genere el mismo *hash.*
- *Dos* mensajes diferentes nunca (o casi nunca) generen el mismo *hash*

Ahora sí, es mucho más eficiente firmar con la clave privada del usuario emisor tan sólo el *hash,* y no todo el mensaje. De esa forma, el *hash* firmado por la clave privada del usuario se convierte en una firma digital, que cualquier otro usuario que tenga acceso a la clave pública podrá comprobar.
A continuación se presenta un esquema funcional del proceso explicado:



http://www.ccfit.nsu.ru/

La firma digital se genera en dos pasos:

1. Se genera un *Message Digest* o *Hash*, mediante algoritmos de hashing. Es un resumen del mensaje original.
2. El "digest" o *hash* es encriptado con la clave privada del transmisor y adjuntado al mensaje.

El receptor desencripta la firma digital mediante la clave pública del transmisor para obtener el digest.

Por otro lado aplica el algoritmo de hashing al mensaje original, que también recibe sin encriptar.

Comparando los dos digest puede decidir si el mensaje ha cambiado en el camino. Si son diferentes digest, el mensaje no es el mismo que se envió -> integridad.

Por otro lado, el receptor se asegura de que lo ha enviado un transmisor concreto (autenticación) ya que su clave pública funciona para desencriptar la firma digital.

Nótese que en este ejemplo no hay confidencialidad, ya que el mensaje también se envía de forma visible junto a la firma digital, tan solo se requiere integridad y autenticación. Si además se deseara confidencialidad se tendría que encriptar el mensaje también con la clave pública del receptor.

1.2.1.4. Certificados digitales

A pesar de todo lo explicado sobre seguridad hasta el momento, las soluciones planteadas de criptografía asimétrica siguen presentando alguna debilidad.

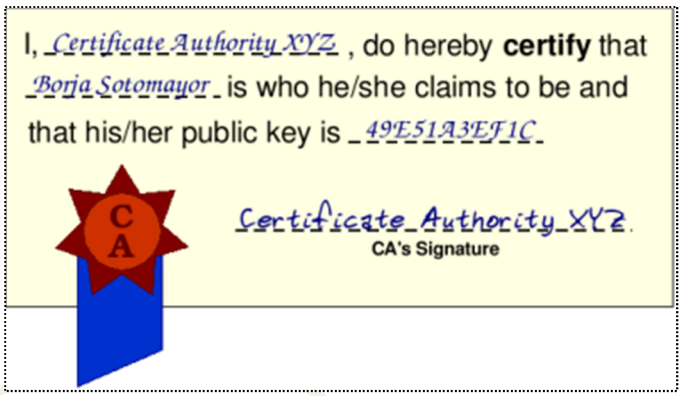
La debilidad radica en el método empleado para que los usuarios compartan sus claves públicas. Para que todo funcione, es necesario que un usuario haga pública su clave pública para que los demás hagan uso de ella. En este punto, un usuario malintencionado podría sustituir la clave de un usuario A en el momento de hacérsela llegar a los demás.

Si el usuario A quiere que le envíen información encriptada, debe hacer llegar a los demás usuarios su clave pública. Si en el momento de enviársela (no se dispone de un canal seguro) un usuario malintencionado intercepta el envío y pone su clave pública en lugar de la del usuario A, entonces los demás usuarios estarán encriptando con la clave pública del usuario malintencionado sin darse cuenta. Eso supone que el usuario malintencionado podría después interceptar el mensaje y desencriptarlo con su propia clave privada, ya que ha sido encriptado para él sin que los emisores se enteren.

Si el usuario A, para el que se pretendía encriptar la información, recibe el mensaje, detectará que su clave privada no lo desencripta, y habrá detectado la intrusión, pero no la habrá podido evitar.
En definitiva, resulta que también es necesario un canal seguro para compartir las claves públicas entre usuarios. Pero si lo tuviéramos, sería más sencillo y efectivo aplicar criptografía simétrica!

En este punto es donde aportarán valor los certificados digitales. El certificado digital no es más que un documento digital que certifica que la clave pública de un usuario le pertenece y es auténtica.

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP



http://www.ccfit.nsu.ru/

Como se observa en la imagen representativa de un certificado digital, éste contiene el nombre de una entidad certificadora (CA, Certification Authority), que es quien emite el certificado, el nombre del usuario, su clave pública, y la firma de la entidad certificadora (firmado con la clave privada de la entidad)
De esta forma, un usuario que reciba este certificado podrá comprobar con la clave pública de la entidad certificadora (que es pública) la autenticidad del certificado. Si lo comprueba correctamente, y muy importante, pone confianza en esa entidad, entonces le queda constancia de que esa clave pública que contiene el certificado, corresponde al usuario que indica el certificado.
Con este proceso se ha conseguido hacer llegar a los usuarios remotos la clave pública de un usuario para que puedan hacer uso de ella. Sin embargo es necesario destacar diversos aspectos:

- Es necesario de un tercero (la CA) para garantizar la autenticidad de la clave pública de un usuario.
- El usuario destinatario del certificado tiene que confiar en la entidad que ha emitido el certificado para que todo funcione.

Nótese como, al final de todo el proceso, alguien tiene que poner la confianza en una CA, sin esta confianza todo se viene abajo, porque, siempre se podría pensar que un usuario malintencionado podría montar una CA y emitir certificados para usuarios. Si los usuarios confían en esa CA, ésta podría hacer un mal uso de las diferentes claves.

Las entidades certificadoras son entidades públicas o privadas que deben tener la reputación necesaria para poder emitir certificados en los que los usuarios confíen. Actualmente ser una entidad certificadora es un negocio. Los certificados se venden, y por lo tanto, como todo negocio, está expuesto a la confianza de los usuarios. También entidades de carácter público se encargan de estas tareas, como por ejemplo en España, la fábrica nacional de moneda y timbre (FNMT).

Los certificados son importantes en el protocolo TLS/SSL sobre http (https) utilizado para encriptar el envío de páginas web a los clientes, aunque también se emplea en otros ámbitos como la firmas digitales de código o el encriptado de correo electrónico.

Desde un punto de vista más técnico, los certificados no sólo contienen los elementos comentados, también tienen fecha de caducidad, versión, n. de serie, etc. Por ejemplo, se presenta a continuación un detalle de los campos contenidos en un certificado digital del tipo X.509 v3.

The structure of an X.509 v3 digital certificate is as follows:

- Certificate
  - Version
  - Serial Number
  - Algorithm ID
  - Issuer
  - Validity
    - Not Before
    - Not After
  - Subject
  - Subject Public Key Info
    - Public Key Algorithm
    - Subject Public Key
  - Issuer Unique Identifier (optional)
  - Subject Unique Identifier (optional)
  - Extensions (optional)
    - ...
- Certificate Signature Algorithm
- Certificate Signature

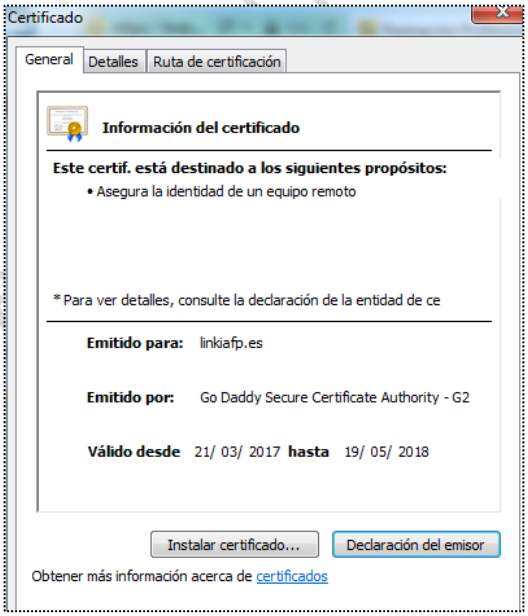
wikipedia.org

Cuando se accede a páginas web protegidas por el protocolo TLS/SSL sobre http (https), como típicamente sucede con las páginas donde los usuarios se tienen que identificar, el protocolo de encriptación asegurará que las credenciales y toda la información de la página web viaje con confidencialidad. Pero antes de eso, el servidor presentará un certificado digital al cliente para que el servidor asegure al cliente que realmente es quien dice ser.

Cuando se accede a una URL protegida mediante HTTPS, el servidor le envía al cliente un certificado digital para demostrarle quien es. El cliente, deberá decidir si confía en ese certificado, y por lo tanto en ese servidor. Este proceso lo realiza automáticamente el sistema operativo, o el navegador, pero si se confía, normalmente el navegador suele mostrar un candado cerrado y la comunicación encriptada comienza.

En el ejemplo siguiente se muestra un acceso a una web, mediante https, y que por lo tanto envía todo su tráfico encriptado.

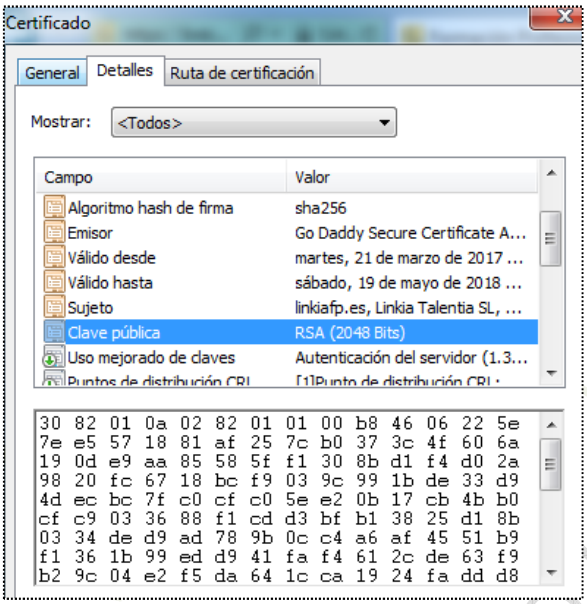
Tal como se observa, el candado ya está cerrado. Significa que el servidor web ya ha presentado el certificado y el cliente ya ha confiado en él.



Se puede inspeccionar los detalles del certificado presentado. El certificado ha sido emitido para el sitio linkiafp.es (y ningún otro) por parte de una entidad certificadora llamada Go Daddy, y tiene un periodo de validez. También tiene un propósito que es el de asegurar la identidad de un equipo remoto (el servidor web).

Para más detalles, se puede incluso observar la clave pública de la entidad, entre otros campos del certificado:

VERSIÓN IMPRIMIBLE LINKIAFP



En este caso la clave pública está generada con RSA y tiene una longitud de 2048 bits, que se muestran en la imagen en formato hexadecimal.

Cuando un cliente web (navegador) accede a un servidor que le presenta un certificado, en el cliente se comprueban tres condiciones básicas para confiar en el certificado:

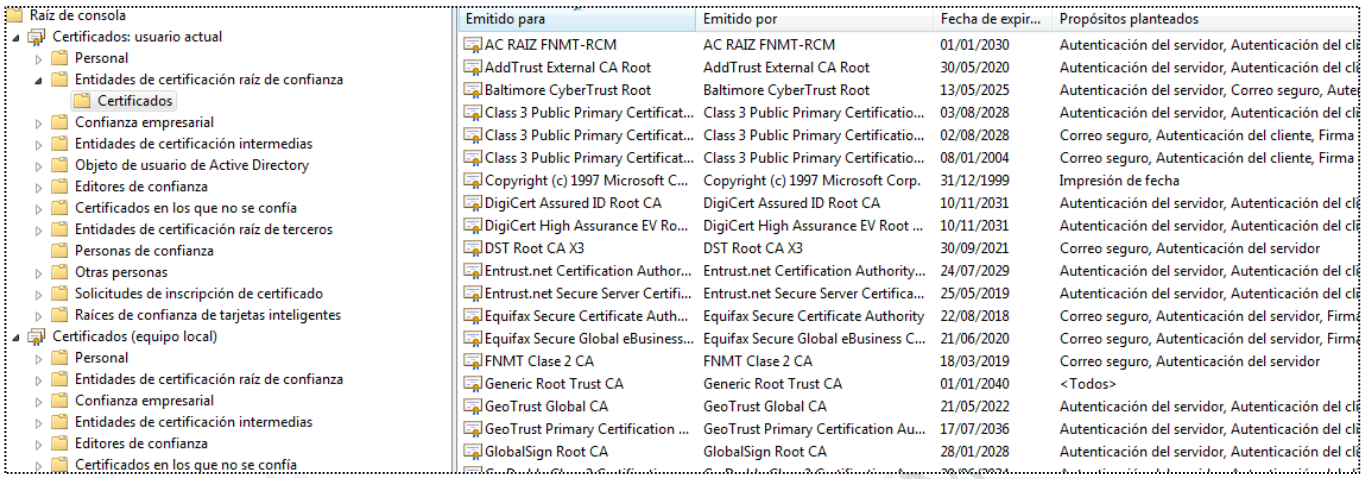
- El nombre de host que se ha escrito en el navegador del cliente coincide con el nombre asignado al certificado.
- El certificado es vigente (no caducado).
- Se confía en la entidad certificadora que emite dicho certificado.

Cuando alguno de los tres aspectos mencionados antes no se cumple, el navegador avisa de esa circunstancia y pide al usuario que decida si quiere continuar con la navegación por ese sitio web, a continuación se muestran dos situaciones diferentes de incumplimiento:

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP

Para comprobar el último punto de los tres, el cliente debe tener, en su sistema operativo, a la entidad emisora en su lista de entidades emisoras de confianza.

Esta lista está disponible en el almacén de certificados de que disponen los sistemas operativos. En un sistema Windows tiene el siguiente aspecto:



Como se puede observar en la imagen, existe una lista de entidades de confianza. Estas entidades, y la confianza en ellas, ya viene hoy en día preinstalada en muchos sistemas operativos. Es por ello que las entidades que quieren publicar sitios web en internet protegidos con TLS/SSL suelen adquirir certificados de las entidades que ya están preinstaladas como de confianza en la mayoría de sistemas operativos cliente (Windows, Linux, Mac, etc.).

Existen otros servicios web en los que no sólo el servidor presenta un certificado, sino que además se le exige al cliente que presente también otro certificado para identificar el usuario que está accediendo. Estos casos son típicos a la hora de hacer trámites con las administraciones públicas vía web.

En este caso, con el certificado de cliente se aseguran de la identidad del usuario que está accediendo. El certificado de usuario lo consigue cada usuario acudiendo en persona a la entidad que lo emita, y para garantizar que lo emite para la persona adecuada, se le exige al usuario que acuda presencialmente y presente la identificación oficial pertinente.

VERSIÓN IMPRIMIBLE ALUMNO

1.2.2. Certificados en Apache

Los certificados se compran a la entidad elegida y se instalan en el servidor web para que empiece a hacer uso de él. Asociados a los servicios de hosting suele haber disponibles servicios y productos adicionales relacionados con los certificados.

Para obtener un certificado se debe realizar una petición (CSR, Certificate Signing Request) con un formato concreto, donde constan los datos del servidor y la entidad que lo pide.

Para propósitos de test, habitualmente se generan en el propio servidor certificados auto-firmados, es decir, no hay ninguna CA externa que lo emita. El propio servidor se lo auto-emite, y como tal, es muy probable que nadie confíe en dicho servidor como entidad emisora, por lo que habrá que instalar manualmente esa confianza en todos los clientes que quieran acceder a él.

Para demostrar el proceso de instalación de un certificado en un servidor se hará con certificados auto-firmados en estos apuntes. La única diferencia en un entorno real sería el conseguir un certificado real emitido por una entidad emisora de confianza a nivel público.

El primer paso para configurar certificados en un servidor apache es habilitar el módulo correspondiente: mod-ssl.

```
sudo a2enmod ssl
```

Un ejemplo del proceso de activación del módulo:

```
root@linkia:/home/linkia# 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@linkia:/home/linkia# service apache2 restart
root@linkia:/home/linkia#
```

Después de habilitar el módulo se puede comprobar que está presente en el directorio mods-enabled. También es necesario reiniciar el servicio Apache.

VERSIÓN PARA ALUMNO LINKIAFP

Para generar el certificado auto-firmado se generará primero una clave privada, mediante la aplicación *openssl:*

```
openssl genrsa -des3 -out server.key 1024

Generating RSA private key, 1024 bit long modulus
.....................................................++++++
........++++++
e is 65537 (0x10001)
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:
```

En el ejemplo mostrado se genera una clave privada de 1024 bits que se almacena en el fichero server.key. La clave será legible en formato ASCII, y quedará asociada a un password que pide a la hora de generar la clave. Ese password (passphrase) da acceso al uso de la clave privada. Después esa clave irá asociada también al servicio Apache del sistema operativo.

A continuación se debe generar una petición CSR. Después se podrá usar esa CSR para un certificado auto-firmado o para pedir un certificado a una entidad emisora externa.

```
openssl req -new -key server.key -out server.csr

Country Name (2 letter code) [GB]:CH
State or Province Name (full name) [Berkshire]:Bern
Locality Name (eg, city) [Newbury]:Oberdiessbach
Organization Name (eg, company) [My Company Ltd]:Akadia AG
Organizational Unit Name (eg, section) []:Information Technology
Common Name (eg, your name or your server's hostname) []:public.akadia.com
Email Address []:martin dot zahn at akadia dot ch
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

La petición se hace con la clave generada anteriormente y el resultado se almacena en el fichero server.csr. Al generar la petición se piden datos del servidor para el que se desea generar el certificado. Todos son descriptivos, pero es especialmente importante el campo *Common Name*¸ en el que consta el nombre (DNS) por el que se accederá a este servidor. Es importante que el nombre que se introduzca en esta petición coincida con el nombre DNS del sitio web o servidor para el que se pide el certificado. Si no coinciden, cuando un cliente acceda mediante un nombre DNS concreto y no coincida con el nombre para el que se ha generado el certificado, el navegador generará un aviso de seguridad.

El password de la clave es importante ya que si alguien consigue acceso al servidor y al fichero de la clave podría poner en riesgo la seguridad del sistema. Sin embargo, ese password será solicitado siempre que el servicio apache sea iniciado, lo cual puede suponer un inconveniente en caso de que el servidor sufra un reinicio esporádico y no

haya un operador presente para introducir el password. En ese caso Apache no arrancaría.

De forma opcional se puede eliminar ese requerimiento de introducir el password cada vez que arranca el servicio. Se hace eliminando la encriptación de la clave:

```
cp server.key server.key.org
openssl rsa -in server.key.org -out server.key
```

Con los comandos mostrados, **server.key** queda con la clave sin encriptar, y **server.key.org** con la clave encriptada.

Para generar un certificado auto-firmado, se hará a partir de la petición (server.csr) y de la clave sin encriptar (server.key) para que no pida el password cada vez que se reinicie el servicio.

```
openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
Signature ok
subject=/C=CH/ST=Bern/L=Oberdiessbach/O=Akadia AG/OU=Information
Technology/CN=public.akadia.com/Email=martin dot zahn at akadia dot ch
Getting Private key
```

En el ejemplo mostrado se ha generado un certificado digital auto-firmado en el fichero server.crt y con una validez de 365 tal como se ha indicado en el comando de generación.

Una vez generados el certificado y la clave, estos archivos se deben colocar en una ubicación segura dentro del propio servidor. Por ejemplo, se podrían copiar en

```
cp server.crt /usr/local/apache/conf/ssl.crt
cp server.key /usr/local/apache/conf/ssl.key
```

Una vez colocados los archivos en su ubicación final, sólo falta configurar en Apache un Virtual Host que escuche por el puerto 443 (estándar para https).

Dentro del virtual host se deberán configurar las siguientes directivas:

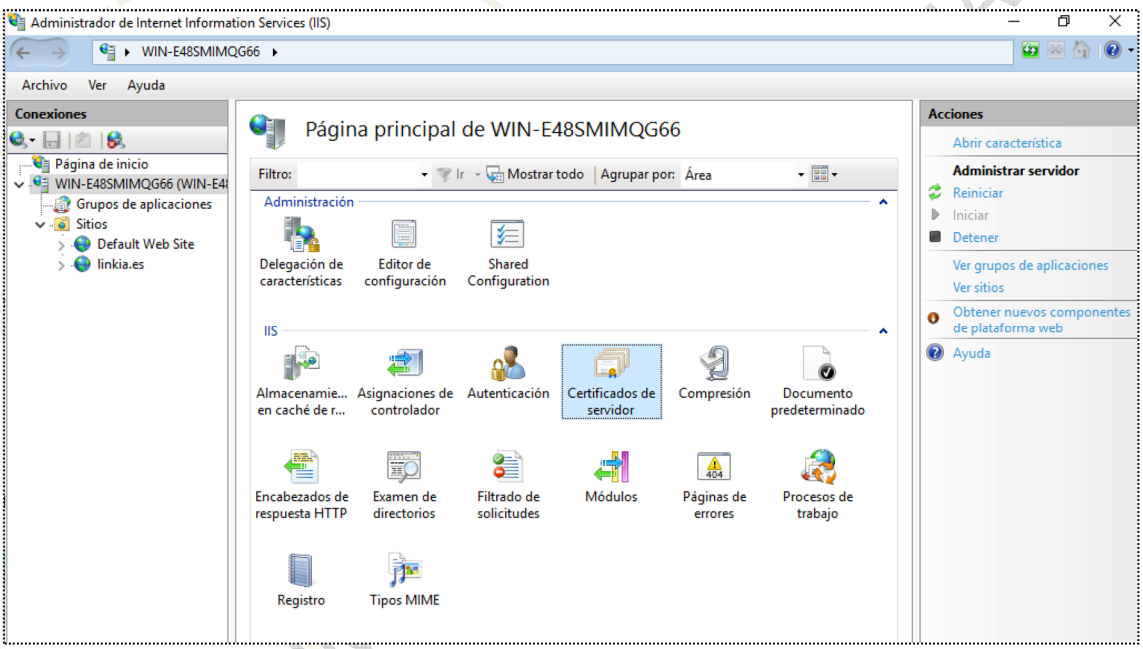
```
SSLEngine on
SSLCertificateFile /usr/local/apache/conf/ssl.crt/server.crt
SSLCertificateKeyFile /usr/local/apache/conf/ssl.key/server.key
```

Y si se desea hacer obligatorio el uso de https para el contenido se debe configurar la directiva *SSLRequireSSL*, que debe ser utilizada obligatoriamente dentro de una etiqueta <Directory>.

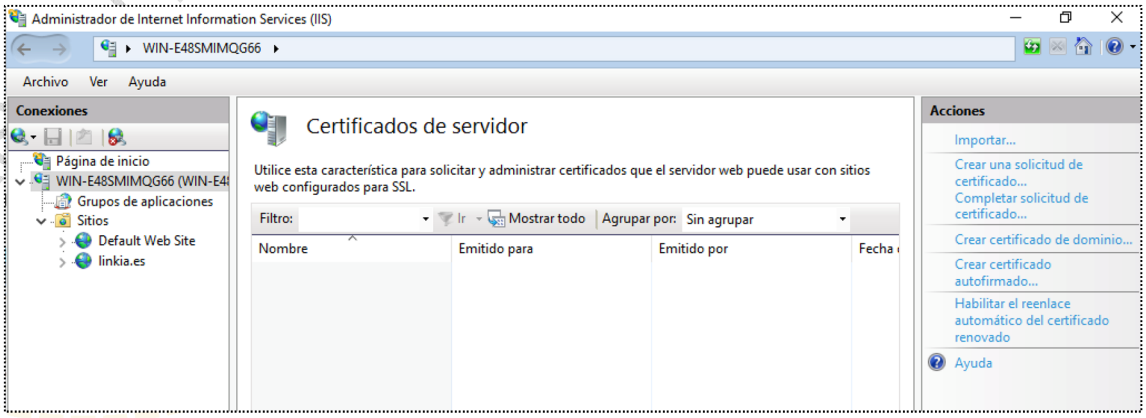
1.2.3. Certificados en IIS

Para poder establecer conexiones seguras mediante https (TLS/SSL sobre http) será necesario instalar primero un certificado en el servidor. En este caso, con motivos de testear el sistema se generará un certificado auto-firmado, aunque en un caso real se procedería a adquirir un certificado de una entidad emisora de confianza.

Desde el administrador de IIS se puede generar el certificado auto-firmado, en la página de configuración principal está la opción "Certificados del servidor".

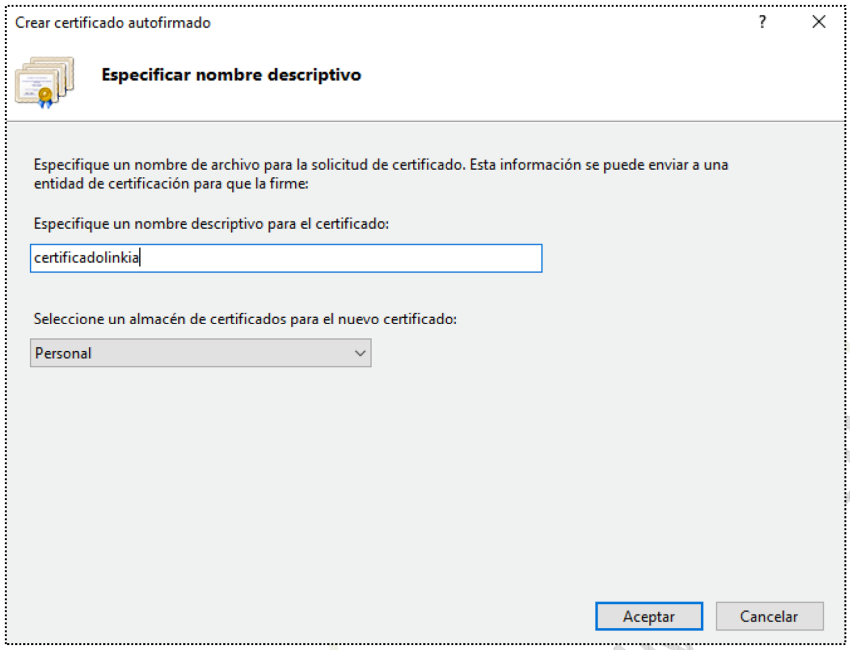


En la columna de la derecha (Acciones) se debe seleccionar la opción de crear un certificado auto-firmado, aunque en la misma lista se dispone de la opción para generar la petición de un certificado para enviar a una entidad emisora:

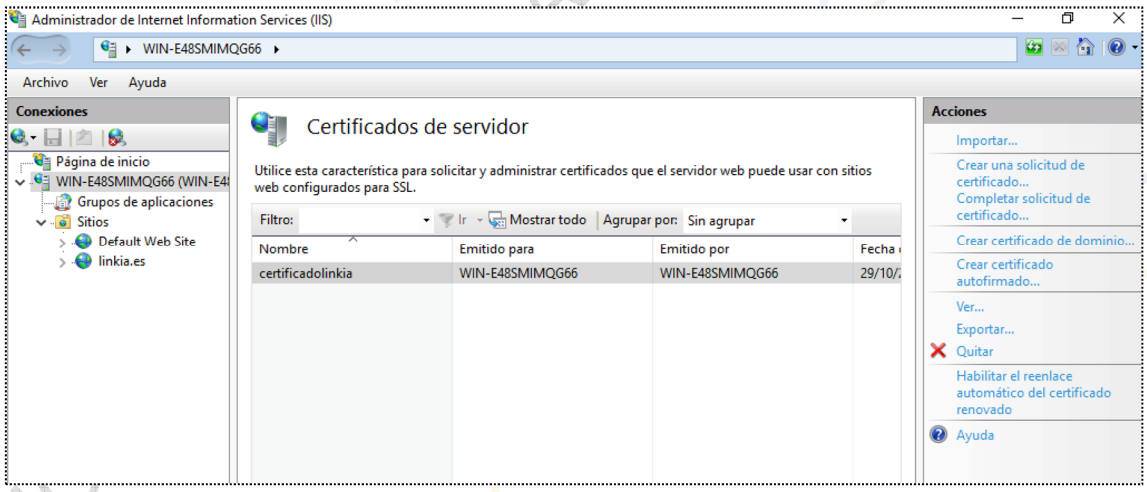


Se puede observar también que el servidor no tiene ningún certificado instalado por defecto.

Al crear el certificado pedirá un nombre descriptivo del archivo resultante:



El certificado se creará al instante y aparecerá en la lista de certificados del servidor:



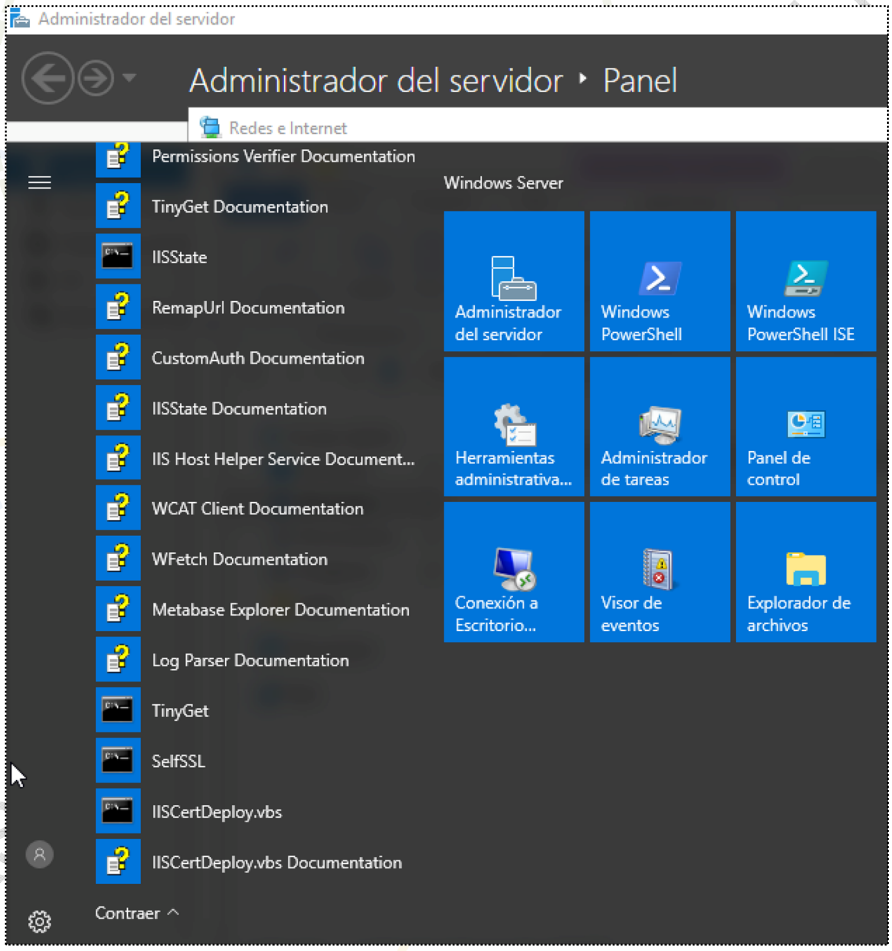
Es de vital importancia observar el campo "Emitido para" del certificado creado. Se observa cómo el certificado ha sido creado para el nombre de máquina Windows del servidor, y no para el nombre DNS del host que está previsto para el site (www.linkia.es).

Si se desea asociar este certificado al site creado anteriormente, cuando un cliente acceda a través de la URL www.linkia.es, el servidor presentará este certificado con el nombre de máquina Windows (que no es www.linkia.es), y esta discrepancia generará una alerta de seguridad en el cliente que causará desconfianza por parte del cliente.

En el caso de que el certificado se solicitara a una entidad emisora se debería hacer constar correctamente el futuro nombre DNS del sitio web, para que coincida sin problemas. En este caso del certificado auto-firmado se deberá cambiar manualmente el nombre (el Common Name) del certificado para que coincida.

Para poder hacer el cambio del Common Name del certificado se puede utilizar la herramienta SelfSSL.exe, que forma parte del *Internet Information Services (IIS) 6.0 Resource Kit Tools*. Dicho paquete de herramientas se puede bajar de: http://www.microsoft.com/en-us/download/confirmation.aspx?id=17275

Una vez instalado el paquete en el servidor se podrá hacer uso de la herramienta SelfSSL.exe a través de la línea de comandos. Se puede acceder a ella mediante el buscador de Windows.



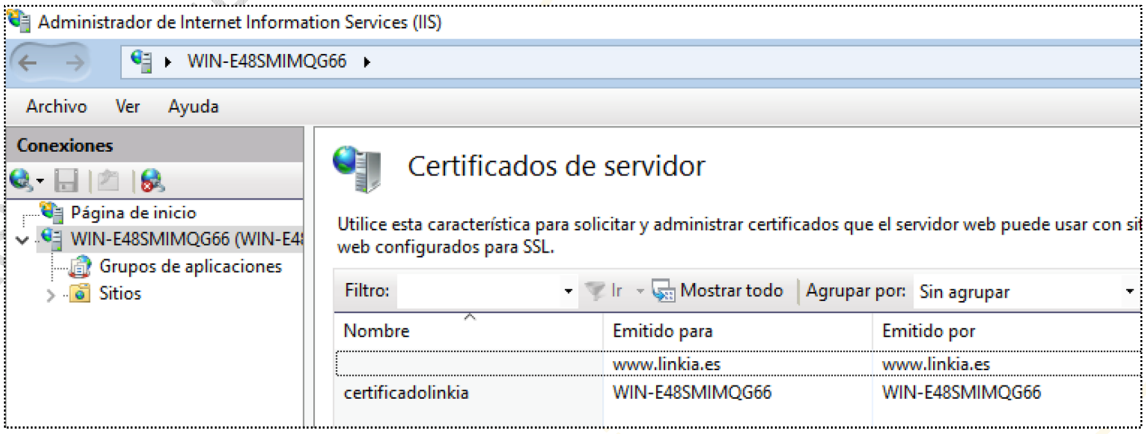
Si se selecciona la herramienta dirige a la línea de comandos, con la ayuda correspondiente:

```
Administrador: SelfSSL
Microsoft (R) SelfSSL Version 1.0
Copyright (C) 2003 Microsoft Corporation. All rights reserved.

Installs self-signed SSL certificate into IIS.
SELFSSL [/T] [/N:cn] [/K:key size] [/S:site id] [/P:port]

/T               Adds the self-signed certificate to "Trusted Certificates"
                 list. The local browser will trust the self-signed certificate
                 if this flag is specified.
/N:cn            Specifies the common name of the certificate. The computer
                 name is used if not specified.
/K:key size      Specifies the key length. Default is 1024.
/V:validity days Specifies the validity of the certificate. Default is 7 days.
/S:site id       Specifies the id of the site. Default is 1 (Default Site).
/P:port          Specifies the SSL port. Default is 443.
/Q               Quiet mode. You will not be prompted when SSL settings are
                 overwritten.

The default behaviour is equivalent with:

selfssl.exe /N:CN=WIN-E48SMIMQG66 /K:1024 /V:7 /S:1 /P:443

C:\Program Files (x86)\IIS Resources\SelfSSL>
```

A continuación se ejecuta el siguiente comando, que cambia el CN del certificado por el nombre de host del site y además establece una validez de 1000 días para el certificado (se genera un certificado nuevo):

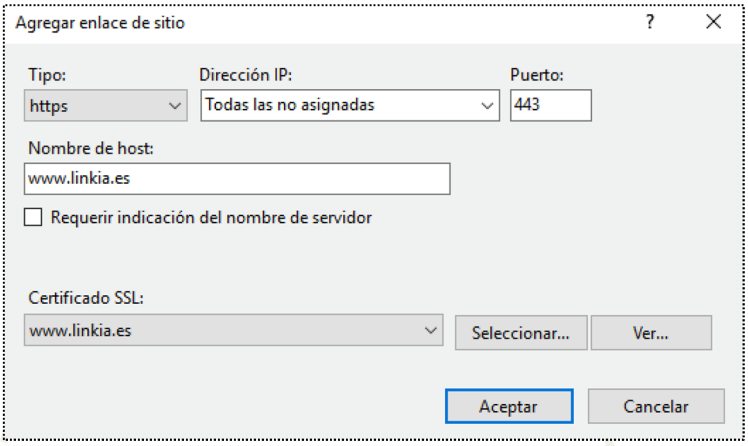
```
C:\Program Files (x86)\IIS Resources\SelfSSL>selfssl /N:CN=www.linkia.es /V:1000
Microsoft (R) SelfSSL Version 1.0
Copyright (C) 2003 Microsoft Corporation. All rights reserved.

Do you want to replace the SSL settings for site 1 (Y/N)?Y
Error opening metabase: 0x80040154
C:\Program Files (x86)\IIS Resources\SelfSSL>
```

Al final del proceso surge un error que se puede ignorar.
A continuación se observa el nuevo certificado en los certificados del servidor desde la herramienta de Administración de IIS:

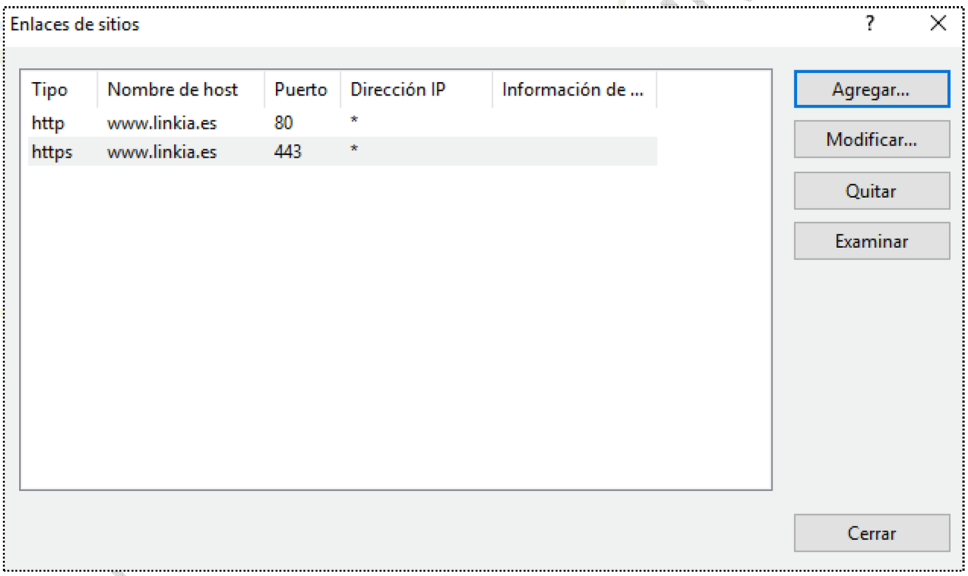


Se observan los dos certificados auto-firmados, el que tiene el nombre de máquina Windows, y el que tiene el nombre de host correcto en el campo "Emitido para". Se puede proceder a borrar el primero.

Para asociarlo al site se deben configurar los enlaces del site y añadir uno nuevo del tipo https, que normalmente se debe asociar al puerto 443. Al agregar el enlace pedirá seleccionar el certificado:



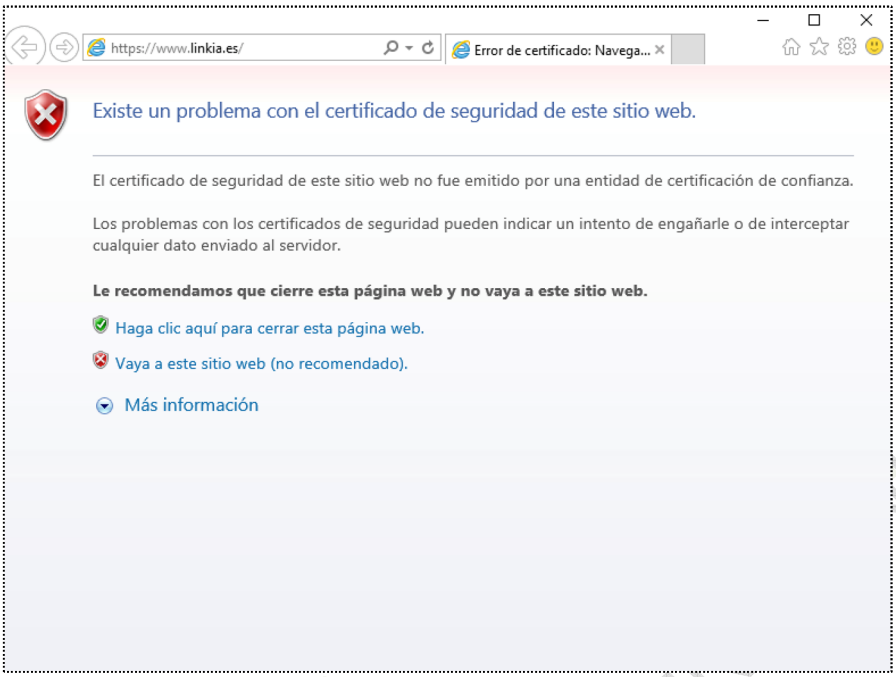
Después se puede decidir si se mantiene el enlace al puerto 80 mediante http, o se desea dejar en exclusiva el enlace seguro https por el puerto 443:



A continuación se realizarán las pruebas pertinentes desde un cliente externo con un navegador web. En este caso se trata de un Windows 7 con Internet Explorer como navegador.

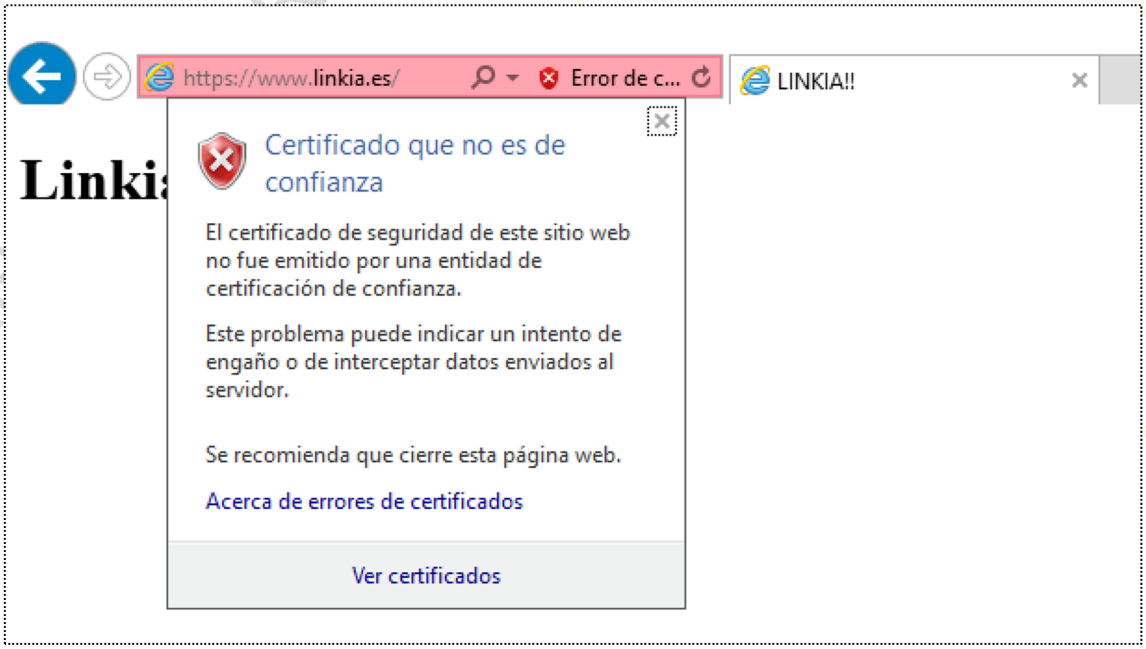
Si se accede directamente a la IP del servidor, se recibe una alerta doble

VERSIÓN LINKIAFP



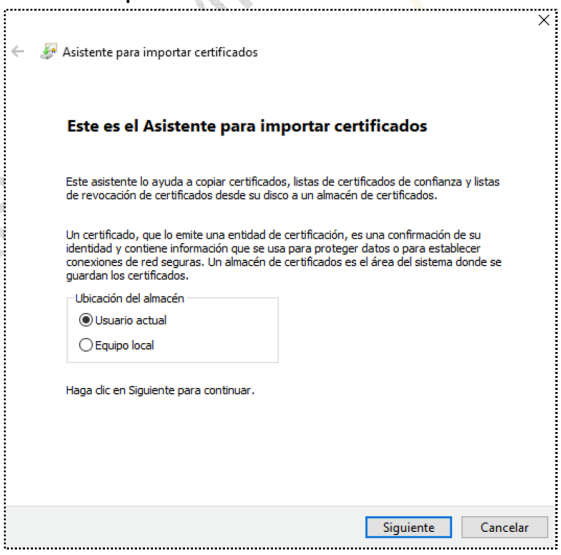
Por un lado informa que no se confía en la entidad que ha emitido el certificado. Este punto es normal ya que el certificado es auto-firmado del servidor, y de hecho nadie confía aún en ese servidor como entidad emisora, porque no lo es. En un entorno real, el servidor presenta un certificado en el que ya el sistema operativo confía, ya que lo emiten compañías o entidades que ya están preinstaladas en los sistemas operativos más comunes

A pesar de la alerta, el navegador deja continuar hasta el contenido web. Una vez allí, marcará el sitio como no seguro:

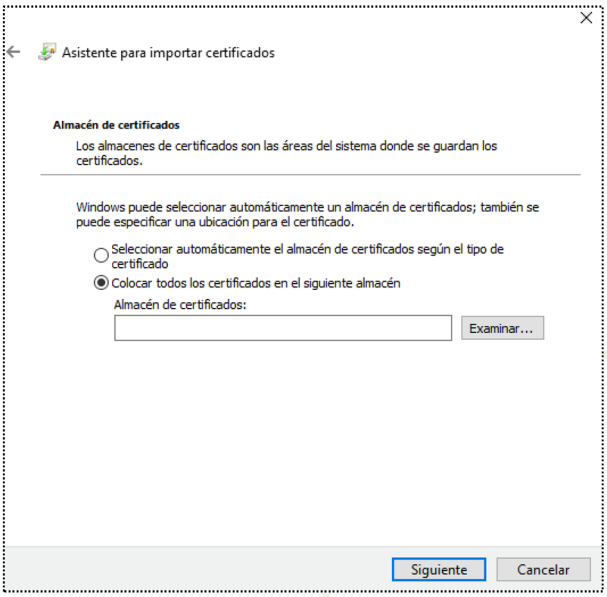


Para arreglar el problema de confianza en este entorno de test, se puede instalar manualmente el certificado en la lista de entidades emisoras de confianza del equipo cliente desde el que se está accediendo.
Si se selecciona la opción "Ver certificados" que sale del "Error de certificado", se podrá instalar el certificado en el cliente.

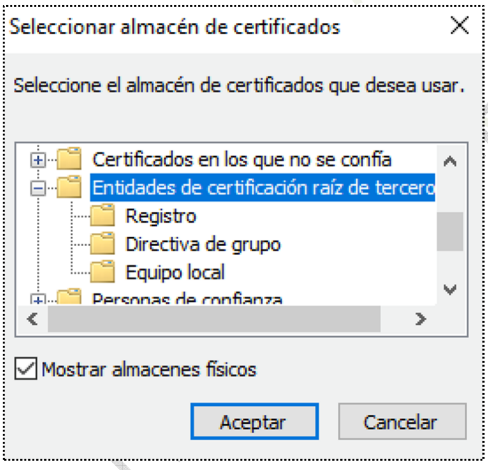
Al instalar el certificado se deberá seleccionar el almacén local donde se desea instalar. Un certificado se puede instalar asociado a la cuenta de usuario actual del cliente, o al sistema operativo cliente entero. A continuación se muestra como instalarlo:



VERSIÓN PARA ALUMNO LINKIAFP



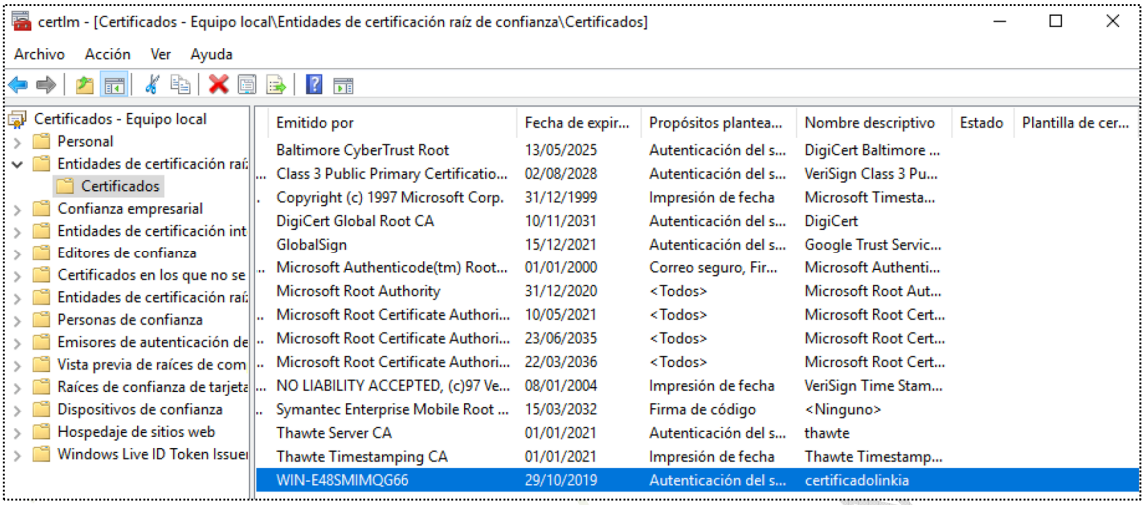
Después de elegir el almacén se selecciona "Examinar" y se marca la opción de Mostrar almacenes físicos":



Entre los almacenes físicos, se debe elegir el de "Entidades de certificación raíz de confianza" y finalmente Aceptar.

Al instalarse el certificado el sistema pide confirmación, ya que a partir de ahora se confiará en todos los certificados emitidos por esa entidad.
Se puede comprobar que el certificado ha quedado instalado si se inspecciona el almacén de certificados con la consola de certificados:

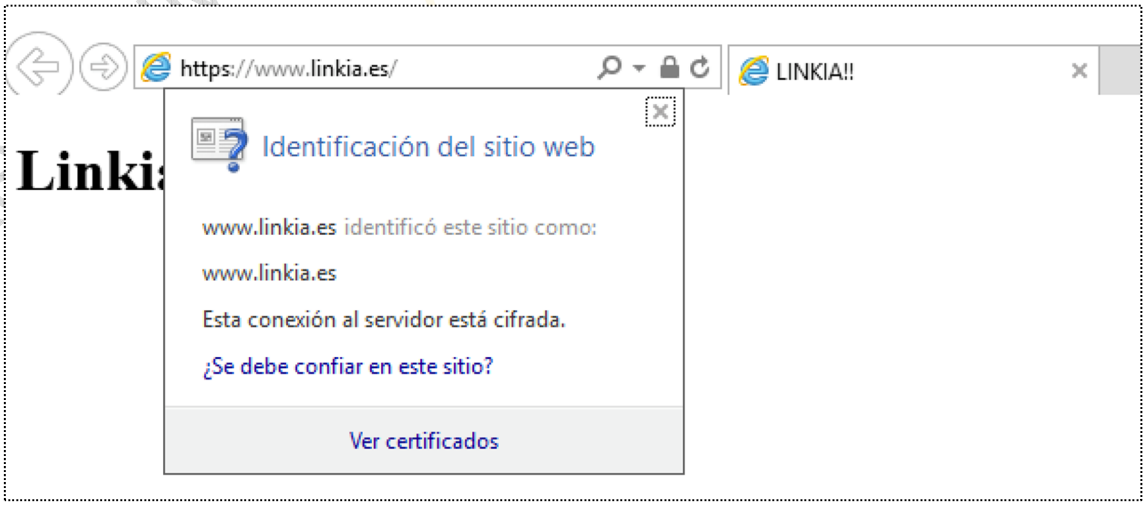
VERSIÓN DE ALUMNO LINKIAFP



El certificado se puede observar en la lista de entidades raíz de confianza, junto a todos los demás que están preinstalados y son de las entidades emisoras típicas a nivel público.

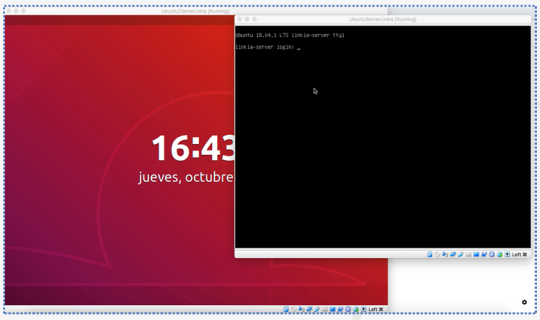
Si a continuación se vuelve a intentar acceder desde el cliente a través de la IP del servidor, en esta ocasión sigue saliendo la alerta pero asociada tan solo a un error. El error de la confianza ha quedado arreglado.

El último error por arreglar consiste simplemente en que el acceso se ha realizado a través de la IP del servidor, mientras que el certificado está emitido para el nombre de host www.site1.com. Al no coincidir, avisa de tal circunstancia.
Tan sólo falta hacer que el cliente llegue al servidor a través del nombre de host indicado, y no de la IP. Para ello se debe configurar la resolución DNS necesaria.
Este punto queda fuera del alcance de este apartado. Consulte los apartados de estos apuntes donde se hace referencia a servicios DNS para más ayuda.
A continuación se presenta el acceso ya correcto desde el cliente:

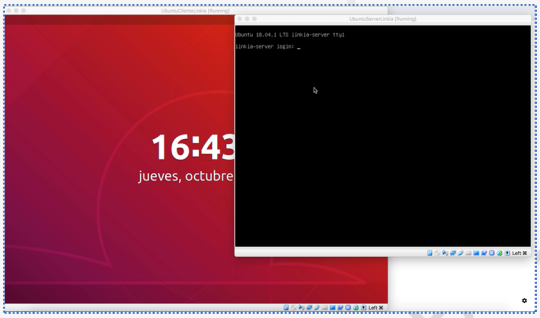


Se puede observar cómo una vez validado el certificado, el navegador informa que la conexión está cifrada.

Recursos/Enlaces



→ Video instalación Apache



→ Video instalación IIS

- Ubuntu server [Enlace]
- Windows 2016 Server [Enlace]

Conceptos clave

- **Apache:** Servidor web HTTP de código abierto.
- **IIS:** Internet Information Services o IIS es un servidor web y un conjunto de servicios para el sistema operativo Microsoft Windows.
- **HTML:** HTML es un lenguaje de programación que se utiliza para el desarrollo de páginas de Internet. Se trata de las siglas que corresponden a HyperText Markup Language, es decir, Lenguaje de Marcas de Hipertexto.

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP

Ponlo en práctica

Actividad 1

Instalación de Servidor Web Apache:

1. Crea una máquina virtual con Ubuntu Server.
2. Instala el servicio de Apache y configura un site.

SOLUCIÓN

Actividad 2

Instalación de Servidor Web IIS:

1. Crea una máquina virtual con Windows 2016 Server.
2. Instala el servicio de IIS y configura un site.

SOLUCIÓN

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP

Test de autoevaluación

1. Pon el significado de cada una de las siglas de IIS:
    a. I - Internet
    b. I - Information
    c. S – Services

2. ¿Cómo se la directiva qué permite acceder a un site solo a través de una única IP?
    a. allow
    b. virtualhost
    c. documentroot
    d. servername

3. ¿En qué ubicación se encuentra por defecto DocumentRoot del servidor IIS?
    a. c:\inetpub\www
    b. c:\var\htdocs\html
    c. c:\inetpub\wwwroot
    d. c:\inetpub\html

VERSIÓN IM

T2 Tema 2: Servicios de transferencia de ficheros

¿Qué aprenderás?

- Instalar y configurar un servidor de transferencia de ficheros en Linux.

¿Sabías que…?

- El Protocolo de transferencia de archivos (en inglés File Transfer Protocol o FTP) es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP (Transmission Control Protocol), basado en la arquitectura cliente-servidor.
- Filezilla Server es uno de los Servidores FTP más fáciles de usar, también nos permite crear certificados para que nuestro sitio FTP sea seguro.

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP

1. Introducción

Transferir ficheros entre sistemas remotos es una de las funciones más básicas y necesarias en las redes de ordenadores.

Los sistemas más utilizados hoy en día para la transferencia de ficheros están integrados para su uso desde navegadores web. Se trata de sistemas de almacenamiento y compartición de ficheros en internet, en la nube, que han sustituido a la manera tradicional de intercambiar archivos. Sistemas tales como Google Drive, Box, DropBox, etc... son protagonistas en el manejo de ficheros en la nube.

Sin embargo existe, y aún está en uso, un protocolo clásico pensado para la transferencia de ficheros grandes. Se trata de FTP (File Transfer Protocol). Es un protocolo de la capa de aplicación, que funciona sobre TCP en la capa de transporte.

Se basa en el modelo cliente-servidor. Para transferir ficheros entre dos hosts, uno debe hacer el papel de servidor a la escucha de peticiones, y el otro debe iniciar la conexión como cliente.

Existen numerosos paquetes de software que ejecutan las dos funciones, tanto servidores como clientes. A continuación se listan algunos de los más relevantes:

- Servidores FTP
  - Filezilla Server
  - Cerberus
  - IIS
  - Vsftpd

- Clientes FTP
  - Filezilla
  - Smart FTP
  - WinSCP

Muchos servicios en Internet aún son accesibles vía FTP. Un ejemplo de estos servicios es el de los servicios de hosting. Al contratarlos, el cliente suele tener acceso vía FTP a sus carpetas personales donde puede subir su contenido web para ser publicado.

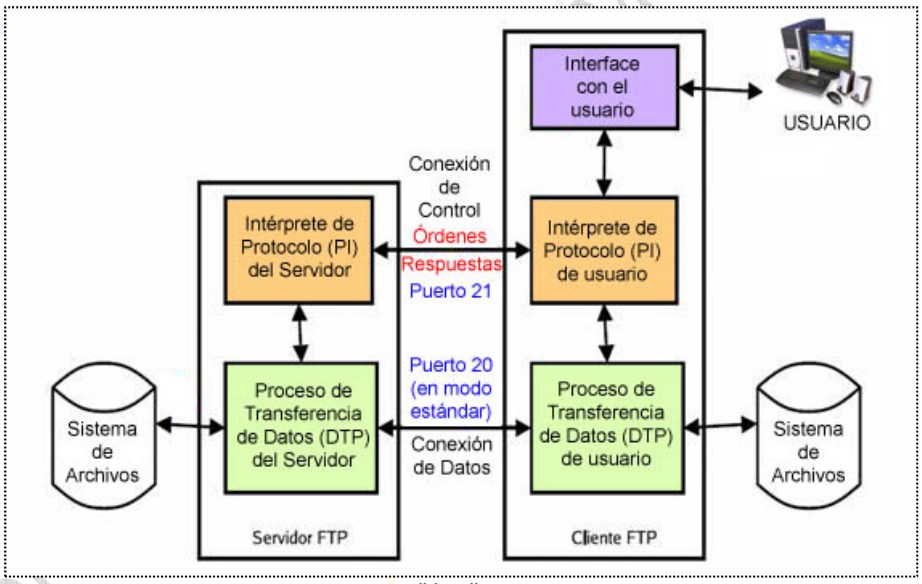
VERSIÓN IMPRIMIBLE ALUMNO LINKIAP

1.1. Protocolo FTP

FTP es un protocolo de la capa de Aplicación, que funciona sobre TCP e IP:

Aplicación	FTP
Transporte	TCP
Red	IP

El protocolo FTP utiliza los puertos 20 y 21 para intercambiar comandos y ficheros respectivamente. Los comandos e instrucciones viajan a través de conexiones TCP de puerto 21, mientras que los datos transferidos viajan por conexiones TCP de puerto 20:

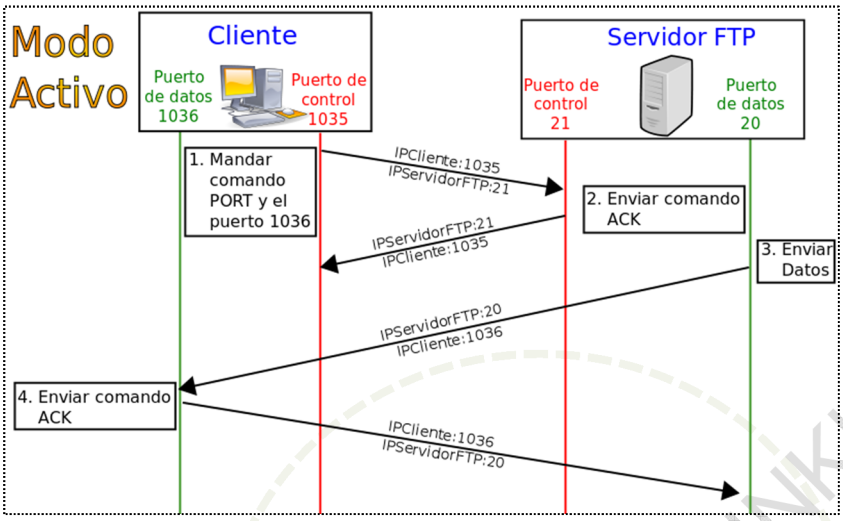


wikipedia.org

Las conexiones FTP pueden funcionar en dos modos diferentes, activo o pasivo.

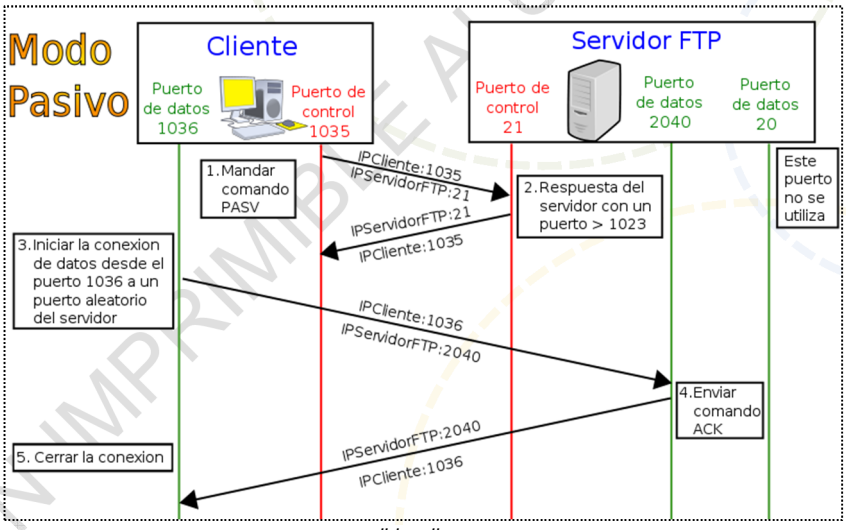
En modo Activo el cliente inicia la conexión contra el puerto 21 del servidor. Éste es el canal de control por el que se mandarán los comandos. Una vez establecida esa conexión, a través de ella, el cliente manda el comando PORT, informando al servidor sobre qué puerto ha abierto el cliente para recibir la conexión de datos (en el ejemplo el puerto 1036). Entonces el servidor inicia la conexión de datos desde su puerto 20 contra el puerto que le ha informado el cliente (1036 en el ejemplo).

A continuación se presenta un esquema de funcionamiento de la conexión en modo activo:



wikipedia.org

En modo pasivo, el cliente inicia también la conexión de control contra el puerto 21 del servidor. El cliente entonces envía el comando PASV. A continuación el servidor envía al cliente el puerto por el que debe conectar (en el ejemplo el 2040). El cliente entonces inicia la conexión de datos contra el puerto del servidor que acaba de informarle el servidor.



wikipedia.org

El puerto 20 del servidor no se utiliza en modo pasivo.

Antes de cada nueva transferencia tanto en el modo Activo como en el Pasivo, el cliente debe enviar otra vez un comando de control (PORT o PASV, según el modo en el que haya conectado), y el servidor recibirá esa conexión de datos en un nuevo puerto aleatorio (si está en modo pasivo) o por el puerto 20 (si está en modo activo).

Para transmitir archivos se debe elegir entre dos modos de transferencia que existen en FTP. Se trata de los modos ASCII y binario. El modo ASCII está pensado para la transmisión de ficheros legibles en caracteres ASCII. El modo binario está pensado para la transmisión de

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP

ficheros comprimidos, imágenes, ejecutables, etc. o cualquier otro que no esté codificado con códigos ASCII.

Al realizar cada envío, se deberá elegir el modo oportuno en el software FTP empleado.

Existen dos tipos de clientes FTP, clientes con interfaz gráfica, como Filezilla, y clientes de línea de comandos. Los de interfaz gráfica en realidad envían en segundo plano todos los comandos necesarios al servidor, mientras que con un cliente de línea de comandos se debe escribir todos los comandos manualmente, y por lo tanto conocerlos junto con su sintaxis correcta.

En la imagen siguiente se observa un establecimiento de conexión FTP mediante un cliente de línea de comandos:

```
root@linkia:/home/linkia# ftp 192.168.1.10
Connected to 192.168.1.10.
220 (vsFTPd 3.0.3)
Name (192.168.1.10:linkia): linkia
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> help
Commands may be abbreviated.  Commands are:

!               dir             mdelete         qc              site
$               disconnect      mdir            sendport        size
account         exit            mget            put             status
append          form            mkdir           pwd             struct
ascii           get             mls             quit            system
bell            glob            mode            quote           sunique
binary          hash            modtime         recv            tenex
bye             help            mput            reget           tick
case            idle            newer           rstatus         trace
cd              image           nmap            rhelp           type
cdup            ipany           nlist           rename          user
chmod           ipv4            ntrans          reset           umask
close           ipv6            open            restart         verbose
cr              lcd             prompt          rmdir           ?
delete          ls              passive         runique
debug           macdef          proxy           send
ftp> _
```

Después de la autenticación del usuario, se envía el comando "help", que devuelve un listado de los comandos FTP disponibles en ese servidor. Con todos esos comandos se pueden configurar y realizar todo tipo de transferencias de ficheros.

Por ejemplo, el comando *put* pone un fichero en el servidor, mientras que el comando *get*, lo trae desde el servidor al cliente.

VERSIÓN LINKIAFP

1.2. VSFTPD

VSFTPD (very secure FTP daemon) es un servidor FTP bien conocido para sistemas UNIX-like incluido Linux.

A continuación se presentarán los procedimientos para su instalación y configuración en Ubuntu Server 18.04.

1.2.1. Instalación y configuración básica

Para instalar el servicio FTP se utiliza el gestor de paquetes apt-get:

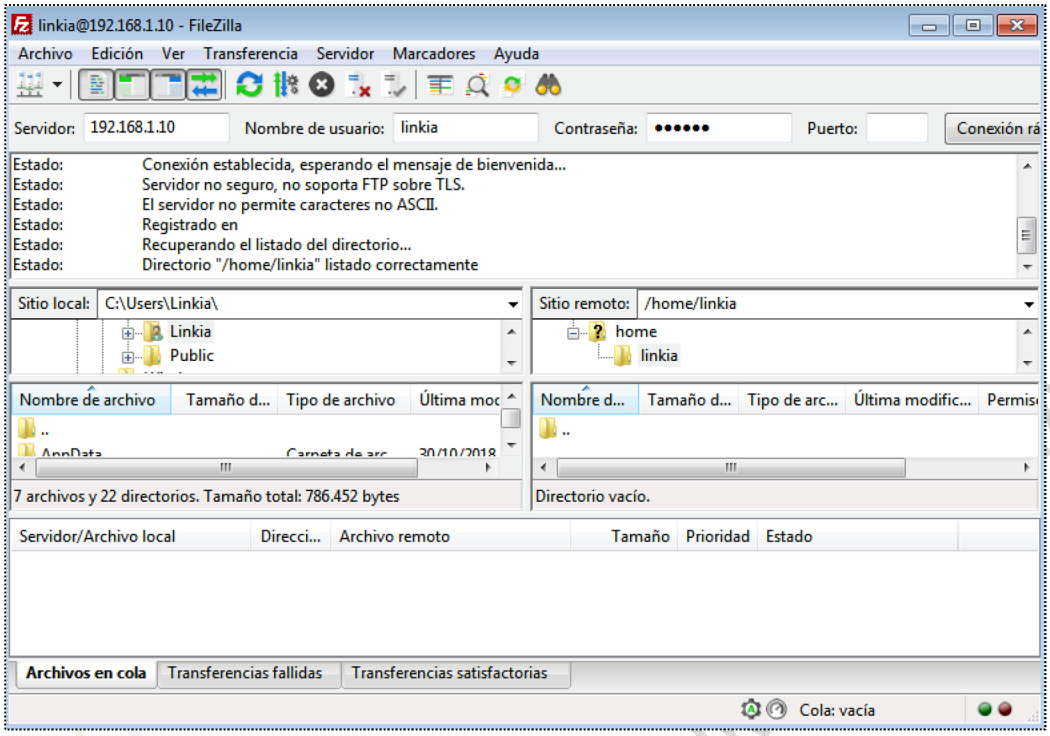
sudo apt install vsftpd

A continuación, se presenta una imagen del proceso:

```
root@linkia:/home/linkia# sudo apt install vsftpd
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes NUEVOS:
  vsftpd
0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 109 no actualizados.
Se necesita descargar 115 kB de archivos.
Se utilizarán 334 kB de espacio de disco adicional después de esta operación.
Des:1 http://archive.ubuntu.com/ubuntu bionic/main amd64 vsftpd amd64 3.0.3-9build1 [115 kB]
Descargados 115 kB en 0s (441 kB/s)
Preconfigurando paquetes ...
Seleccionando el paquete vsftpd previamente no seleccionado.
(Leyendo la base de datos ... 102899 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../vsftpd_3.0.3-9build1_amd64.deb ...
Desempaquetando vsftpd (3.0.3-9build1) ...
Procesando disparadores para ureadahead (0.100.0-20) ...
Configurando vsftpd (3.0.3-9build1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/vsftpd.service → /lib/systemd/system/vs
tpd.service.
Procesando disparadores para systemd (237-3ubuntu10) ...
Procesando disparadores para man-db (2.8.3-2) ...
Procesando disparadores para ureadahead (0.100.0-20) ...
root@linkia:/home/linkia#
```

El servicio queda en funcionamiento y totalmente operativo después de instalarlo.
Para comprobarlo se establece una sesión desde un cliente FTP con interfaz gráfica. En concreto se mostrará la conexión con Filezilla, que es un cliente de los más conocidos.

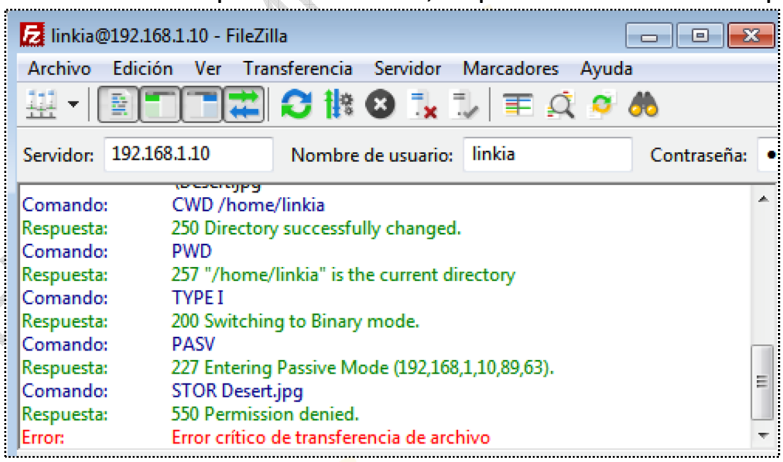
VERSIÓN NO LINKIAFP



En la parte superior se pueden ver los parámetros de la conexión: IP del servidor y usuario y contraseña. Los usuarios con acceso son los mismos que existen en el sistema operativo. En la ventana de la parte superior se observan los comandos enviados al servidor.

Nada más entrar, el cliente muestra en la parte izquierda un navegador de ficheros para el disco local del cliente, y en la parte derecha un navegador de archivos para el disco del servidor. Con el ratón se pueden hacer traspasos de ficheros entre uno y otro. En la parte superior se observan todos los comandos enviados que se generan con las acciones del ratón que se realizan en el cliente.

Si se intenta traspasar un fichero, se producirá un error de permisos:



Antes de empezar a traspasar ficheros se deberá examinar la configuración por defecto. Los parámetros más importantes para la configuración del servicio están presentes en el fichero:

/etc/vsftpd.conf

Modificando los parámetros de este fichero se gestiona la configuración del servicio. El fichero está bastante explicado con comentarios de ayuda. Los comentarios son líneas que comienzan con el carácter "#". Cada vez que se modifique algún valor de este fichero se debe reiniciar el servicio.

A continuación se muestra parte del contenido del fichero:

```
  GNU nano 2.9.3                          /etc/vsftpd.conf

# Example config file /etc/vsftpd.conf
#
# The default compiled in settings are fairly paranoid. This sample file
# loosens things up a bit, to make the ftp daemon more usable.
# Please see vsftpd.conf.5 for all compiled in defaults.
#
# READ THIS: This example file is NOT an exhaustive list of vsftpd options.
# Please read the vsftpd.conf.5 manual page to get a full idea of vsftpd's
# capabilities.
#
#
# Run standalone?  vsftpd can run either from an inetd or as a standalone
# daemon started from an initscript.
listen=NO
#
# This directive enables listening on IPv6 sockets. By default, listening
# on the IPv6 "any" address (::) will accept connections from both IPv6
# and IPv4 clients. It is not necessary to listen on *both* IPv4 and IPv6
# sockets. If you want that (perhaps because you want to listen on specific
# addresses) then you must run two copies of vsftpd with two configuration
# files.
listen_ipv6=YES
#
# Allow anonymous FTP? (Disabled by default).
anonymous_enable=NO
#
# Uncomment this to allow local users to log in.
local_enable=YES
#
# Uncomment this to enable any form of FTP write command.
#write_enable=YES
#
```

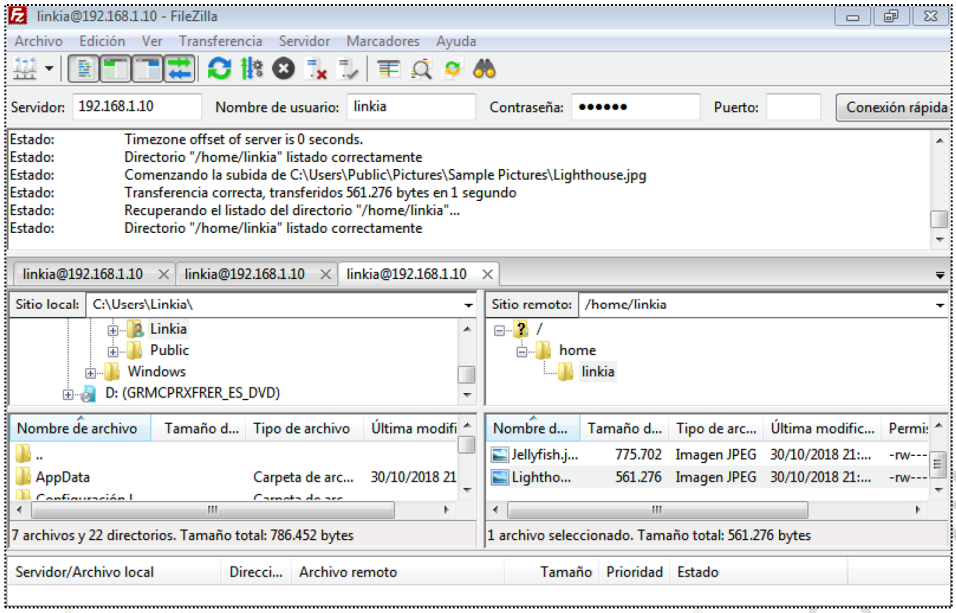
Si, por ejemplo, se descomenta (se quita el carácter # de la línea) la opción *write_enable*=YES, se habilitará la subida de archivos al servidor.

Después de descomentar la opción se debe reiniciar el servicio mediante el comando:

service vsftpd restart

Si se vuelve a intentar la transferencia hacia el servidor, en está ocasión se realizará correctamente:

VERSIÓN IMPRIMIBLE



1.2.2. Otras configuraciones

En el fichero /etc/vsftp.conf se encuentran todas las configuraciones posibles del servicio. Entre las más comunes se encuentran las siguientes:

- Habilitar Acceso anónimo:
  - anonymous_enable=YES
- Permitir que el usuario anónimo pueda subir archivos al servidor:
  - anon_upload_enable
- Hacer que los archivos subidos al servidor tengan un propietario concreto:
  - chown_uploads=YES
  - chown:username=usuario
- Mensaje de bienvenida
  - ftpd_banner=Bienvenido al servidor
- Que cada usuario quede "encerrado" en su directorio home:
  - chroot_local_user=YES

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP

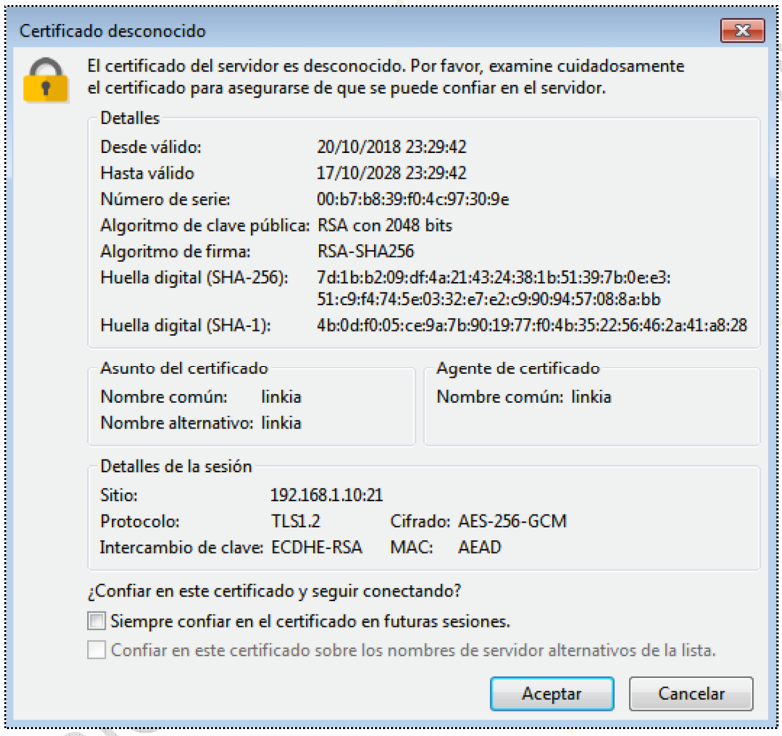
1.2.3. FTP seguro

Se debe tener en cuenta que el protocolo FTP es inseguro. Los comandos y los ficheros transmitidos viajan entre servidor y cliente sin encriptar. Existe la posibilidad de configurar FTP Seguro, que consiste en encapsular FTP con TLS/SSL, que es conocido como FTPS.

Para activar FTPS, se debe añadir al fichero de configuración:

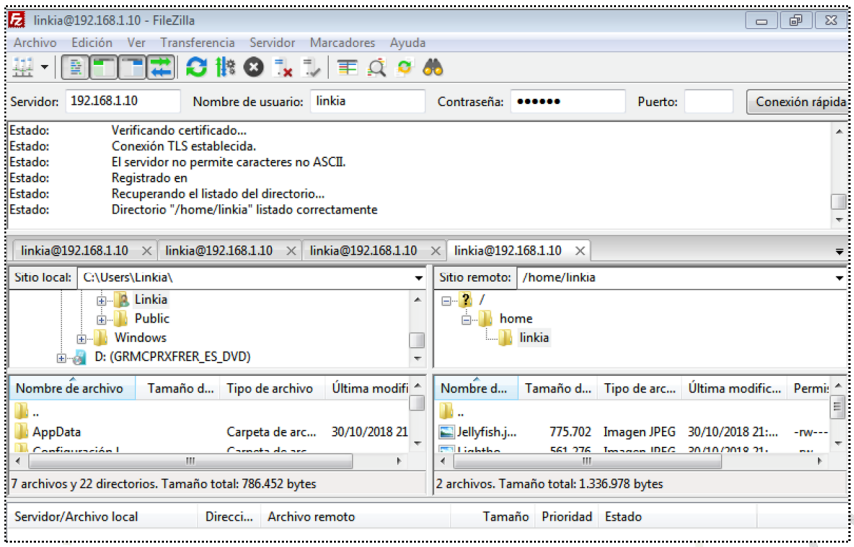
```
ssl_enable=Yes
```

Al volver a conectar desde el cliente, ahora surge una advertencia avisando que se establecerá una sesión segura:



VERSIÓN LINKIAFP

Si se acepta se establece la conexión:



El certificado que presenta por defecto el servidor está referenciado en el fichero de configuración mediante las siguientes líneas:

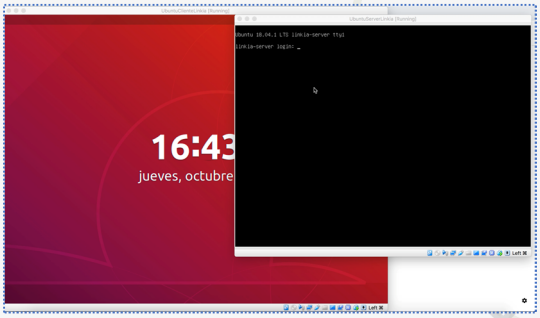
```
rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
```

En un entorno real este certificado y clave debería ser sustituido por uno de una entidad emisora de confianza.

VERSIÓN IMPRIMIBLE - CIUMNO LINKIAFP

Recursos/Enlaces

→ Video instalación VSFTPD



→ Video instalación IIS

- Ubuntu server [Enlace]

Conceptos clave

- **VSFTPD:** Vsftpd significa "Very Secure FTP Daemon" (Demonio FTP muy seguro), y es un servidor FTP para Linux y otros sistemas operativos UNIX.
- **Usuario "encerrado" / "enjaulado":** Permite que los usuarios no puedan navegar por las carpetas del servidor, y solo puedan acceder y utilizar la carpeta home asociada.

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP

Ponlo en práctica

Actividad 1

En una máquina virtual con Ubuntu Server instala un servidor VSFTPD que permita la conexión de un usuario y la subida de ficheros.

SOLUCIÓN

Actividad 2

Completa la siguiente frase con las palabras del listado:

En una conexión de tipo ____ el cliente envía la petición de conexión al puerto ____ del servidor. Éste le responde con un posible puerto para la ____ de datos, siendo mayor del ____. El ____ en este momento es capaz de iniciar la comunicación y el envío de datos a través del puerto indicado y lo envía hacia el puerto ___ del servidor.

cliente	1023	20	21	básico	transferencia

SOLUCIÓN

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP

Test de autoevaluación

1. ¿Qué puertos utiliza el protocolo FTP?
    a. 21 y 22
    b. 20 y 21
    c. 19 y 20
    d. 20
2. ¿El servicio de Ubuntu que permite instalar un servicio para la transferencia de datos es?
    a. VSFTPD
    b. VSFTP
    c. Filezilla Client
    d. Filezilla Server

VERSIÓN IMPRIMIBLE ALU

ANEXO I: Directivas de Apache http server

A continuación se presenta la documentación de ayuda oficial extraída de la página oficial de Apache para las principales Directivas mencionadas en estos apuntes. Para más detalle debe acudir a la página oficial de Apache.

<VirtualHost> Directiva

Descripción:	Contains directives that apply only to a specific hostname or IP address
Sintaxis:	`<VirtualHost addr[:port] [addr[:port]] ...> ... </VirtualHost>`
Contexto:	server config
Estado:	Core
Módulo:	core

`<VirtualHost>` and `</VirtualHost>` are used to enclose a group of directives that will apply only to a particular virtual host. Any directive that is allowed in a virtual host context may be used. When the server receives a request for a document on a particular virtual host, it uses the configuration directives enclosed in the `<VirtualHost>` section. *Addr* can be:

- The IP address of the virtual host;
- A fully qualified domain name for the IP address of the virtual host (not recommended);
- The character `*`, which is used only in combination with `NameVirtualHost *` to match all IP addresses; or
- The string `_default_`, which is used only with IP virtual hosting to catch unmatched IP addresses.

Example

```
<VirtualHost 10.1.2.3>
   ServerAdmin webmaster@host.example.com
   DocumentRoot /www/docs/host.example.com
   ServerName host.example.com
   ErrorLog logs/host.example.com-error_log
   TransferLog logs/host.example.com-access_log
</VirtualHost>
```

IPv6 addresses must be specified in square brackets because the optional port number could not be determined otherwise. An IPv6 example is shown below:

```
<VirtualHost [2001:db8::a00:20ff:fea7:ccea]>
   ServerAdmin webmaster@host.example.com
   DocumentRoot /www/docs/host.example.com
   ServerName host.example.com
   ErrorLog logs/host.example.com-error_log
   TransferLog logs/host.example.com-access_log
</VirtualHost>
```

Each Virtual Host must correspond to a different IP address, different port number or a different host name for the server, in the former case the server machine must be configured to accept IP packets for multiple addresses. (If the machine does not have multiple network interfaces, then this can be accomplished with the `ifconfig alias` command -- if your OS supports it).

Note

The use of `<VirtualHost>` does **not** affect what addresses Apache httpd listens on. You may need to ensure that Apache httpd is listening on the correct addresses using `Listen`.

When using IP-based virtual hosting, the special name `_default_` can be specified in which case this virtual host will match any IP address that is not explicitly listed in another virtual host. In the absence of any `_default_` virtual host the "main" server config, consisting of all those definitions outside any VirtualHost section, is used when no IP-match occurs.

You can specify a `:port` to change the port that is matched. If unspecified then it defaults to the same port as the most recent `Listen` statement of the main server. You may also specify `:*` to match all ports on that address. (This is recommended when used with `_default_`.)

A `ServerName` should be specified inside each `<VirtualHost>` block. If it is absent, the `ServerName` from the "main" server configuration will be inherited.

If no matching virtual host is found, then the first listed virtual host that matches the IP address will be used. As a consequence, the first listed virtual host is the default virtual host.

ServerAdmin Directiva

Descripción:	Email address that the server includes in error messages sent to the client
Sintaxis:	`ServerAdmin email-address|URL`
Contexto:	server config, virtual host
Estado:	Core
Módulo:	core

The `ServerAdmin` sets the contact address that the server includes in any error messages it returns to the client. If the `httpd` doesn't recognize the supplied argument as an URL, it assumes, that it's an *email-address* and prepends it with `mailto:` in hyperlink targets. However, it's recommended to actually use an email address, since there are a lot of CGI scripts that make that assumption. If you want to use an URL, it should point to another server under your control. Otherwise users may not be able to contact you in case of errors.

It may be worth setting up a dedicated address for this, e.g.

```
ServerAdmin www-admin@foo.example.com
```

as users do not always mention that they are talking about the server!

DocumentRoot Directiva

Descripción:	Directory that forms the main document tree visible from the web
Sintaxis:	`DocumentRoot directory-path`
Valor por defecto:	`DocumentRoot /usr/local/apache/htdocs`
Contexto:	server config, virtual host
Estado:	Core
Módulo:	core

This directive sets the directory from which `httpd` will serve files. Unless matched by a directive like `Alias`, the server appends the path from the requested URL to the document root to make the path to the document. Example:

```
DocumentRoot /usr/web
```

then an access to `http://www.my.host.com/index.html` refers to `/usr/web/index.html`. If the *directory-path* is not absolute then it is assumed to be relative to the `ServerRoot`.

The `DocumentRoot` should be specified without a trailing slash.

ErrorLog Directiva

Descripción:	Location where the server will log errors
Sintaxis:	`ErrorLog file-path|syslog[:facility]`
Valor por defecto:	`ErrorLog logs/error_log (Unix) ErrorLog logs/error.log (Windows and OS/2)`
Contexto:	server config, virtual host
Estado:	Core
Módulo:	core

The `ErrorLog` directive sets the name of the file to which the server will log any errors it encounters. If the *file-path* is not absolute then it is assumed to be relative to the `ServerRoot`.

Example

```
ErrorLog /var/log/httpd/error_log
```

If the *file-path* begins with a pipe character "|" then it is assumed to be a command to spawn to handle the error log.

Example

```
ErrorLog "|/usr/local/bin/httpd_errors"
```

See the notes on piped logs for more information.

Using `syslog` instead of a filename enables logging via syslogd(8) if the system supports it. The default is to use syslog facility `local7`, but you can override this by using the `syslog:facility` syntax where *facility* can be one of the names usually documented in syslog(1). The facility is effectively global, and if it is changed in individual virtual hosts, the final facility specified affects the entire server.

Example

```
ErrorLog syslog:user
```

SECURITY: See the security tips document for details on why your security could be compromised if the directory where log files are stored is writable by anyone other than the user that starts the server.

Note

When entering a file path on non-Unix platforms, care should be taken to make sure that only forward slashed are used even though the platform may allow the use of back slashes. In general it is a good idea to always use forward slashes throughout the configuration files.

CustomLog Directive

Description:	Sets filename and format of log file
Syntax:	`CustomLog file|pipe format|nickname [env=[!]environment-variable| expr=expression]`
Context:	server config, virtual host
Status:	Base
Module:	mod_log_config

The `CustomLog` directive is used to log requests to the server. A log format is specified, and the logging can optionally be made conditional on request characteristics using environment variables.

The first argument, which specifies the location to which the logs will be written, can take one of the following two types of values:

file

A filename, relative to the `ServerRoot`.

pipe

The pipe character "|", followed by the path to a program to receive the log information on its standard input. See the notes on piped logs for more information.

> **Security:**
>
> If a program is used, then it will be run as the user who started `httpd`. This will be root if the server was started by root; be sure that the program is secure.

> **Note**
>
> When entering a file path on non-Unix platforms, care should be taken to make sure that only forward slashed are used even though the platform may allow the use of back slashes. In general it is a good idea to always use forward slashes throughout the configuration files.

The second argument specifies what will be written to the log file. It can specify either a *nickname* defined by a previous `LogFormat` directive, or it can be an explicit *format* string as described in the log formats section.

For example, the following two sets of directives have exactly the same effect:

```
# CustomLog with format nickname
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog "logs/access_log" common

# CustomLog with explicit format string
CustomLog "logs/access_log" "%h %l %u %t \"%r\" %>s %b"
```

The third argument is optional and controls whether or not to log a particular request. The condition can be the presence or absence (in the case of a '`env=!name`' clause) of a particular variable in the server environment. Alternatively, the condition can be expressed as arbitrary boolean expression. If the condition is not satisfied, the request will not be logged. References to HTTP headers in the expression will not cause the header names to be added to the Vary header.

Environment variables can be set on a per-request basis using the `mod_setenvif` and/or `mod_rewrite` modules. For example, if you want to record requests for all GIF images on your server in a separate logfile but not in your main log, you can use:

```
SetEnvIf Request_URI \.gif$ gif-image
CustomLog "gif-requests.log" common env=gif-image
CustomLog "nongif-requests.log" common env=!gif-image
```

Or, to reproduce the behavior of the old RefererIgnore directive, you might use the following:

```
SetEnvIf Referer example\.com localreferer
CustomLog "referer.log" referer env=!localreferer
```

Listen Directive

Description:	IP addresses and ports that the server listens to
Syntax:	`Listen [IP-address:]portnumber [protocol]`
Context:	server config
Status:	MPM
Module:	`mpm_netware`, `mpm_winnt`, `mpmt_os2`, `prefork`, `worker`, `event`
Compatibility:	The *protocol* argument was added in 2.1.5

The `Listen` directive instructs Apache httpd to listen to only specific IP addresses or ports; by default it responds to requests on all IP interfaces. `Listen` is now a required directive. If it is not in the config file, the server will fail to start. This is a change from previous versions of Apache httpd.

The `Listen` directive tells the server to accept incoming requests on the specified port or address-and-port combination. If only a port number is specified, the server listens to the given port on all interfaces. If an IP address is given as well as a port, the server will listen on the given port and interface.

Multiple `Listen` directives may be used to specify a number of addresses and ports to listen to. The server will respond to requests from any of the listed addresses and ports.

For example, to make the server accept connections on both port 80 and port 8000, use:

```
Listen 80
Listen 8000
```

To make the server accept connections on two specified interfaces and port numbers, use

```
Listen 192.170.2.1:80
Listen 192.170.2.5:8000
```

IPv6 addresses must be surrounded in square brackets, as in the following example:

```
Listen [2001:db8::a00:20ff:fea7:ccea]:80
```

The optional *protocol* argument is not required for most configurations. If not specified, `https` is the default for port 443 and `http` the default for all other ports. The protocol is used to determine which module should handle a request, and to apply protocol specific optimizations with the `AcceptFilter` directive.

You only need to set the protocol if you are running on non-standard ports. For example, running an `https` site on port 8443:

```
Listen 192.170.2.1:8443 https
```

Error condition

Multiple `Listen` directives for the same ip address and port will result in an `Address already in use` error message.

ServerName Directiva

Descripción:	Hostname and port that the server uses to identify itself
Sintaxis:	`ServerName [scheme://]fully-qualified-domain-name[:port]`
Contexto:	server config, virtual host
Estado:	Core
Módulo:	core

The `ServerName` directive sets the request scheme, hostname and port that the server uses to identify itself. This is used when creating redirection URLs.

Additionally, `ServerName` is used (possibly in conjunction with `ServerAlias`) to uniquely identify a virtual host, when using name-based virtual hosts.

For example, if the name of the machine hosting the web server is `simple.example.com`, but the machine also has the DNS alias `www.example.com` and you wish the web server to be so identified, the following directive should be used:

```
ServerName www.example.com:80
```

The `ServerName` directive may appear anywhere within the definition of a server. However, each appearance overrides the previous appearance (within that server).

If no `ServerName` is specified, then the server attempts to deduce the hostname by performing a reverse lookup on the IP address. If no port is specified in the `ServerName`, then the server will use the port from the incoming request. For optimal reliability and predictability, you should specify an explicit hostname and port using the `ServerName` directive.

If you are using name-based virtual hosts, the `ServerName` inside a `<VirtualHost>` section specifies what hostname must appear in the request's `Host:` header to match this virtual host.

Sometimes, the server runs behind a device that processes SSL, such as a reverse proxy, load balancer or SSL offload appliance. When this is the case, specify the `https://` scheme and the port number to which the clients connect in the `ServerName` directive to make sure that the server generates the correct self-referential URLs.

See the description of the `UseCanonicalName` and `UseCanonicalPhysicalPort` directives for settings which determine whether self-referential URLs (e.g., by the `mod_dir` module) will refer to the specified port, or to the port number given in the client's request.

Failure to set `ServerName` to a name that your server can resolve to an IP address will result in a startup warning. `httpd` will then use whatever hostname it can determine, using the system's `hostname` command. This will almost never be the hostname you actually want.

```
httpd: Could not reliably determine the server's fully qualified
domain name, using rocinante.local for ServerName
```

NameVirtualHost Directiva

Descripción:	Designates an IP address for name-virtual hosting
Sintaxis:	`NameVirtualHost addr[:port]`
Contexto:	server config
Estado:	Core
Módulo:	core

A single `NameVirtualHost` directive identifies a set of identical virtual hosts on which the server will further select from on the basis of the *hostname* requested by the client. The `NameVirtualHost` directive is a required directive if you want to configure name-based virtual hosts.

This directive, and the corresponding `VirtualHost`, *must* be qualified with a port number if the server supports both HTTP and HTTPS connections.

Although *addr* can be a hostname, it is recommended that you always use an IP address or a wildcard. A wildcard NameVirtualHost matches only virtualhosts that also have a literal wildcard as their argument.

In cases where a firewall or other proxy receives the requests and forwards them on a different IP address to the server, you must specify the IP address of the physical interface on the machine which will be servicing the requests.

In the example below, requests received on interface 192.0.2.1 and port 80 will only select among the first two virtual hosts. Requests received on port 80 on any other interface will only select among the third and fourth virtual hosts. In the common case where the interface isn't important to the mapping, only the "*:80" NameVirtualHost and VirtualHost directives are necessary.

```
NameVirtualHost 192.0.2.1:80
NameVirtualHost *:80

<VirtualHost 192.0.2.1:80>
  ServerName namebased-a.example.com
</VirtualHost>

<VirtualHost 192.0.2.1:80>
  Servername namebased-b.example.com
</VirtualHost>

<VirtualHost *:80>
  ServerName namebased-c.example.com
</VirtualHost>

<VirtualHost *:80>
  ServerName namebased-d.example.com
</VirtualHost>
```

If no matching virtual host is found, then the first listed virtual host that matches the IP address and port will be used.

IPv6 addresses must be enclosed in square brackets, as shown in the following example:

```
NameVirtualHost [2001:db8::a00:20ff:fea7:ccea]:8080
```

Argument to `<VirtualHost>` directive

Note that the argument to the `<VirtualHost>` directive must exactly match the argument to the `NameVirtualHost` directive.

```
NameVirtualHost 192.0.2.2:80
<VirtualHost 192.0.2.2:80>
# ...
</VirtualHost>
```

Allow Directive

Description:	Controls which hosts can access an area of the server
Syntax:	`Allow from all|host|env=[!]env-variable [host|env=[!]env-variable] ...`
Context:	directory, .htaccess
Override:	Limit
Status:	Extension
Module:	mod_access_compat

The `Allow` directive affects which hosts can access an area of the server. Access can be controlled by hostname, IP address, IP address range, or by other characteristics of the client request captured in environment variables.

The first argument to this directive is always `from`. The subsequent arguments can take three different forms. If `Allow from all` is specified, then all hosts are allowed access, subject to the configuration of the `Deny` and `Order` directives as discussed below. To allow only particular hosts or groups of hosts to access the server, the *host* can be specified in any of the following formats:

A (partial) domain-name

```
Allow from example.org
Allow from .net example.edu
```

Hosts whose names match, or end in, this string are allowed access. Only complete components are matched, so the above example will match `foo.example.org` but it will not match `fooexample.org`. This configuration will cause Apache httpd to perform a double DNS lookup on the client IP address, regardless of the setting of the `HostnameLookups` directive. It will do a reverse DNS lookup on the IP address to find the associated hostname, and then do a forward lookup on the hostname to assure that it matches the original IP address. Only if the forward and reverse DNS are consistent and the hostname matches will access be allowed.

A full IP address

```
Allow from 10.1.2.3
Allow from 192.168.1.104 192.168.1.205
```

An IP address of a host allowed access

A partial IP address

```
Allow from 10.1
Allow from 10 172.20 192.168.2
```

The first 1 to 3 bytes of an IP address, for subnet restriction.

A network/netmask pair

```
Allow from 10.1.0.0/255.255.0.0
```

A network a.b.c.d, and a netmask w.x.y.z. For more fine-grained subnet restriction.

A network/nnn CIDR specification

```
Allow from 10.1.0.0/16
```

Similar to the previous case, except the netmask consists of nnn high-order 1 bits.

Note that the last three examples above match exactly the same set of hosts.

IPv6 addresses and IPv6 subnets can be specified as shown below:

```
Allow from 2001:db8::a00:20ff:fea7:ccea
Allow from 2001:db8::a00:20ff:fea7:ccea/10
```

The third format of the arguments to the `Allow` directive allows access to the server to be controlled based on the existence of an environment variable. When `Allow from env=env-variable` is specified, then the request is allowed access if the environment variable *env-variable* exists. When `Allow from env=!env-variable` is specified, then the request is allowed access if the environment variable *env-variable* doesn't exist. The server provides the ability to set environment variables in a flexible way based on characteristics of the client request using the directives provided by `mod_setenvif`. Therefore, this directive can be used to allow access based on such factors as the clients `User-Agent` (browser type), `Referer`, or other HTTP request header fields.

```
SetEnvIf User-Agent ^KnockKnock/2\.0 let_me_in
<Directory "/docroot">
    Order Deny,Allow
    Deny from all
    Allow from env=let_me_in
</Directory>
```

In this case, browsers with a user-agent string beginning with `KnockKnock/2.0` will be allowed access, and all others will be denied.

Merging of configuration sections

When any directive provided by this module is used in a new configuration section, no directives provided by this module are inherited from previous configuration sections.

Deny Directive

Description:	Controls which hosts are denied access to the server
Syntax:	`Deny from all|host|env=[!]env-variable [host|env=[!]env-variable] ...`
Context:	directory, .htaccess
Override:	Limit
Status:	Extension
Module:	mod_access_compat

This directive allows access to the server to be restricted based on hostname, IP address, or environment variables. The arguments for the `Deny` directive are identical to the arguments for the `Allow` directive.

VERSIÓN IMAFP

Order Directive

Description:	Controls the default access state and the order in which `Allow` and `Deny` are evaluated.
Syntax:	`Order ordering`
Default:	`Order Deny,Allow`
Context:	directory, .htaccess
Override:	Limit
Status:	Extension
Module:	mod_access_compat

The `Order` directive, along with the `Allow` and `Deny` directives, controls a three-pass access control system. The first pass processes either all `Allow` or all `Deny` directives, as specified by the `Order` directive. The second pass parses the rest of the directives (`Deny` or `Allow`). The third pass applies to all requests which do not match either of the first two.

Note that all `Allow` and `Deny` directives are processed, unlike a typical firewall, where only the first match is used. The last match is effective (also unlike a typical firewall). Additionally, the order in which lines appear in the configuration files is not significant -- all `Allow` lines are processed as one group, all `Deny` lines are considered as another, and the default state is considered by itself.

Ordering is one of:

`Allow,Deny`
First, all `Allow` directives are evaluated; at least one must match, or the request is rejected. Next, all `Deny` directives are evaluated. If any matches, the request is rejected. Last, any requests which do not match an `Allow` or a `Deny` directive are denied by default.

`Deny,Allow`
First, all `Deny` directives are evaluated; if any match, the request is denied **unless** it also matches an `Allow` directive. Any requests which do not match any `Allow` or `Deny` directives are permitted.

`Mutual-failure`
This order has the same effect as `Order Allow,Deny` and is deprecated in its favor.

Keywords may only be separated by a comma; *no whitespace* is allowed between them.

Match	Allow,Deny result	Deny,Allow result
Match Allow only	Request allowed	Request allowed
Match Deny only	Request denied	Request denied
No match	Default to second directive: Denied	Default to second directive: Allowed
Match both Allow & Deny	Final match controls: Denied	Final match controls: Allowed

In the following example, all hosts in the example.org domain are allowed access; all other hosts are denied access.

```
Order Deny,Allow
Deny from all
Allow from example.org
```

In the next example, all hosts in the example.org domain are allowed access, except for the hosts which are in the foo.example.org subdomain, who are denied access. All hosts not in the example.org domain are denied access because the default state is to `Deny` access to the server.

```
Order Allow,Deny
Allow from example.org
Deny from foo.example.org
```

On the other hand, if the `Order` in the last example is changed to `Deny,Allow`, all hosts will be allowed access. This happens because, regardless of the actual ordering of the directives in the configuration file, the `Allow from example.org` will be evaluated last and will override the `Deny from foo.example.org`. All hosts not in the `example.org` domain will also be allowed access because the default state is `Allow`.

The presence of an `Order` directive can affect access to a part of the server even in the absence of accompanying `Allow` and `Deny` directives because of its effect on the default access state. For example,

```
<Directory "/www">
    Order Allow,Deny
</Directory>
```

will Deny all access to the `/www` directory because the default access state is set to `Deny`.

The `Order` directive controls the order of access directive processing only within each phase of the server's configuration processing. This implies, for example, that an `Allow` or `Deny` directive occurring in a `<Location>` section will always be evaluated after an `Allow` or `Deny` directive occurring in a `<Directory>` section or `.htaccess` file, regardless of the setting of the `Order` directive. For details on the merging of configuration sections, see the documentation on How Directory, Location and Files sections work.

Merging of configuration sections

When any directive provided by this module is used in a new configuration section, no directives provided by this module are inherited from previous configuration sections.

VERSIÓN IMPRIMIBLE AFP

Alias Directive

Description:	Maps URLs to filesystem locations
Syntax:	`Alias URL-path file-path|directory-path`
Context:	server config, virtual host
Status:	Base
Module:	mod_alias

The `Alias` directive allows documents to be stored in the local filesystem other than under the `DocumentRoot`. URLs with a (%-decoded) path beginning with *URL-path* will be mapped to local files beginning with *directory-path*. The *URL-path* is case-sensitive, even on case-insensitive file systems.

```
Alias "/image" "/ftp/pub/image"
```

A request for `http://example.com/image/foo.gif` would cause the server to return the file `/ftp/pub/image/foo.gif`. Only complete path segments are matched, so the above alias would not match a request for `http://example.com/imagefoo.gif`. For more complex matching using regular expressions, see the `AliasMatch` directive.

Note that if you include a trailing / on the *URL-path* then the server will require a trailing / in order to expand the alias. That is, if you use

```
Alias "/icons/" "/usr/local/apache/icons/"
```

then the URL `/icons` will not be aliased, as it lacks that trailing /. Likewise, if you omit the slash on the *URL-path* then you must also omit it from the *file-path*.

Note that you may need to specify additional `<Directory>` sections which cover the *destination* of aliases. Aliasing occurs before `<Directory>` sections are checked, so only the destination of aliases are affected. (Note however `<Location>` sections are run through once before aliases are performed, so they will apply.)

In particular, if you are creating an `Alias` to a directory outside of your `DocumentRoot`, you may need to explicitly permit access to the target directory.

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

Any number slashes in the *URL-path* parameter matches any number of slashes in the requested URL-path.

Redirect Directive

Description:	Sends an external redirect asking the client to fetch a different URL
Syntax:	`Redirect [status] URL-path URL`
Context:	server config, virtual host, directory, .htaccess
Override:	FileInfo
Status:	Base
Module:	mod_alias

The Redirect directive maps an old URL into a new one by asking the client to refetch the resource at the new location.

The old *URL-path* is a case-sensitive (%-decoded) path beginning with a slash. A relative path is not allowed.

The new *URL* may be either an absolute URL beginning with a scheme and hostname, or a URL-path beginning with a slash. In this latter case the scheme and hostname of the current server will be added.

Then any request beginning with *URL-Path* will return a redirect request to the client at the location of the target *URL*. Additional path information beyond the matched *URL-Path* will be appended to the target URL.

```
# Redirect to a URL on a different host
Redirect "/service" "http://foo2.example.com/service"

# Redirect to a URL on the same host
Redirect "/one" "/two"
```

If the client requests `http://example.com/service/foo.txt`, it will be told to access `http://foo2.example.com/service/foo.txt` instead. This includes requests with `GET` parameters, such as `http://example.com/service/foo.pl?q=23&a=42`, it will be redirected to `http://foo2.example.com/service/foo.pl?q=23&a=42`. Note that `POST`s will be discarded.
Only complete path segments are matched, so the above example would not match a request for `http://example.com/servicefoo.txt`. For more complex matching using regular expressions, see the `RedirectMatch` directive.

> **Note**
> Redirect directives take precedence over Alias and ScriptAlias directives, irrespective of their ordering in the configuration file.

If no *status* argument is given, the redirect will be "temporary" (HTTP status 302). This indicates to the client that the resource has moved temporarily. The *status* argument can be used to return other HTTP status codes:

permanent
: Returns a permanent redirect status (301) indicating that the resource has moved permanently.

temp
: Returns a temporary redirect status (302). This is the default.

seeother
: Returns a "See Other" status (303) indicating that the resource has been replaced.

gone
: Returns a "Gone" status (410) indicating that the resource has been permanently removed. When this status is used the *URL* argument should be omitted.

Other status codes can be returned by giving the numeric status code as the value of *status*. If the status is between 300 and 399, the *URL* argument must be present. If the status is *not* between 300 and 399, the *URL* argument must be omitted. The status must be a valid HTTP status code, known to the Apache HTTP Server (see the function `send_error_response` in http_protocol.c).

```
Redirect permanent "/one" "http://example.com/two"
Redirect 303 "/three" "http://example.com/other"
```

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP

Linkia FP

Formación Profesional Oficial a Distancia

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP