Tema 4. Servidores Web. HTTP.

1. Características. 3
2. Componentes. 4
3. Nombres y direcciones. 4
4. Clientes web (navegadores o exploradores). 4
5. Servidores web. 5
6. Proxies web. 6
7. Protocolo HTTP. 6
7.1. Mensajes HTTP. 7
7.2. Métodos de petición. 7
7.3. Cabeceras. 8
7.4. Códigos de estado y error. 9
7.5. Autentificación y Seguridad. 9
7.6. Conexiones persistentes. 10
8. HTTPS. Protocolo seguro. 10
9. Alojamiento virtual. 12
10. Instalación y configuración IIS en Windows Server. 12
10.1. Instalación de IIS en Windows Server 2012. 12
10.2. Servidor virtual por defecto. 13
10.3. Configuración de servidores virtuales. 13
10.3.1. Servidores virtuales por nombre. 14
10.4. Directorios virtuales. 14
10.5. Servidores virtuales seguros. 15
10.5.1. Certificados autofirmados mediante IIS. 15
10.5.2. Certificados autofirmados mediante OpenSSL. 15
11. Instalación y configuración de Apache en Linux. 17
11.1. Instalación de Apache. 17
11.2. Servidor virtual por defecto. 19
11.3. Directivas. 19
11.3.1. DirectoryIndex. 20
11.3.2. Directory y Options Indexes. 20
11.3.3. Logs. 21
11.3.4. Codigos de error. 21

11.4. Directorios virtuales. 21
11.5. Configuración modular. 22
11.5.1. Módulo *userdir*. 22
11.6. Control de acceso por direcciones IP y nombres de dominio. 23
11.7. Autentificación y autorización. 25
11.8. Los archivos .htaccess. 26
11.9. Configuración de servidores virtuales. 27
11.9.1. Servidores virtuales por nombre. 28
11.9.2. Servidores virtuales por dirección IP. 29
11.9.3. Servidores virtuales por puerto. 30
11.9.4. Combinando diferentes tipos de servidores virtuales 30
11.10. Configuración de un servidor virtual seguro (https). 32
11.10.1. Servidor virtual HTTPS por defecto. 32
11.10.2. Servidores virtuales HTTPS. 33
Anexo 1. Certificados autofirmados mediante OpenSSL 34
Anexo 2. Control de acceso por IP y nombres de dominio en Apache 2.2 34
Anexo 3. HTTP/2 35

Tema 4. Servidores Web (HTTP).

HTTP (*Hyper Text Transfer Protocol*) es un protocolo de la capa de aplicación que permite el acceso a la información de hipertexto (HyperText Markup Language) ubicada en servidores usando el modelo cliente/servidor en una red TCP/IP.

1. Características.

La llamada Web o WWW (*World Wide Web*) es un servicio que permite el acceso y distribución de información (texto, imágenes, audio, video, etc.), las llamadas páginas web, que se encuentran en servidores disponibles en Internet. Todos estos recursos, identificados por medio de un URL, se conectan entre sí mediante hiperenlaces. Los clientes web (navegadores o exploradores) realizan la conexión a los sitios web y presentan los recursos solicitados.

Estos contenidos básicamente pueden ser *estáticos* o *dinámicos*.

En las llamadas páginas *estáticas* los contenidos simplemente se muestran y la única interacción que se permite por parte de los usuarios es el acceso a otras páginas mediante los hiperenlaces o el relleno de formularios cuyos datos se envían posteriormente a un servidor.

Las páginas *dinámicas* son programadas permitiendo a los usuarios interaccionar con ellas modificando los contenidos a mostrar, así como las respuestas que el servidor envía a los clientes. Algunas de las tecnologías usadas por las páginas dinámicas son:

- PHP (*PHP Hypertext Pre-processor*). Es un lenguaje de programación de *script* que se ejecuta en el servidor web. Cuando el cliente hace una petición al servidor para que le envíe una página web, el servidor ejecuta el intérprete de PHP. Éste procesa el *script* solicitado que generará el contenido HTML de manera dinámica. El resultado es enviado por el intérprete de PHP al servidor web, quien a su vez se lo envía al cliente.
- ASP (*Active Server Page*). Es una tecnología de Microsoft del tipo en el lado del servidor para páginas web generadas dinámicamente, que ha sido comercializada como un anexo a Internet Information Services (IIS) usando como lenguajes Visual Basic o C#. Al igual que PHP el código se ejecuta en el servidor, el resultado HTML se envía al servidor web, el cual lo envía al cliente.
- Servlets y applets de Java. Un servlet es un programa Java que se ejecuta en el servidor y que genera páginas web como respuesta a las peticiones del cliente. Los applets son programas Java que se descargan del servidor y que se ejecutan en el cliente como parte de la página solicitada.
- JSP (*Java Server Pages*). Básicamente el funcionamiento consiste en un servidor de aplicaciones que interpreta el código contenido en la página JSP para construir el código Java del servlet a generar. Este servlet será el que genere el documento (típicamente HTML) que finalmente se presentará en la pantalla del navegador del usuario.
- JavaScript. Es un lenguaje de programación interpretado que se ejecuta normalmente en el cliente (navegador) generando páginas dinámicas y una mejor presentación.
- CGI (*Common Gateway Interfaz*). Es una interfaz de programación que permite la comunicación entre el servidor web y una aplicación externa (escrita en Perl, Python, C, C++, Java, etc.), llamada aplicación CGI. Con este sistema, el servidor web pasa las solicitudes del cliente al programa externo cuya salida es enviada como tipo MIME al cliente, en lugar del archivo estático tradicional.
- AJAX (*Asynchronous JavaScrpit And XML*). Esta tecnología permite crear aplicaciones que se ejecutan en el cliente, mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. La comunicación asíncrona permite que los datos adicionales que se soliciten al servidor, se carguen en segundo plano sin interferir con la visualización ni el comportamiento de la página. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad y la velocidad en las aplicaciones.

2. Componentes.

El servicio web se basa en el modelo cliente-servidor y está formado por los siguientes componentes:

- Recursos. Documentos, vídeos, imágenes, audio, aplicaciones, etc., accesibles a través de servidores web y conectados mediante hiperenlaces.
- Nombres y direcciones (URIs y URLs). En la Web para identificar los recursos y acceder a ellos se usan cadenas de caracteres que los identifican unívocamente denominados URIs (Uniform Resource Identifier). Un URL identifica un recurso por su localización dentro del sistema.
- Clientes web (exploradores o navegadores). Permiten acceder a los recursos disponibles en los servidores web, estableciendo las conexiones e interpretando la información obtenida mostrándola a los usuarios. También se denominan *agentes de usuario*.
- Servidores web. Atienden las peticiones de los clientes y les envían los recursos solicitados.
- *Proxies* web. Programas que hacen de intermediarios entre los clientes y servidores web actuando como cortafuegos y/o almacenando en caché los datos para aumentar el rendimiento.
- Protocolo HTTP. Normas y reglas que permiten el diálogo entre clientes y servidores web usando TCP como protocolo de transporte.
- Tecnologías web. Utilidades para desarrollar aplicaciones basadas en la Web (HTML, XHTML, XML, CSS, XSL, DOM, XPath, XQuery, JSON, etc.)

3. Nombres y direcciones.

Como hemos comentado, para localizar en la Web los recursos y acceder a ellos se usan cadenas de caracteres que los identifican denominadas URI (*Uniform Resource Identifier)*. Una URL (*Uniform Resource Locator*) es un tipo de URI que identifica un recurso por su localización.

El formato de una URL se compone de las siguientes partes:

- Esquema o servicio: indica el protocolo que se usa para acceder al servicio. Puede ser: *http, https, ftp, mailto, ldap, file, gopher, telnet, news* o *data*.
- // separador.
- Servidor: dirección IP o nombre de dominio del servidor que contiene el recurso.
- Puerto: número de puerto por el que escucha el servidor. Por defecto se usa el puerto 80/TCP.
- Ruta del recurso: ruta relativa al directorio raíz del servidor que contiene el recurso.
- Recurso: el recurso a acceder. Si se omite se buscará un archivo índice por defecto.
- Parámetros y valores enviados al servidor.

Ejemplos:

http://www.pardillos.com/descargas/apuntes/tema1.pdf

https://203.45.121.5:8080/buscador.php?id=comic&personaje=filemon

Cuando un cliente indica una URL en el navegador, una vez resuelta la dirección IP del recurso mediante DNS, se establece una conexión TCP con el puerto 80 del servidor (puerto por defecto), que permanece a la escucha de las solicitudes HTTP. Realizada la conexión, el servidor envía el recurso solicitado, o un mensaje de error en forma de página web si no existe o no está disponible, mostrándose en el navegador.

4. Clientes web (navegadores o exploradores).

Los clientes web son aplicaciones que ejecuta el usuario y que permite acceder a los recursos de un servidor web introduciendo la URL del recurso. Aunque inicialmente se diseñaron como clientes HTTP, permiten actuar como clientes de otros protocolos, como por ejemplo FTP.

Muchos de ellos son multiplataforma, y la mayoría de los sistemas operativos instalan por defecto clientes web, aunque siempre podemos instalar y usar el que más nos convenga.

Cada navegador permite establecer ciertos parámetros de configuración relativos al aspecto, comportamiento, registro de páginas visitadas, información almacenada temporalmente o seguridad.

Mediante la instalación de complementos (*plugins* o *add-ons*) se permite ampliar la funcionalidad de los navegadores y la interpretación de recursos que no están en formato HTML, como por ejemplo la posibilidad de abrir documentos PDF, visualizar animaciones en Flash, vídeos, etc. Si el recurso que recibe el navegador no lo puede interpretar, normalmente lo redirige a una aplicación externa que sí pueda hacerlo, o pregunta qué se quiere hacer con él.

Aunque hay cientos de navegadores, la mayoría de ellos usan alguno de los principales *motores de renderizado* como: ***Gecko*** (*Mozilla, Firefox, K-Meleon, Camino, Flock,*), ***Webkit*** (*Safari, Arora, …*), *Trident (**Microsoft Internet Explorer)**, **KHTML** (Konqueror), **Blink** (Opera, Google Chrome, Chromiun), **EdgeHTML** (Edge).*

También existen navegadores en modo texto como *Lynx, Links, Bobcat* o *W3m*.

5. Servidores web.

Los servidores web o servidores HTTP, son aplicaciones que atienden las peticiones HTTP (por defecto en el puerto 80/TCP), procesan e interpretan código que puede estar escrito en diferentes lenguajes y envían a los clientes los recursos solicitados. Puede enviar contenido estático (archivos en general) o dinámico (como resultado de ejecutar programas).

Los recursos pueden estar localizados en el mismo equipo donde se ejecuta el servidor web o en otros equipos de la red a los que el servidor puede acceder usando protocolos adicionales.

Los recursos se almacenan en los servidores en un directorio concreto denominado "*sitio web*" donde se establece una jerarquía de directorios para organizar las páginas y elementos. Si la URL no especifica ningún recurso, normalmente se accede a la llamada "página inicial" (normalmente *index.html*) situada en el directorio raíz de la jerarquía, usándose de índice para el resto de recursos y elementos del sitio.

El contenido a transmitir debería ser tipo MIME (*Multipurpose Internet Mail Extension*). Los tipos MIME son una forma estandarizada y extensible de clasificar el tipo de contenido de los datos. Como su nombre indica fueron en un primer momento utilizados para extender las características del correo electrónico, aunque su uso se ha generalizado. Actualmente define los formatos, tipos de letra y características de una página para que sea visible por distintos navegadores. Los MIME se componen de un tipo y un subtipo. Por ejemplo, un archivo que es un documento de texto y que ha sido escrito en HTML, tendrá como tipo MIME: text/html.

El registro de los tipos MIME los controla la *IANA* (*Internet Asigned Numbers Authority*), y en su sitio web podemos obtener la lista completa y actualizada de los tipos registrados. Es importante el registro de tipos MIME, esto asegura que dos tipos de contenido distintos no acaban con el mismo nombre.

Los tipos MIME se usan, por ejemplo para informar al navegador del tipo de datos que recibe, permitir la negociación de contenido o en los mensajes de correo, encapsular una o más entidades dentro del cuerpo de mensaje, mediante los tipos MIME multipart.

Es aconsejable indicar el tipo MIME cuando se crea una página web, para que los servidores conozcan el tipo del recurso a servir. Esta información se escribe en el <head> del documento HTML. Ejemplo:

```
<meta http-equiv="Content-Type" content="text/html" />
```

A la hora de gestionar los sitios web, la extensión al protocolo HTTP 1.1 WebDAV (*Web-based Distributed Authoring and Versioning*) permite la elaboración y administración de sitios web de forma colaborativa y remota. (Microsoft Office y otras aplicaciones incorporan soporte para WebDAV).

Los servidores web permiten múltiples configuraciones gracias a su arquitectura modular, lo que permite ampliar o reducir funcionalidades de una forma sencilla. Así, aparte del comportamiento básico del servidor, éste podrá realizar ciertas tareas adicionales dependiendo de los módulos que estén cargados.

Existen múltiples servidores web siendo los más utilizados: *Apache HTTP Server, IIS (Internet information Sever), Nginx, Servidor Web de Google, Cherokee, Lighttpd*.

En la página web https://trends.builtwith.com/web-server se pueden obtener informaciones sobre análisis y comparativas web, estadísticas de uso de los diferentes servidores web y datos sobre cuál es el servidor web que se ejecuta en un sitio web.

6. Proxies web.

En general la funcionalidad de un proxy es la de permitir a otros equipos conectarse a una red de forma indirecta a través de él. Así, cuando un equipo de la red desea acceder a una información o recurso, es realmente el proxy quien realiza la comunicación y a continuación traslada el resultado al equipo inicial. Las razones principales para usar un proxy como intermediario son porque a veces la comunicación directa no es posible, o bien porque el proxy añade una funcionalidad adicional.

En particular un *proxy web* aparte de la utilidad general de un proxy, proporciona una caché para las páginas web y los contenidos descargados, que es compartida por todos los equipos de la red; con la consiguiente mejora en los tiempos de acceso para consultas coincidentes. Al mismo tiempo libera la carga de los enlaces hacia Internet.

Los *proxies web* también pueden filtrar el contenido de las páginas web servidas o controlar el acceso de los usuarios actuando como cortafuegos. Otros tipos de proxy permiten cambiar el formato de las páginas web para mostrarlas en dispositivos distintos a un monitor, como un teléfono móvil o una PDA.

7. Protocolo HTTP.

El protocolo HTTP define las reglas que usan los servidores y clientes para comunicarse entre ellos. Es un protocolo que no usa una conexión de control y que define los posibles mensajes que los clientes pueden enviar, así como la estructura y formato de las respuestas. El protocolo define una estructura de datos llamadas cabeceras que se utilizan tanto en las peticiones como en las repuestas. Existen diferentes versiones del protocolo: HTTP/1.0, HTTP/1.1 (la más usada), HTTP/1.2 (versión experimental) y la más reciente HTTP/2 que de forma gradual va ganando terreno (Consultar el Anexo 3 al final del tema).

El funcionamiento básico de comunicación entre un cliente y un servidor es el siguiente:

- El usuario introduce la URL del recurso en la barra del navegador o bien sigue un enlace de un documento HTML.
- El cliente Web descodifica la URL, separando sus diferentes partes. Se identifica el protocolo de acceso, la dirección DNS o IP del servidor, el posible puerto opcional (el valor por defecto es 80) y el objeto requerido del servidor.
- Se abre una conexión TCP/IP con el servidor, llamando al puerto TCP correspondiente y se realiza la petición. Para ello, se envía el mensaje de petición necesario (GET, POST, HEAD,…), la dirección del objeto requerido (el contenido de la URL que sigue a la dirección del servidor), la versión del protocolo HTTP empleada (casi siempre HTTP/1.1) y un conjunto variable de información, que incluye datos sobre las capacidades del browser, datos opcionales para el servidor,…etc.
- El servidor devuelve la respuesta al cliente. Consiste en un código de estado y el tipo de dato MIME de la información enviada, seguido de la propia información solicitada.
- Se cierra la conexión TCP/IP.

Importante. Cada transacción HTTP es una comunicación distinta en donde en cada una de ellas se intercambian mensajes HTTP. Así cuando "descargamos una página web", en realidad se establecen decenas de comunicaciones que van descargando los distintos recursos de la página.

Por ejemplo, si requiere obtener recursos del servidor con la IP W.X.Y.Z, la forma de obtenerlos sería utilizando los mensajes de petición del protocolo HTTP. Para ello se usa un navegador, que es lo que normalmente se hace, o bien una conexión *telnet* con el servidor web accediendo a su puerto 80.

```
#telnet W.X.Y.Z 80
```

Realizada la conexión enviaríamos el mensaje de petición indicando el recurso y el protocolo a usar. En el ejemplo se solicita la página web por defecto del directorio raíz del sitio:

```
GET / HTTP/1.0
```

El mensaje de respuesta enviado por el servidor estará compuesto de una línea inicial con el código de estado, seguida de un encabezado (del que luego explicaremos su formato y significado) y termina con el cuerpo del mensaje, en este caso con el código HTML de la página web del directorio raíz del servidor.

7.1. Mensajes HTTP.

Los mensajes HTTP/1.X son en texto plano conteniendo las órdenes y parámetros según la sintaxis definida en el protocolo. Pueden ser de dos tipos: de *petición* y de *respuesta*.

- Los **mensajes de petición** están formados por tres partes:
  - Línea inicial de petición: indica el método de petición utilizado (GET, POST, …), la URL solicitada, y opcionalmente la versión del protocolo. Por ejemplo: GET / HTTP/1.1
  - Líneas de cabecera: conjunto de pares nombre/valor denominados cabeceras o encabezados que determinan cómo será procesada la petición por parte del servidor. Suelen enviar información referente a la solicitud y/o del cliente (identificación del navegador, sistema operativo, tipos MIME aceptados, etc.). Cada cabecera se muestra en una línea, dejando una línea en blanco detrás de la última cabecera. Por ejemplo la cabecera Accept: text/html indica que el navegador es capaz de procesar código HTML.
  - Cuerpo del mensaje (opcional): contiene datos o archivos a enviar al servidor. Debe estar separado de la cabecera mediante una línea en blanco.
- Los **mensajes de respuesta** están formados por tres partes:
  - La línea inicial de respuesta (línea de estado): informa de la versión HTTP usada por el servidor, continuando con un código de estado o de error que indica al cliente cómo ha sido procesada la petición, y termina con un texto explicativo del código de estado. Por ejemplo: HTTP/1.1 200 OK.
  - Líneas de cabecera: conjunto de pares nombre/valor denominados cabeceras que describen los datos y la forma en que son enviados al cliente. Cada cabecera se muestra en una línea, dejando una línea en blanco detrás de la última cabecera. Por ejemplo la cabecera: Content-Lenght: 57 indica que la respuesta del servidor tiene un tamaño de 57 bytes.
  - Cuerpo del mensaje (opcional): Quedará determinado por el tipo de recurso solicitado. Por ejemplo: <html><head></head><body> Hola Pepe </body></html>

7.2. Métodos de petición.

Los métodos de petición especifican la operación que quiere realizar el cliente en el servidor. La versión HTPP 1.1 contempla 7 métodos:

- Método **GET**. Es el más usado y se emplea para obtener cualquier tipo de información del servidor. Se invoca cuando colocamos una URL en el navegador, pinchamos en un enlace o cuando se envía un formulario mediante GET (<form method="get" …>), lo que permite enviar parámetros (datos) llamados *Query String* al servidor en la misma URL de petición.

  Por ejemplo si en la URL de un navegador usamos www.sitio.com/datos.php?nombre=pepe&edad=21, el mensaje de petición tendría el formato:

  ```
  GET /datos.php?nombre=pepe&edad=21 HTTP/1.1
  Host: www.sitio.com
  ```

  Las peticiones GET no envían cuerpo del mensaje, y no se pueden usar para subir archivos o elementos pesados al servidor.
- Método **POST**. Se emplea para indicar al servidor que acepte información adjunta en una petición. POST sí envía cuerpo del mensaje y en él se incluyen los parámetros y los datos que se necesiten enviar. (En este caso los parámetros no son visibles en la URL). Se invoca normalmente al enviar un formulario HTML POST. Se utilizan en operaciones que no deberían ser repetidas (los navegadores advierten cuando se refresca una página con una petición POST). No hay límites en la cantidad de datos que se envían al servidor, por lo tanto es el método para subir archivos al servidor.

  Si la petición del ejemplo anterior hubiera sido de tipo POST, entonces sería:

  ```
  POST /datos.php HTTP/1.1
  Host: www.sitio.com
  Content-Type: application/x-www-form-urlencoded
  Content-Length: 36

  nombre=pepe&edad=21
  ```

- Método **OPTIONS**. Permite solicitar al servidor información sobre qué métodos de petición tiene disponibles para una URL específica.
- Método **HEAD**. Es como una petición GET, pero recuperando en la respuesta sólo las cabeceras.
- Método **PUT**. Permite enviar recursos, generalmente archivos, al servidor.
- Método **DELETE**. Permite eliminar recursos del servidor.
- Método **TRACE**. Permite obtener la traza de una petición a través de *proxies* y cortafuegos.

7.3. Cabeceras.

Las cabeceras están formadas por pares de nombre/valor que se pueden incluir en los mensajes HTTP. Definen información acerca de los servidores, los clientes y de los datos que se intercambian entre ellos.

Existen muchos tipos de cabeceras y la podemos clasificar en:

- **Generales**. Definen información que puede ser de utilidad para los clientes y los servidores. Por ejemplo información de control de la caché (*Caché-Control*), fechas (*Date*), sistema de códigos usado en la transferencia, (*Transfer-Encoding*), etc.
- De **petición**. Usadas por los clientes para enviar información al servidor. Como por ejemplo el tipo de navegador (*User-Agent*), nombre del servidor al que se le realiza la petición (*Host*), tipos MIME soportados, uso de compresión de datos, idiomas que acepta el navegador, *cookies*, etc.
- De **respuesta**. La usa el servidor para enviar información añadida al cliente. Por ejemplo el TTL o edad de la respuesta (*Age*), la fecha en la que se espera que esté disponible un recurso que en cierto momento no lo está (*Retry-after*), el tipo de autentificación necesaria para acceder el recurso (*WWW-Authenticate*), el tipo y versión del servidor (*Server*), etc.
- De **entidad**. Es información que está relacionada directamente con el recurso a proporcionar al cliente, como por ejemplo el tipo de codificación (*Content-Encoding*), idioma (*Content-Language*), longitud (*Content-Length*), tipo MIME, etc.

La versión HTTP/1.0 define 16 cabeceras, todas ellas opcionales; y la HTTP/1.1 define 46, donde la única obligatoria es la cabecera de petición *host* si se configuran servidores virtuales por nombre.

Por otro lado, las aplicaciones web pueden definir sus propias cabeceras para añadir información adicional en los mensajes HTTP. Estas cabeceras personalizadas usan por convenio el prefijo X- para identificarlas. Por ejemplo X-temperatura.

Un ejemplo de cabeceras es el siguiente mensaje de respuesta proporcionado por un servidor web:

```
HTTP/1.0 400 Bad Request
Server: AkamaiGHost
Mime-Version: 1.0
Content-Type: text/html
Content-Length: 194
Expires: Sat, 21 Apr 2012 17:34:16 GMT
Date: Sat, 21 Apr 2012 17:34:16 GMT
Connection: close
<HTML><HEAD><TITLE>Invalid URL</TITLE></HEAD><BODY><H1>Invalid URL</H1>The requested URL
"&#47;", is invalid.<p>Reference&#32;&#35;9&#46;2cffef50&#46;1335029656&#46;1b70c15f </BODY>
</HTML>
```

Los navegadores disponen de complementos para visualizar y consultar las cabeceras de los sitios web visitados. También las veremos en la pestaña *Red* de las *Herramientas para desarrolladores* (F12).

Existen ciertas cabeceras que por su importancia las comentaremos más detenidamente.

- Almacenamiento en caché. De las informaciones que almacena un servidor, algunas de ellas no cambian durante mucho tiempo. HTTP permite el almacenamiento en caché durante un tiempo para disminuir el tráfico en la red. Las cabeceras Caché-Control, Last-Modified, Expires, Age, Etag, If-Match, if-Modified-Since permiten controlar qué datos se almacenan en caché, durante cuánto tiempo, qué no se puede almacenar, si hay que actualizar la caché, etc.
- Compresión. Los servidores pueden comprimir los recursos solicitados para reducir el tráfico en la red. Para ello los clientes usan en los mensajes de petición la cabecera Accept-Encoding para indicar que aceptan compresión; y los servidores la cabecera Content-Encoding para informar al cliente que va a enviar los datos comprimidos.

- Cookies. La conexión entre el cliente y el servidor se libera cuando finaliza la recepción del recurso solicitado. En el protocolo HTTP no se describe cómo recordar conexiones anteriores. Es decir, de una petición a la siguiente no se conserva ningún tipo de información, cada conexión es independiente y en principio no hay una memoria de conexiones del cliente. Para solventar esta situación, aparte de la caché, se crearon las *cookies*, que son ficheros de texto que intercambian los clientes y servidores que permiten recordar conexiones anteriores.

  Las cookies permiten a los servidores enviar en la cabecera texto en una respuesta HTTP, la cual podrá quedar almacenada en el cliente si la configuración de éste lo permite. Posteriormente el navegador podrá enviar la cookie al mismo servidor cuando realice una petición posterior.

  Así, una manera de identificar a los usuarios y conexiones aunque se vayan visitando distintas páginas web es almacenando cookies en el cliente y luego enviarlas al servidor para ver si sus valores son los mismos o han cambiado.

  Los servidores envían las cookies usando las cabeceras Cookies y Set-Cookie incluyendo información relativa al nombre de la cookie (*Name*), su valor (*Value*), fecha de expiración (*Expires*), ruta donde almacenarse (*Path*), nombre de dominio del servidor que la envía (*Domain*), etc.

  Cuando el navegador realiza una petición a un servidor web, consulta las cookies que tiene almacenadas, y si existe alguna que no haya caducado y cuya ruta y dominio coinciden con los de la petición, se las envía al servidor usando las cabeceras Cookies y Set-Cookie.

  Es importante recordar que las cookies pueden ser aceptadas, bloqueadas o borradas según esté configurado el navegador del cliente.

7.4. Códigos de estado y error.

Los códigos de estado y error los envían los servidores en las respuestas HTTP con el fin de informar a los clientes de cómo ha sido procesada la petición. A continuación del código se presenta un texto explicativo del código enviado.

Los códigos son números de tres cifras que se clasifican en 5 tipos en función del primer dígito.

- 100 – 199. Son códigos informativos que indican que el servidor ha recibido la petición pero que no ha sido todavía procesada.
- 200 – 299. Indican éxito en la petición. Por ejemplo el típico 200 OK.
- 300 – 399. De redirección indicando que la petición ha sido procesada, pero redirigida a otra localización.
- 400 – 499. Errores del cliente debido a un error en la petición o que ésta no se puede conceder (errores de sintaxis, acceso no autorizado, recurso no disponible, etc.)
- 500 – 599. Errores del servidor al recibir la petición (fallo de configuración, sobrecarga, etc.)

Para obtener una lista de los códigos de error y sus significados se puede consultar en la dirección http://es.wikipedia.org/wiki/Anexo:Códigos_de_estado_HTTP.

7.5. Autentificación y Seguridad.

El protocolo HTTP permite mecanismos de autentificación para controlar los recursos que ofrecen los servidores. Con la cabecera WWW-Authenticate el servidor pide autentificación al cliente, y con la cabecera Authoritation el cliente envía al servidor sus credenciales (usuario y clave de acceso). HTTP establece dos mecanismos para enviar al servidor los credenciales del cliente de forma codificada:

- **Basic**. El cliente envía al servidor sus credenciales codificados con el algoritmo *base64*. Este algoritmo no es seguro y además es muy fácil de capturar y descifrar.
- **Digest**. En este caso se envía al servidor un resumen criptográfico (*hash*) en el que se combinan el nombre de usuario, la petición y la contraseña usando el algoritmo *MD5*. Aunque es más seguro que el anterior, es también vulnerable.

Los mecanismos de autentificación anteriores realmente no son seguros, de forma que con ellos no se garantiza que el cliente sea quien dice ser. Respecto a los servidores, HTTP no provee mecanismos de autentificación para ellos, de forma que realmente no se puede comprobar si el servidor es quien dice ser.

El uso de cookies para autentificar a usuarios frente a servidores tampoco es nada seguro. Existe la posibilidad de falsificar las *cookies* y los parámetros que se envían mediante una petición GET en la URL pudiéndose suplantar la identidad de los usuarios.

El propio protocolo HTTP no es nada seguro ya que el intercambio de petición/respuesta se realiza en texto plano y por lo tanto es vulnerable a los ataques de análisis de tráfico de red.

Además, los propios clientes, servidores y aplicaciones web tienen vulnerabilidades que pueden ser aprovechadas por atacantes para obtener datos o desestabilizar los sistemas.

Estas carencias en HTTP ha provocado que la autentificación se establezca mediante aplicaciones web (se envía los credenciales previamente cifrados al servidor mediante un formulario donde se tratan mediante una aplicación), y que la seguridad se garantice usando certificados digitales y HTTPS.

7.6. Conexiones persistentes.

Las llamadas conexiones persistentes en HTTP consiste en que varias peticiones y sus respuestas se transfieren usando la misma conexión TCP. Por lo tanto al reducirse el número de conexiones, se reduce el uso de memoria y tiempo de CPU y por lo tanto disminuye el tiempo de respuesta.

Dependiendo de la versión del protocolo, este tipo de conexiones están desactivadas o no. En HTTP /1.0 por defecto se establece una conexión TCP por cada petición/respuesta. Se pueden activar usando la cabecera *Connection:keep-alive*.

En HTTP/1.1 las conexiones persistentes están activadas por defecto, pero es necesario controlar el tiempo de conexión y el número máximo de peticiones en una misma conexión. Para ello se usan las cabeceras *Keep AliveTimeout* y *MaxKeepAliveRequests* respectivamente,

No todo son ventajeas en el uso de conexiones persistentes. De hecho, supone un gran problema para las aplicaciones web que trabajan con bases de datos y con múltiples usuarios en un mismo instante. La cantidad de conexiones abiertas a una base de datos es limitada y podemos saturar al servidor de bases de datos.

8. HTTPS. Protocolo seguro.

HTTPS (*HTTP Secure*) es un protocolo que utiliza SSL (*Secure Sockets Layer*) o TLS (*Transport Layer Security*) para encapsular de forma cifrada los mensajes HTTP. Estos protocolos proporcionan mecanismos de encriptación de la información para garantizar la privacidad de la información, y el uso de certificados digitales para asegurar la autenticidad de los servidores y de los clientes. Este protocolo es ampliamente utilizado, y de hecho los soportan todos los navegadores.

Los servidores web que usan el protocolo HTTPS, por defecto escuchan las peticiones de los clientes por el puerto 443/TCP; y los clientes deben indicar en la URL el protocolo **https** en vez de http.

Como hemos comentado, el uso de certificados permite identificar a las partes que intervienen en una conexión. Los *certificados de servidor* permiten acreditar a un servidor web en el sentido de que dicho certificado asegura que el servidor web es realmente quien dice ser. Un certificado lo emite una autoridad certificadora (AC), una especie de notario digital, en el cual podemos confiar (o no) en los certificados que emite. Si un usuario confía en una AC e instala su "certificado raíz", está asumiendo que confía en todos los certificados que emita dicha AC.

Por ejemplo, en España, la FNMT (Fábrica Nacional de Moneda y Timbres) es una AC que emite certificados. La AEAT (la Agencia Tributaria) dispone de servidores web certificados por la FNMT. Si un usuario instala el certificado raíz emitido por la FNMT, quiere decir que confiamos en la FNMT como AC y por lo tanto en el certificado que nos presente la AEAT emitido por la FNMT.

Si al conectarnos a un sitio web mediante HTTPS nos presenta un certificado, no quiere decir que sea un sitio seguro, sino que tiene un certificado firmado por una AC, que bien pudiera ser ¡el mismo! Por eso, los navegadores nos advierten si reciben un certificado emitido por una AC que no es de nuestra confianza (no tenemos instalado su certificado raíz), y nos preguntan si confiamos en dicho certificado y en la AC que los emitió. En definitiva, es responsabilidad del cliente la decisión de aceptar o no el certificado.

Los sistemas operativos *Windows Server* en sus distintas versiones disponen de la herramienta *Certificate Server* para gestionar certificados. Esta función del servidor permite implantar una AC en un sistema Windows Server que actúe como controlador de dominio (AD), de tal forma que puede emitir

certificados que autentifiquen a los objetos del dominio. Se suele usar sólo dentro de organizaciones, de tal forma que los elementos de la organización pertenecientes al dominio (AD) confían en los certificados emitidos por la AC instalada en el controlador del dominio.

OpenSSL es un proyecto de software libre consistente en un paquete de herramientas de administración y bibliotecas relacionadas con la criptografía, que suministran funciones criptográficas a otros protocolos como son:

- SSH para usar terminal remoto seguro, y SCP para la copia remota de archivos.
- HTTPS para navegadores y servidores web permitiendo el acceso seguro a sitios web.
- FTPS para clientes y servidores FTP para permitir la transferencia de archivos segura.
- SMTPS para clientes y servidores de correo seguro; y clientes POPS e IMAPS para acceso a seguro los buzones de mensajes.

Estas herramientas que suministra *OpenSSL* ayudan al sistema a implementar los protocolos SSL y TLS, y a crear certificados digitales que pueden usarse para identificar a un servidor web.

Cuando un sitio web necesita un certificado para identificarse, dependiendo del entorno puede usar las herramientas *Certificate Server* (Windows Server) u *OpenSSL (software libre)*. Existen dos maneras de generar un certificado.

- Generar una *petición de certificado* que se envía a una AC, la cual verifica la identidad del solicitante; y si son ciertos los datos. genera un certificado firmado por dicha AC. Casi la totalidad de las AC cobran por emitir certificados, aunque van apareciendo algunas que los emiten gratuitamente.
- Una alternativa sin coste es usar certificados *autofirmados*, es decir certificados donde la petición es firmada ¡por el propio solicitante!. Se suelen usar en servidores para pruebas o en entornos privados. Veremos cómo generar certificados autofirmados tanto en Windows Server como en Linux.

En las comunicaciones seguras, aparte de los certificados de servidor que autentifique de alguna manera a los servidores, se necesita además que se asegure la privacidad de la información transmitida. Pare ello se usan mecanismo de cifrado para encriptar la información.

Los mecanismos de cifrado se basan en la utilización de claves para cifrar y descifrar los mensajes. La criptografía simétrica usa la misma clave para cifrar y descifrar los mensajes, es un sistema rápido de cifrado pero tiene el inconveniente de que es menos seguro ya que el emisor y el receptor deben intercambiarse las claves previamente.

En la criptografía asimétrica, la que se usa en SSL y TLS, existen dos claves diferentes: una para cifrar y otra para descifrar. La clave de cifrado es pública, esto es, conocida por todo el mundo; la de descifrado (clave privada) solamente es conocida por su propietario. Ambas constituyen un par de claves.

Para comprender como es el funcionamiento supongamos que tanto Anastasia como Bartolo tienen un par de claves pública/privada. Si Anastasia desea enviar un mensaje a Bartolo, los pasos que debe seguir son los siguientes:

- Anastasia obtiene la clave pública de Bartolo.
- Anastasia compone el mensaje y lo cifra con la clave pública de Bartolo.
- Anastasia envía el mensaje cifrado.
- Bartolo recibe el mensaje y le aplica su clave privada para descifrarlo, obteniendo el mensaje.

Es decir, cualquiera que disponga de la clave pública de Bartolo, puede cifrar un mensaje y enviárselo, pero solamente él podrá descifrar los mensajes que le llegan. Nadie más; ni siquiera Anastasia podrá descifrar el mensaje que ella misma acaba de cifrar.

La base de la seguridad en los procesos de cifrado está en la fortaleza de los algoritmos de cifrado o encriptación. Algunos de ellos son RSA, DES, TDES (Triple DES), AES, X.509.

Volviendo al caso de los servidores web. Los certificados de servidor que emite una AC contienen, además de la información que identifica al servidor, la clave pública del servidor. Como hemos dicho, esta clave pública es la que permite a los clientes encriptar la información que envían al servidor, de forma que solo el servidor web puede decodificarla con su clave privada.

Así que cuando un usuario desde un navegador se conecta a un servidor seguro (que tiene activada la seguridad), el servidor le manda su certificado. Como este certificado ha sido emitido por una AC, el navegador verifica si es correcto o no. Si es correcto, utiliza la clave pública del servidor web que recibió para encriptar mensajes y comunicarse de forma segura con el servidor.

9. Alojamiento virtual.

El alojamiento virtual de sitios web (*virtual hosting*) consiste en simular que existen varios equipos con sus respectivos sitios web sobre un único servidor web. Por ejemplo se podrían alojar los sitios web *www.infor.es* y *apuntes.infor.es* en una misma máquina o equipo.

El uso de servidores web virtuales permite reducir el número de máquinas físicas para alojar los sitios web y aprovechar mejor los recursos (CPU, memoria…) de los equipos.

El alojamiento virtual se puede implementar de tres formas no excluyentes entre sí:

- Alojamiento virtual basado en IPs. El equipo servidor usará tantas direcciones IP como servidores web virtuales. Para ello el servidor deberá tener varias tarjetas de red, o bien asignar varias IP (alias o interfaces virtuales) a una misma tarjeta de red. Este tipo de alojamiento es muy limitado debido a la necesidad de usar tantas IP como sitios web.
- Alojamiento virtual basado en nombres. Se permite alojar varios nombres de dominio sobre la misma IP, donde cada servidor virtual atiende las peticiones correspondiente a un nombre de dominio. Para ello hay que configurar un servidor DNS que asocie los diferentes nombres de dominio con la misma dirección IP, y que el servidor Web soporte un protocolo superior al HTTP/1.0. Estos protocolos admiten en el mensaje de petición la cabecera `Host`, a la cual se le otorga como valor el nombre de dominio del servidor solicitado. Así el servidor Web recibirá el valor de esta cabecera y sabrá qué sitio es el solicitado de todos que aloja. Este método es el más usado.
- Alojamiento virtual basado en puertos. Cada servidor virtual atiende por una misma dirección IP pero por un puerto diferente. En este caso el cliente tiene que añadir a la URL de la petición el puerto.

Los tipos anteriores se pueden combinar, todo depende de que el software del servidor permita este tipo de configuraciones mixtas.

10. Instalación y configuración IIS en Windows Server.

Los servicios de *Internet Information Server* (IIS), permiten la creación, configuración y administración de sitios web, además de permitir otras funciones de Internet gracias a que soporta los protocolos *File Transfer Protocol* (FTP) o protocolo de transferencia de archivos, *Post Office Protocol* (POP; sólo en Windows 2003 y anteriores) o protocolo de Oficina de Correos, así como *Simple Mail Transfer Protocol* (SMTP) o protocolo simple de transferencia de correo.

Los servicios de *Internet Information Server* permiten la publicación de información en una Intranet o en Internet, así como comunicaciones seguras mediante el protocolo SSL/TLS. Además, utilizando componentes y lenguajes de servidor, se pueden crear contenidos dinámicos mediante lenguajes de script como *Active Server Page* (ASP) o PHP.

10.1. Instalación de IIS en Windows Server 2012.

La instalación del servicio IIS comienza desde el "Administración del servidor", y una vez situados en el “Panel”, activando la opción *Agregar Roles y características*. Se lanzará el asistente de instalación. Pulsando *Siguiente* elegimos la opción por defecto *Instalación basada en características o roles.* A continuación elegimos sobre qué equipo servidor se instalará el servicio. Por defecto se presenta el equipo local. Al pulsar *Siguiente* aparece una lista con los roles a instalar. Marcamos la casilla *Servidor web (IIS)*. Se presentará una nueva ventana informando de las características adicionales que se instalarán. Pulsamos en el botón *Agregar características* para cerrar la ventana y pulsamos *Siguiente* para avanzar en la instalación. Se presenta una nueva lista de características la cual ignoraremos pulsando en *Siguiente*.

Llegado a este punto, es posible agregar diferentes opciones de configuración al servidor web. Por defecto aparece una lista con aquellas opciones que se instalarán por defecto, pero podríamos marcar aquellas otras que nos interesen. No obstante, una vez instalado el servicio, siempre se pueden añadir o modificar estas opciones más tarde. Pulsamos *Siguiente* y posteriormente el botón *Instalar* para empezar la instalación del servicio.

Cuando finalice el proceso de instalación, se creará dentro de "*Herramientas de administración*" el acceso directo "*Administrador de Internet Information Server*". Activando este enlace se mostrará la ventana de administración de los servicios de IIS.

IIS se administra editando propiedades en la consola de administración. Se dispone de cuatro niveles de administración: nivel servidor, del sitio web, nivel directorio y nivel de archivos. Hay que tener en cuenta que lo que se configure a un nivel, lo heredan de modo automático los niveles inferiores, aunque siempre se pueden sobrescribir los parámetros heredados y configurar ésta herencia mediante la "Delegación de características".

Podemos comprobar que el servidor está a la escucha en TCP/80 con el comando:

```
netstat -a -p TCP -n
```

10.2. Servidor virtual por defecto.

En el panel izquierdo del administrador de IIS se presentan los equipos a administrar. En principio sólo aparece la máquina local, pero se puede realizar una conexión a otro servidor y administrar los servicios web de dicho servidor si tenemos cuenta de administrador en él.

Expandiendo el icono que representa el servidor, aparecen carpetas con los nombres de los servicios instalados, y entre ellas "*Sitios*". Si expandimos esta carpeta aparece otra con el nombre "*Sitio Web predeterminado*", la cual contiene la información de configuración de dicho sitio web.

Si ubicados sobre el sitio web "*Sitio web predeterminado*" pulsamos en la columna de la derecha en *Configuración básica* se mostrará una ventana en la que se indica que la ruta del directorio raíz de este sitio web es %SystemDrive%\inetpub\wwwroot.

Normalmente no se suele usar este sitio web que por defecto instala el servidor IIS para la publicación web, básicamente porque el mismo ha sido creado con fines administrativos para gestionar aplicaciones tales como el acceso web de Escritorio remoto, o el acceso web a las impresoras del equipo.

Para comprobar el funcionamiento de este sitio web, desde el propio equipo servidor, escribimos en la URL de un navegador la IP 127.0.0.1 o bien localhost, presentándose la página por defecto de dicho sitio (iisstart.htm). También responderá si usamos la dirección IP del servidor o su nombre de dominio siempre que un servidor DNS lo resuelva.

Si creamos el archivo C:\inetpub\wwwroot\asir\saludo.html con el contenido HTML que queramos, y luego desplegamos el sitio web predeterminado en la consola de administración, veremos que se ha creado una entrada para el directorio asir. Si observamos las opciones de configuración del directorio comprobamos que son las mismas que las definidas a nivel del sitio web (las del directorio raíz).

Desde la consola de administración es fácil determinar para el sitio web y para los directorios que cuelgan del directorio raíz, cuáles serán los archivos a servir por defecto. Para ello se usa la opción *Documento Predeterminado"*.

También cabe la posibilidad de que en vez de presentar un archivo al acceder al sitio a o sus subdirectorios, se presente una lista con el contenido de dicho sitio o directorio. Esto se consigue habilitando la opción *Examen de Directorios* y deshabilitando la opción *Documento Predeterminado"*.

Se pueden modificar las páginas y contenidos a servir cuando se producen errores. Para ello se usa la opción *Páginas de errores* y a continuación el número de error a modificar.

10.3. Configuración de servidores virtuales.

El servidor IIS permite la administración de varios sitios web (servidores virtuales). Para crear un nuevo sitio web, desde la consola de administración y con el elemento *Sitios* seleccionado, hacemos *click* con el botón derecho del ratón y elegimos *Agregar sitio web…* Esta operación presenta una ventana en donde deberemos completar los siguientes datos:

- Nombre del sitio: un nombre cualquiera que sirva de identificador del sitio.
- Ruta de acceso física: directorio local o remoto donde se alojará la jerarquía de páginas y subdirectorios del sitio web.
- Dirección IP del servidor por las que se atenderá las peticiones y el puerto de acceso. Por defecto atenderá por cualquier dirección IP y por el puerto 80 para HTTP y 443 para HTTPS. Si se utiliza HTTPS, se puede especificar el certificado SSL a utilizar.

- Nombre de hosts: Permite indicar un nombre de dominio para el sitio, que lógicamente un servidor DNS debería resolver.

Si el sitio web creado con el asistente usa la misma IP y puerto que el servidor web predeterminado, no se permitirá acceder a ambos simultáneamente. Debemos detener uno de ellos e iniciar el otro. Para detener o iniciar un servidor web podemos hacerlo a través de las opciones correspondientes del menú contextual del icono que representa a dicho servidor web.

Para probar el servidor web creado con el asistente debemos detener el servidor web predeterminado e iniciar el recién creado. El sitio web creado deberá alojar en la ruta indicada un archivo cuyo nombre deberá aparecer en la lista de los *Documentos predeterminados*. Podemos elegir uno de la lista que hay por defecto o crear uno para el sitio creado.

Una vez creado un sitio web, se pueden establecer otros muchos parámetros a través de las diferentes opciones y propiedades del sitio web. Los parámetros principales está relacionadas con:

- Modificación de enlaces para configurar la dirección IP y puerto de escucha de las peticiones. Un mismo sitio web puede escuchar por diferentes IP, así como por diferentes puertos. así como el tiempo de espera determinado para las conexiones y llevar un registro de acceso de los usuarios.
- Rendimiento. Permite establecer el ancho de banda de la conexión y el número máximo de clientes.
- Directorio particular. Establece el directorio raíz del sitio web y los privilegios de los usuarios.
- Documentos. Se indican los documentos en orden de preferencia que pueden actuar como índice del sitio. Se pueden agregar, eliminar o cambiar el orden de preferencia de los documentos.
- Errores personalizados. Se indican los códigos de error y las páginas web con los mensajes de error asociados. Se puede modificar esta asociación a otro documento con un nuevo mensaje.
- Encabezados HTTP. Permite establecer los tiempos en que las páginas se mantendrán en caché, definir encabezados personalizados que el servidor enviará a los clientes cuando éstos hagan peticiones al servidor, y asociar extensiones de archivos a tipos MIME.
- Autentificación y Autorización. (Agregando los *roles* correspondientes). Permite restringir los equipos, grupos de equipos o dominios a los que se les concederá o negará conexión al servidor. Se puede habilitar o no el acceso anónimo y/o pedir autentificación a los clientes. También permite habilitar la utilización de un canal seguro SSL y la creación de un certificado de servidor.

10.3.1. Servidores virtuales por nombre.

Como ya comentamos IIS permite disponer de varios servidores virtuales por IP, por puerto o por nombres. Para configurar servidores virtuales por nombres, una vez creados los servidores debemos mediante sus propiedades, activar la opción "*Modificar enlaces…*" de cada uno de ellos. En la ventana que aparece "*Enlace de sitios*" hay que sustituir el contenido del cuadro de texto "*Direccion IP*" *Ninguna asignada*, por la dirección IP de la interfaz por donde se comunicará el servidor; y asignar un nombre de dominio para el servidor virtual. Esta operación habrá que realizarla para cada uno de los servidores virtuales.

Así por ejemplo podemos tener dos servidores virtuales con nombres de dominio sw1.aula.izv y sw2.aula.izv en la misma dirección IP y puerto. Evidentemente debe haber un servidor DNS que resuelva los nombres de dominio de los servidores con la dirección IP de la interfaz de red donde se ubican.

Además, un mismo servidor virtual puede responder usando diferentes nombres. Sólo es necesario añadir en la ventana "*Enlace de sitios*" un nuevo enlace para el sitio con los mismos valores para la dirección IP y puerto, pero con un nombre de host distinto que por supuesto se deberá resolver.

10.4. Directorios virtuales.

Un directorio virtual no es un directorio que como tal esté incluido dentro del sitio, sino que opera al estilo de un acceso directo. Aunque se incluye dentro del sitio web, realmente se hace referencia a un directorio local o remoto que está ubicado fuera de la jerarquía de directorios del sitio web.

Para crearlos, una vez seleccionado el sitio web, se elije la opción del menú contextual *Agregar directorio virtual*. En una ventana se solicitará el nombre o alias para este directorio virtual, la carpeta a la que se hará referencia, y los permisos de acceso al directorio.

10.5. Servidores virtuales seguros.

Como ya comentamos, para implantar un servidor web seguro, necesitamos tener un certificado de servidor que autentifique dicho servidor. Si el servidor web está ubicado en Internet, lógicamente el certificado debe estar firmado por una Autoridad Certificadora (AC); pero para pruebas o intranets podemos crear certificados de servidor autofirmados.

Tenemos básicamente dos opciones para crear certificados autofirmados. La primera es usar la característica *Certificado de Servidor* que incorpora IIS, y la segunda es usar el software *OpenSSL* en su versión para Windows.

10.5.1. Certificados autofirmados mediante IIS.

Desde la consola de administración de IIS, seleccionamos en el panel izquierdo el nodo del equipo, y en el panel central activamos *Certificados de servidor*. En el panel derecho de *Acciones* activamos la opción *Crear certificado autofirmado…* En la ventana que se muestra se nos pedirá un nombre para identificar el certificado que a continuación se generará.

Para usar el certificado en un servidor web, se selecciona en la consola de administración de IIS el sitio web y activamos la opción "*Modificar enlaces…*" de su menú contextual. En la ventana *Enlaces de sitios"* activamos *Agregar* para agregar un nuevo sitio al servidor. Seleccionamos como protocolo *htpps* (HTTP Secure), seleccionamos la IP de la interfaz de red, cambiamos o confirmamos el puerto 443 y finalmente seleccionamos el certificado autofirmado de la lista de certificados.

Para probar el servidor debemos usar el protocolo HTTPS en la URL del navegador de la forma https://sw1.aula.izv suponiendo que sw1.aula.izv es el nombre de dominio del servidor web que hemos configurado.

El navegador nos presentará un mensaje de aviso diciendo que la conexión no es segura ya que la identidad del servidor no puede ser verificada. Esto ya la sabemos porque el certificado no está emitido por una AC de confianza para el navegador (no tiene instalado su certificado raíz). Tendremos que añadir una excepción y confirmarla para poder acceder al sitio web.

En la configuración del servidor seguro hemos posibilitado su acceso mediante HTTPS, pero todavía podemos seguir accediendo usando HTTP. Para obligar a que sólo se pueda acceder a través de HTTPS, deberemos desde la consola de administración del IIS, seleccionar el sitio web seguro, activar la característica *Configuracion SSL* y marcar la casilla "*Requerir SSL*". Finalmente pulsamos *Aplicar*.

10.5.2. Certificados autofirmados mediante OpenSSL.

Con las herramientas *OpenSSL* se pueden generar claves privadas y los llamados CSR (solicitud de firma de certificado). Normalmente dicha solicitud se envía a una CA, la cual genera el certificado pedido avalado por su firma; pero ahora queremos generar certificados autofirmados.

De OpenSSL existen distintas versiones ya compiladas para diferentes entornos. En la web http://slproweb.com podemos encontrar varias de ellas. Para Windows en su versión de 64 bits usaremos el archivo https://sourceforge.net/projects/openssl/files/openssl-1.0.2j-fips-x86_64. Puede, dependiendo de la versión de Windows usada, que antes debamos instalar en el sistema un conjunto de librerías que permitan ejecutar código escrito en Visual C++. Desde las páginas oficiales de Microsoft se pueden descargar.

Si la versión de *OpenSSL* es instalable, durante su instalación, se nos preguntará dónde colocar las bibliotecas de código que necesita la aplicación. Le indicaremos que no use el mismo lugar que el que usa Windows. También puede ser necesario añadir una variable de entorno al sistema para que localice sin problemas el archivo de configuración openssl.cnf. Suponiendo que se ubica en C:\OpenSSL-Win64\bin, sería:

```
set OPENSSL_CONF=C:\OpenSSL-Win64\bin
```

Supondremos que ya se dispone de un servidor virtual por nombre denominado seg.aula.izv. para el cual crearemos un certificado de servidor autofirmado.

1. Generación de una clave privada.

Para firmar digitalmente necesitamos generar una clave privada. Crearemos una de tipo RSA de 1024 bits, cifrada con Triple-DES y guardada en formato PEM (legible). Para ello ejecutamos la orden:

```
C:\OpenSSL-Win64\bin>openssl genrsa -des3 -out servidor.key 1024
```

```
Loading 'screen' into random state - done
Generating RSA private key, 1024 bit long modulus
.....................................++++++
.....................................++++++
e is 65537 (0x10001)
Enter pass phrase for server.key: ******
Verifying - Enter pass phrase for server.key: ******
```

Vemos como nos pide una contraseña, la cual habrá que repetir. Está contraseña protege nuestra clave privada y nos la pedirá cada vez que firmemos con ella. La clave generada se ha almacenado en el archivo servidor.key.

2. Generar un CSR (Solicitud de firma de certificado).

Para realizar la petición de un certificado hay que generar un archivo con la información referente a quien realiza la petición del certificado y firmada con la clave privada del solicitante.

Para ello ejecutamos:

```
C:\OpenSSL-Win64\bin>openssl req -new -key servidor.key  -out servidor.csr
Enter pass phrase for server.key: ******
Loading 'screen' into random state - done
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Granada
Locality Name (eg, city) []:Granada
Organization Name (eg, company) [Internet Widgits Pty Ltd]:IZV
Organizational Unit Name (eg, section) []:Dpto. Informatica
Common Name (e.g. server FQDN or YOUR name) []:seg.aula.izv
Email Address []:yomismo@gmail.com
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:******
An optional company name []:IESZV
```

En este comando debemos indicar cuál es el fichero con la clave privada generada en el paso anterior (servidor.key), el nombre del archivo que contendrá la petición (servidor.csr) y dar los datos que queremos que aparezcan en el certificado. Entre los datos, lo más importante es indicar correctamente el nombre del servidor web para el cual generaremos el certificado (Common Name).

3. Crear un certificado autofirmado.

La petición de certificado generada anteriormente la firmaremos nosotros mismos (por eso es autofirmado…), usando para firmarla nuestra clave privada. El certificado generado lo guardaremos en el archivo certificado.crt. Para ello ejecutamos el comando:

```
C:\OpenSSL-Win64\bin>openssl x509 -req -days 365 -in servidor.csr -signkey servidor.key  -out certificado.crt
Loading 'screen' into random state - done
Signature ok
subject=/C=ES/ST=Granada/L=Granada/O=izv/OU=Dpto. Informatica/CN=seg.aula.izv
Getting CA Private Key
Enter pass phrase for servidor.key: ******
```

4. Conversión a formato *pfx*.

El certificado generado debe importarse por parte de IIS para que pueda usarlo, pero desgraciadamente IIS sólo importa certificados en formato *pfx*. Así que debemos convertirlo a dicho

formato. Para ello primero convertimos el certificado a formato PEM, y luego lo convertimos junto a la clave privada, en formato *pfx*.

```
C:\OpenSSL-Win64\bin>openssl x509 -in certificado.crt -out certificado.pem
C:\OpenSSL-Win64\bin>openssl pkcs12  -export -out certificado.pfx  -inkey servidor.key -in certificado.pem
```

5. Importar certificado.

Para importar el certificado, desde la consola de administración de IIS, seleccionamos en el panel izquierdo el nodo del equipo, y en el panel de la derecha activamos *Certificados de servidor*. En el panel derecho de Acciones activamos *Importar…* apareciendo una ventana donde se seleccionará el archivo que contiene el certificado y se introducirá también la contraseña de la clave privada para su protección. De esta forma queda almacenado en IIS para usarlo para el sitio seg.aula.izv.

11. Instalación y configuración de Apache en Linux.

La Apache Software Foundation (http://www.apache.org) es una fundación que patrocina y colabora en múltiples desarrollos software, siendo uno de ellos el servidor web *open source* Apache. El servidor web Apache es uno de los más usados a nivel mundial existiendo versiones para diferentes sistemas operativos (Unix, BSD, Linux, Windows, Novell NetWare, etc.).

Apache está desarrollado de forma modular, lo que permite ampliar la funcionalidad mediante la instalación de módulos específicos. Por otra parte sus posibilidades de configuración son muy numerosas, aunque como veremos, nada más instalado ya es completamente funcional.

11.1. Instalación de Apache.

La instalación del servidor web se realiza mediante:

```
$ sudo apt-get install apache2
```

Cuando termina la instalación se inicia el servidor, pero puede además mostrar un mensaje de advertencia indicando que no es capaz de determinar cuál es el nombre de dominio completo del equipo. Posteriormente comentaremos a que se debe esta advertencia.

Es importante señalar que algunas directivas pueden cambiar según la versión de Apache. Terminada la instalación debemos comprobar que Apache está en ejecución, y que el servicio está a la escucha en el puerto 80. Para ello ejecutamos:

```
$ ps -ef | grep apache
$ netstat -ltn
```

El servicio de Apache se manipula usando los siguientes comandos dependiendo de cuál sea la versión del sistema de inicio y gestión de servicios de la distribución empleada.

```
$ sudo /etc/init.d/apache2 {start | stop | restart | reload | status}
$ sudo service apache2 {start | stop | restart | reload | status}
$ sudo systemctl {start | stop | restart | reload | status} apache2
```

También se puede usar el comando apachectl el cual invoca directamente al software servidor httpd

```
$ sudo apachectl {start | stop | restart | reload | status | fullstatus | configtest} apache2
```

Al finalizar la instalación, aparte de crear los archivos de configuración, se crean

- El usuario *www-data* incluido en el grupo *www-data*. Este usuario es el propietario de los procesos hijos de Apache que se encargan de atender las peticiones de los clientes.
- El directorio /var/www/html, cuyo propietario y grupo es *root*. Es el directorio raíz del servidor virtual por defecto y contiene el archivo index.html, el cual se sirve cuando acedemos a dicho servidor. (Ojo. Este directorio puede cambiar según la versión de Apache).

Para acceder al servidor por defecto, solo debemos usar en la URL del navegador http://127.0.0.1 o cualquier dirección perteneciente al bucle local (127.0.0.0/8). Si como es normal, en el archivo /etc/hosts se tiene la entrada localhost asociada a una IP del bucle local (127.0.0.0/8), también responderá por http://localhost. De igual manera sucederá si se usa otro nombre asociado a una IP del bucle local.

Si por ejemplo el archivo /etc/hosts contiene la línea

```
127.0.0.1       localhost ubu www.ubu azofaifa bartolillo
```

Cualquier nombre de la lista que quede asociado a cualquier dirección del bucle local se podrá usar para acceder al servidor por defecto.

Apache se configura editando directivas en los archivos de configuración ubicados en */etc/apache2*. Los archivos principales son:

- /etc/apache2/apache2.conf. En este archivo se determina el comportamiento general del servidor. Aquí encontramos entre otras las directivas *ServerRoot, User, Group, Timeout, KeepAlive, ErrorLog*. También están las directivas de seguridad para el acceso o no a ciertos directorios. Las directivas que no se especifiquen en este archivo de configuración toman el valor por defecto. Desde este archivo se incluyen (usando las directivas *Include* e *IncludeOptional*) otros ficheros de configuración.
- /etc/apache2/ports.conf. Aquí se define los puertos, o las interfaces de red junto a los puertos por los que escuchará el servidor.
- /etc/apache2/envvars. Se definen variables de entorno para el comando apache2ctl.

Además de estos archivos, existen otros ubicados en subdirectorios contenidos en /etc/apache2.

- Directorios de configuración de módulos.
    - /etc/apache2/mods-available. Es el directorio de configuración de los módulos *disponibles*. Contiene archivos .load y .conf. Los archivos .load contienen lo necesario para cargar un módulo en concreto; y los archivos .conf asociados a los anteriores, contiene la configuración básica para iniciar dichos módulos.
    - /etc/apache2/mods-enabled. Es el directorio de configuración de los módulos *habilitados*. Contiene enlaces simbólicos a los archivos del directorio mods-available de forma que los enlaces que se encuentre en este directorio serán los módulos a cargar al iniciar Apache.

      Para activar un módulo se emplea el comando a2enmod modulo creándose el enlace. Para desactivar un módulo, se elimina el enlace con el comando a2dismod módulo.
- Directorios de configuración de servidores virtuales.
    - /etc/apache2/sites-available. Directorio de configuración de sitios virtuales *disponibles*. Cada archivo del directorio contiene la configuración de un sitio virtual disponible. Por defecto está creado el archivo 000-default.conf con la configuración del servidor virtual por defecto.
    - /etc/apache2/sites-enabled. Es el directorio de configuración de sitios virtuales *habilitados*. Contiene enlaces simbólicos a los ficheros del directorio sites-available, de forma que los enlaces que se encuentren en este directorio serán los servidores virtuales que están habilitados. Por defecto está creado el enlace 000-default.conf al archivo 000-default.conf del directorio sites-available.

      Para habilitar un servidor virtual se emplea el comando a2ensite archivo_sitio creándose el enlace. Para deshabilitar un servidor virtual, se elimina el enlace con el comando a2ensite archivo_sitio.
- Directorios de configuraciones genéricas.
    - /etc/apache2/conf-available. Contiene archivos con configuraciones relacionadas con la internacionalización de los mensajes de error, personalización del conjunto de caracteres (por defecto UTF-8). Además suele ser el lugar donde se incluyen los archivos de configuración de las aplicaciones web que se instalan para ejecutarse sobre Apache. (Este directorio hace el papel que hacía /etc/apache2/conf.d en versiones anteriores de Apache).
    - /etc/apache2/conf-enabled. Contiene enlaces simbólicos a los ficheros del directorio conf-available, de forma que los enlaces que se encuentren en este directorio serán los archivos de configuración que están activos.

      Para activar una configuración concreta se emplea el comando a2enconf archivo_configuración creándose el enlace. Para desactivar una configuración, se elimina el enlace con el comando a2disconf archivo_configuración.

Si se realizan cambios en los archivos de configuración de Apache, deberemos reiniciar el servicio

11.2. Servidor virtual por defecto.

Apache diferencia entre el *servidor principal* y los *servidores virtuales*. Si no se configuran servidores virtuales, el servidor principal es el que atiende las peticiones. Pero si se activan servidores virtuales, el servidor principal no tiene efecto.

En las distribuciones tipo *Debian*, la instalación genera una configuración predeterminada para usar servidores virtuales por nombre (de hecho define uno) y no se define ninguna para el servidor principal.

Podemos comprobarlo editando el archivo /etc/apache2/ports.conf y observando cómo existe la directiva Listen 80 indicando que el servidor estará escuchando por todas las IP por el puerto 80, y viendo el archivo /etc/apache2/sites-available/000-default.conf conteniendo la configuración de un servidor virtual que como ya comentamos es el único habilitado gracias al enlace /etc/apache2/sites-enabled/000-default.conf. Este es el servidor virtual por defecto porque es el único habilitado por defecto cuando se instala Apache.

En el archivo /etc/apache2/sites-available/000-default.conf están las siguientes directivas:

- <VirtualHost *:80>. Cada servidor virtual se define mediante una directiva <VirtualHost IP:puerto>. En este caso, la del servidor virtual por defecto, escucha en todas las direcciones IP (por eso el *) y por el puerto 80.
- DocumentRoot permite indicar cuál es el directorio raíz del servidor, siendo en este caso /var/www/html.
- La directiva ErrorLog indica el archivo de *logs* del servidor virtual por defecto.
- La directiva CustomLog indica el archivo con el registro de accesos al servidor virtual por defecto.

Para comprobar el funcionamiento del servidor virtual por defecto podemos realizar las siguientes operaciones:

1. Usando en el navegador las URLs 127.0.0.1, localhost o la IP de nuestra interfaz de red, se presentará la página del servidor virtual por defecto. El motivo es que este servidor se sirve para todas aquellas peticiones que se realicen por cualquier interfaz de red. (Recordar el *).
2. Si queremos usar un nombre de dominio pero solo accesible desde la máquina local, bastará con que en el archivo /etc/hosts añadamos una entrada con dicho nombre de dominio. Si queremos que dicho nombre de dominio esté disponible de forma general, necesitaríamos un servidor DNS que resolviera dicho nombre de dominio y lo asociara a la IP de nuestra interfaz de red.
3. Crear en /var/www/html el archivo sri.html. Probar el acceso a la URL http://localhost/sri.html
4. Crear el directorio /var/www/html/datos y dentro de él, un archivo, por ejemplo datos1.html.
5. Probar el acceso a la URL http://localhost/datos/datos1.html
6. Configurar el servidor DNS para acceder al servidor por su nombre de dominio.

11.3. Directivas.

Las directivas permiten la configuración del servidor. Según donde se escriban afectarán al servidor en general, a un servidor virtual, a un directorio concreto, a un archivo, etc.

Existen directivas, como <VirtualHost>, <Directory>, <Files>, que actúan como contenedores de otras directivas permitiendo configurar ámbitos del servidor concretos.

Como ya comentamos, el archivo /etc/apache2/apache2.conf define la configuración general del servidor. Las directivas que aparecen se pueden clasificar en tres tipos, las cuales pueden aparecer mezcladas, desordenadas o ubicadas en otros archivos de configuración.

- **Directivas globales**. Definen los aspectos globales del servidor. Por ejemplo, número máximo de clientes simultáneos, tiempo de cancelación de conexión por falta de respuesta (timeout), directorios de los archivos de configuración (ServerRoot), conexiones persistentes, etc.
- **Directivas de funcionamiento del servidor principal**. Agrupa las directivas que define el comportamiento del servidor principal, y por ende se define el comportamiento por defecto de todos los servidores virtuales, en particular el servidor virtual por defecto. (Directivas User, Group, IncludeOptional mods-enabled/*.load, IncludeOptional mods-enabled/*.conf, Include ports.conf)
- **Directivas de configuración de los servidores virtuales.** Directivas IncludeOptional conf-enabled/*.conf, IncludeOptional sites-enabled/*.conf

Importante. Hay que tener en cuenta que las directivas se heredan, de forma que las establecidas para el servidor principal se heredan en los servidores virtuales, salvo que en éstos se especifiquen otras. Comentaremos algunas de las directivas más usuales.

11.3.1. DirectoryIndex.

En los archivos de configuración de los servidores virtuales (como por ejemplo 000-default.conf) se puede usar una directiva para indicar cuáles son los archivos a servir cuando no se especifica ninguno. Por herencia se establece que sea el archivo index.html, pero podemos cambiarlo. Por ejemplo podemos editar el archivo 000-default.conf y añadir la directiva DirectoryIndex sri.html

```
<VirtualHost *:80>
        DocumentRoot /var/www/html
        DirectoryIndex sri.html
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Ahora si accedemos a http://localhost se nos servirá el archivo sri.html en lugar del archivo index.html.

11.3.2. Directory y Options Indexes.

La directiva <Directory> permite determinar cómo Apache sirve el contenido de un directorio. Por herencia, todos los directorios contenidos en uno dado, heredan la configuración de éste, salvo que usemos la directiva <Directory> para indicar cómo se debe servir el directorio contenido.

En el archivo de configuración general /etc/apache2/apache2.conf aparece la siguientes directivas:

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

que nos dice cómo servir el directorio /var/www y por herencia todos los contenidos en él, como es el caso del directorio /var/www/html del servidor virtual por defecto.

La directiva Options pueda usar varios parámetros. Indexes presenta en el navegador un listado de los archivos y directorios contenidos en el directorio, pero sólo cuando no exista una directiva DirectoryIndex que actúe en dicho directorio, en cuyo caso se presentaría el archivo indicado en DirectoryIndex.

En el caso de que para el directorio a servir, no actúe una directiva DirectoryIndex, ni tampoco la directiva Options Indexes, entonces el servidor retornará el código *403 Forbbiden*.

Por ejemplo, ¿qué sucedería si pido http://localhost/datos?. Pues que se debería servir el archivo /var/www/html/datos/sri.html. pero como no existe el archivo sri.html y la directiva Options Indexes actúa por herencia (en la configuración de /var/www está activada la opción *Indexes*), se mostraría un listado con el contenido del directorio /var/www/html/datos.

Si queremos deshabilitar el listado de un directorio que actúa por herencia de otro superior, podemos usar la sintaxis Options -Indexes en la configuración de dicho directorio.

Cambiemos el contenido del archivo 000-default.conf añadiendo una directiva <Directory> que defina explícitamente como definir el directorio /var/www/html/datos

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
        <Directory /var/www/html/datos>
                DirectoryIndex datos1.html
                Options -Indexes
        </Directory>
</VirtualHost>
```

¿Qué sucedería ahora al solicitar http://localhost/datos?. Si existiera el archivo datos1.html, éste sería el que se mostraría, y si dicho archivo no existiese, al tener deshabilitada la opción del listado del directorio que actuaba por herencia, el servidor retornaría el código *403 Forbbiden*.

11.3.3. Logs.

En los archivos de configuración existen directivas que permiten configurar los asuntos relacionados con los *logs* del servicio. Estas directivas son:

- ErrorLog permite indicar el archivo de *logs* de errores.
- LogLevel indica el nivel de prioridad. (Mostrar sólo los errores, los errores y advertencias, etc.)
- CustomLog permite indicar cuál es el archivo que almacenará el registro con los *logs* de acceso.
- LogFormat indica el formato del archivo de *logs*. Si no se especifica ninguno con LogFormat, se usará el formato especificado en apache2.conf para el servidor principal.

Así en el archivo 000-default.conf de configuración del servidor virtual por defecto encontramos:

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

El valor de la variable de entorno APACHE_LOG_DIR se define en el archivo /etc/apache2/envvars mediante la directiva export y su valor es /var/log/apache2, por lo que quedan determinados cuales son los archivos con los *logs* de errores y accesos para el servidor virtual por defecto. Además con la opción combined se almacena la URL pedida por el cliente e información del navegador que la realiza.

En /etc/apache2/apache2.conf se declara la directiva LogLevel warn fijando que todos los mensajes de advertencia y aquellos de nivel superior serán registrados. Los posibles valores para la directiva LogLevel son: debug, info, notice, warn, error, crit, alert y emerg.

11.3.4. Codigos de error.

La directiva ErrorDocument permite configurar el texto o el documento que se presentará cuando se produzca un error en el servidor. Por ejemplo, se puede añadir al archivo 000-default.conf la directiva

```
ErrorDocument 404 "Página no encontrada en el servidor"
```

O bien se puede indicar que se envíe al cliente una página concreta en vez de un mensaje de error:

```
ErrorDocument 404  /error404.html
```

Lógicamente el documento /var/www/html/error404.html debe existir para que pueda ser mostrado.

11.4. Directorios virtuales.

Los directorios virtuales son directorios accesibles mediante el sitio web, pero que están fuera del directorio raíz (DocumentRoot) del servidor. Se pueden crear mediante la directiva Alias, o creando enlaces simbólicos dentro del directorio raíz del servidor para que apunten a otro directorio.

- Usando la **directiva Alias**. En esta directiva se especifica la ruta "ficticia", es decir la que indicaríamos en el navegador a continuación del nombre del servidor, y luego se indica la ruta real (lugar en el disco donde se encuentra realmente los archivos del sitio).

  Por ejemplo supongamos que el directorio /home/manolo/wiki contiene una página de nombre indexwiki.html de forma que cuando se acceda con el navegador con http://localhost/wiki obtenemos la página /home/manolo/wiki/indexwiki.html. Para ello debemos usar la directiva Alias de la forma:

  ```
  Alias /wiki  /home/manolo/wiki
  ```

  Así por ejemplo, si queremos crear este directorio virtual en el servidor virtual por defecto, editaríamos el archivo 000-default.conf añadiendo el *alias*, y como servir dicho directorio:

  ```
  Alias /wiki /home/manolo/wiki
  <Directory /home/manolo/wiki>
        DirectoryIndex indexwiki.html
        Require all granted
  </Directory>
  ```

- Con la directiva **Options FollowSymLinks** se habilita la posibilidad de que si en el directorio de un sitio web existen enlaces simbólicos *(soft links*) a otros directorios, Apache siga dichos enlaces y acceda a dichos directorios. Para ello se deben crear con anterioridad dichos enlaces simbólicos con el comando Linux ln.

  Por ejemplo supongamos que el directorio /home/manolo/blog contiene una página de nombre indexblog.html de forma que cuando se acceda con el navegador con http://localhost/blog obtenemos

la página /home/manolo/blog/indexblog.html. Si queremos implementar este directorio virtual en el servidor virtual por defecto, lo primero sería crear el enlace simbólico:

```
$ sudo ln -s /home/manolo/blog /var/www/html/blog
```

A continuación editamos el archivo 000-default.conf indicando como servir dicho directorio:

```
<Directory /var/www/html/blog>
        DirectoryIndex indexblog.html
</Directory>
```

Y lo más importante: hemos de asegurarnos que en la configuración del directorio /var/www/html (donde está el enlace), la opción Options FollowSymLinks esté activada (que lo está por herencia de la configuración de /var/www), para que los enlaces simbólicos sean seguidos por Apache..

11.5. Configuración modular.

Como ya comentamos anteriormente, Apache es un servidor modular permitiéndose añadir nuevas funcionalidades añadiendo módulos adicionales. Cada módulo agrupa un conjunto de funcionalidades y directivas para configurarlas. El número de módulos y sus funcionalidades son muy amplios pudiéndose consultar en http://httpd.apache.org/docs/2.4/es/mod.

Existen dos tipos de módulos: *estáticos* (se añaden cuando se compila Apache) y *dinámicos* (se cargan de forma dinámica cuando se inicia el servidor).

Para conocer cuáles son los módulos estáticos cargados se usa la orden:

```
$ apache2ctl  -l
```

Los módulos dinámicos se cargan con la directiva LoadModule; y mediante la directiva <IfModule nombremodulo> se permite especificar directivas que se tendrán en cuenta sólo si el módulo indicado está cargado.

Para comprobar cuales son los módulos que se han cargado dinámicamente al arrancar el servidor, sólo debemos mirar el contenido del directorio /etc/apache2/mods-enabled. Estos archivos son enlaces simbólicos a archivos .load y .conf del directorio /etc/apache2/mods-available.

Por ejemplo, el archivo alias.load contiene la directiva

```
LoadModule alias_module /usr/lib/apache2/modules/mod_alias.so
```

que indica cuál y dónde está el código del módulo a cargar.

El archivo alias.conf contiene la *directiva* <IfModule alias_module> la cual incluye las directivas que se ejecutarán si se carga dicho módulo.

Para deshabilitar un módulo, es decir, borrar el enlace simbólico del directorio mods-enabled se usa el comando a2dismod de la forma

```
$ sudo a2dismod nombre_modulo
```

Y para habilitar un módulo, mediante el comando a2enmod.

```
$ sudo a2enmod nombre_modulo
```

En el directorio /usr/lib/apache2/modules se pueden ver los módulos disponibles para carga.

Mediante el siguiente comando

```
$ sudo apt-cache search libapache2-mod
```

podemos mostrar los paquetes disponibles en los repositorios de Ubuntu que permiten instalar módulos adicionales para Apache.

11.5.1. Módulo *userdir*.

La activación del módulo userdir permite que cualquier usuario con cuenta en el equipo donde corre el servidor, cree su propio sitio web en un directorio dentro de su *home*. Si editamos el archivo /etc/apache2/mods-available/userdir.conf podemos observar que en la configuración por defecto del módulo se habilita el uso de directorios personales (excepto para *root*) y que public_html es el nombre del directorio que pueden crear los usuarios en su *home* para poner en él sus páginas personales (pudiera ser otro directorio).

Para comprobar el uso de directorios personales en el servidor virtual por defecto, deberíamos primeramente habilitar el módulo:

```
$ sudo a2enmod userdir
```

A continuación se puede crear el directorio /home/manolo/public_html y colocar en él una página index.html. El acceso a esta página sería:

```
http://localhost/~manolo
```

11.6. Control de acceso por direcciones IP y nombres de dominio.

Anteriormente a la versión 2.4, se podía definir desde qué direcciones IP y/o nombres de dominios se puede acceder a un servidor empleando las directivas Order, Allow y Deny. Éstas se pueden seguir usando si activamos el módulo de compatibilidad mod_access_compat.

Las directivas actuales (Apache 2.4) para el control de acceso son: Require, <RequireAll>, <RequireAny> y <RequireNone>; siendo las tres últimas directivas contenedoras.

El acceso a los recursos por dirección IP o por nombre de dominio del cliente es controlado mediante la declaración de distintas entradas que se pueden aplicar a 3 niveles diferentes: a nivel de directorios del sistema de ficheros, a nivel de la URLs de la petición y a nivel de nombres de archivos solicitados. La especificación de las entradas se expresan de estas tres formas:

```
<Directory "directorio">
        ...
</Directory>

<Location "URL">
        ...
</Location>

<Files "fichero">
        ...
</Files>
```

La entrada <Directory "directorio"> implica que las órdenes especificadas en el control de acceso se aplican al directorio especificado y a sus subdirectorios, excepto que exista una regla más específica para alguno de los subdirectorios. Ejemplos:

```
<Directory "/var/www">
        ...
</Directory>
```

indica que las condiciones se aplicarán al directorio /var/www y sus subdirectorios. La entrada

```
<Files "topami.html">
        ...
</Files>
```

se aplicará a todos los ficheros de nombre topami.html, independientemente del directorio donde se encuentren dichos ficheros.

Todas las entradas anteriores tienen sus equivalentes <DirectoryMatch>, <LocationMatch> y <FilesMatch>, que permiten utilizar expresiones regulares en el nombre, de forma que se pueda hacer referencia a múltiples directorios, URL, o ficheros. Así, la siguientes directivas se aplicarían a cualquier fichero que comience por la cadena ".ht" ubicado en cualquier directorio que sirva Apache.

```
<FilesMatch "^\.ht">
        ...
</FilesMatch>
```

El orden en que se aplican estos tres tipos de entradas tiene importancia en su funcionamiento, por ello es necesario tener en cuenta que en primer lugar se aplican las secciones <Directory>, luego las secciones <DirectoryMatch>, a continuación <Files> y <FilesMatch> sin ninguna prioridad entre ellas, y por último <Location> y <LocationMatch>, también sin ninguna prioridad entre ellas. En el ejemplo:

```
<Location "/">
        Require all granted
</Location>

<Directory "/var/www/html">
        Require all denied
</Directory>
```

podemos ver como la entrada <Directory> restringe el acceso al directorio raíz del servidor virtual por defecto, pero la entrada <Location>permitiría el acceso a dicho directorio. Al evaluarse en último lugar la entrada <Location>, la restricción de la entrada <Directory> queda sin efecto.

Dentro de las entradas especificadas, se indican las directivas que permiten el acceso o la denegación de acceso a cada una de ellas. La directiva que controla este acceso es Require. Esta directiva se usa para controlar el acceso por dirección IP o por nombre de dominio, y puede tomar las formas:

```
Require all granted
Require all denied
Require [not] host {nombre | dominio}
Require [not] ip {ip | subred}
```

Require all granted y Require all denied indican, respectivamente, que todos los clientes tienen permitido el acceso o denegado el acceso.

Require host {nombre | dominio} indica que si el ordenador o dominio del cliente se cumple, el requisito es verdad (o no es verdad si está precedida de not).

Require ip {ip | subred} indica que si la dirección IP del cliente o la red/subred a la que pertenece se cumple, el requisito es verdad (o no es verdad si está precedida de not). Ejemplo:

Require ip 192.168.210.0/24 será verdad si el ordenador pertenece a la red 192.168.210.0/24, mientras que:

Require not host ugr.es será verdad si el nombre de dominio del cliente no pertenece al dominio ugr.es.

La directiva Require puede combinarse mediante directivas contenedoras <RequireAll>...</RequireAll>, <RequireAny>...</RequireAny> y <RequireNone>...</RequireNone> para formar reglas más complejas.

<RequireAll> … </RequireAll> se utiliza para encerrar un grupo de directivas de autorización de las cuales ninguna debe fallar para que la autenticación se cumpla.

En el ejemplo siguiente se exige que los usuarios pertenezcan al grupo *alpha* o *beta* y que no estén incluidos en el grupo *reject*:

```
<RequireAll>
    Require group alpha beta
    Require not group reject
</RequireAll>
```

<RequireAny> … </RequireAny > se utiliza para encerrar un grupo de directivas de autorización de las cuales al menos una debe tener éxito para que la autenticación se cumpla.

Por ejemplo, para poder acceder al sitio web desde el equipo local, o desde los equipos de la red 192.168.210.0/24 escribiríamos en el archivo de configuración:

```
<RequireAny>
    Require local
    Require ip 192.168.210.0/24
</RequireAny>
```

<RequireNone> … </RequireNone> se utiliza para encerrar un grupo de directivas de autorización de las cuales ninguna debe tener éxito para que la autenticación se cumpla.

En el siguiente ejemplo, extraído de la web de Apache, se permite el acceso a todo el mundo excepto si pertenecen a la red 192.168.205.0/24, su host es "*.phishers.example.com", "*.moreidiots.example" o "*.ke":

```
<RequireAll>
    Require all granted
    <RequireNone>
        Require ip 192.168.205.0/24
        Require host phishers.example.com moreidiots.example
        Require host ke
    </RequireNone>
</RequireAll>
```

Cuando se realiza una conexión, el servidor comprueba si el acceso al recurso está autorizado, y si no lo está, devuelve un código de error del tipo *403 Forbbiden*.

11.7. Autentificación y autorización.

La documentación de Apache dice literalmente "La *autentificación* es cualquier proceso mediante el cual se verifica que alguien es quien dice ser. La *autorización* es cualquier proceso por el cual a alguien se le permite estar donde quiere ir, o tener la información que quiere tener."

Las directivas que usaremos necesitan situarse en el archivo de configuración del sitio, dentro de una sección <Directory>, o en archivos de configuración por directorios mediante archivos .htaccess. (Los archivos .htaccess los comentaremos en el próximo apartado).

La principal directiva de autentificación es AuthType. Indica el tipo de autenticación de usuarios. Existen dos valores posibles Basic y Digest.

Basic envía la contraseña entre el cliente y el servidor sin cifrar, por lo que su seguridad depende del canal de comunicación, mientras que Digest envía la contraseña como un compendio MD5, por lo que nunca es posible capturar la contraseña en texto plano, solo su compendio, pero solo es soportado por algunos clientes Web.

Para usar estos tipos de autentificación, lo primero sería comprobar en etc/apache2/mods-enabled, si los módulos auth_basic o auth_digest correspondientes están habilitados. Si no lo estuvieran deberíamos habilitarlos mediante el comando a2enmod.

La directiva AuthName es una cadena de texto que se mostrará en la página de acceso para que el usuario sepa con qué contraseña debe identificarse.

La directiva AuthUserFile indica el nombre del fichero que contiene los nombres de usuarios y sus contraseñas debiendo ser accesible por Apache. Este fichero se crea o modifica utilizando el comando:

```
/usr/sbin/htpasswd
```

La directiva AuthGroupFile indica el nombre del fichero que contiene los nombres de los grupos de usuarios, y los usuarios que conforman ese grupo, consistiendo en un fichero de texto plano. Para cada grupo se deberá indicar una línea con la siguiente sintaxis:

```
<nombre del grupo>: <usuario1> <usuario2> ... <usuarioN>
```

Una vez establecidos los usuarios o grupos de usuarios con las directivas anteriores, se emplea la directiva Require para indicar los nombres de los usuarios, grupos o todos los usuarios a los que se permite el acceso, siempre que proporcionen de forma correcta su contraseña. Las sintaxis de la directiva es:

```
Require user usuario1 [...usuarioN] ; indica qué usuarios concretos podrán autentificarse

Require group grupo1 [...grupoN] ; indica qué grupos a los que deben pertenecer los usuarios podrán autentificarse

Require valid-user ; indica que cualquier usuario que esté en el archivo de contraseñas podrá autentificarse
```

Un ejemplo de fichero de control del acceso a los recursos mediante usuario es el siguiente:

```
AuthType Basic
AuthName "Acceso Restringido"
AuthUserFile /etc/apache2/users ; archivo conteniendo los usuarios válidos junto a sus contraseñas
Require user usuario1 usuario2   ;de los usuarios pertenecientes al archivo de contraseñas, sólo usuario1 y usuario2
```

Mientras que un ejemplo de un fichero que utilice el control de acceso mediante grupos es el siguiente:

```
AuthType Basic
AuthName "Acceso Restringido"
AuthUserFile /var/www/users ; archivo conteniendo los usuarios válidos junto a sus contraseñas
AuthGroupFile /etc/apache2/groups ; archivo con los grupos y los usuarios de cada grupo
Require group grupo1 grupo2 ; solo los usuarios pertenecientes a cualquiera de los 2 grupos y registrados en el
                            ; archivo de contraseñas
```

El comando htpasswd permite crear o modificar el fichero con los usuarios y contraseñas de autenticación de los usuarios. Su sintaxis básica es:

```
htpasswd -c [-p | -d | -m | -s] <fichero> <usuario> ; crea el archivo y le añade los credenciales del usuario
htpasswd [-p | -d | -m | -s] <fichero> <usuario>    ; añade al archivo los credenciales del usuario
htpasswd -D <fichero> <usuario>                     ; elimina del archivo los credenciales del usuario
```

Las diferentes opciones son:

Con el parámetro -c se crea el fichero indicado, pero si ya existiera, el archivo de claves se borra y se crearía de nuevo insertando el usuario indicado.

Las opciones -p, -d, -m y –s indican el modo de cifrado de las contraseñas en el fichero. El modo por defecto es -d, que es un cifrado usando crypt(), el algoritmo estándar de UNIX, mientras que la opción -p indica un cifrado en texto plano, esto es, sin cifrar, y las opciones -m y -s indican cifrado utilizando MD5 y SHA respectivamente.

Ejemplo: Vamos a configurar la autentificación HTTP *Basic* sobre el directorio /var/www/html/basic para que sólo puedan acceder los usuarios *alu1* y *alu2* mediante credenciales.

Primero creamos el archivo de usuarios y contraseñas (lo llamaremos passwd) y añadimos el usuario *alu1*. (La opción -c es para crear el archivo).

```
$ sudo htpasswd -c /etc/apache2/passwd alu1
```

Y luego le añadimos el usuario *alu2*

```
$ sudo htpasswd /etc/apache2/passwd alu2
```

Editamos el fichero de configuración 000-default.conf para permitir el acceso privado al directorio /var/www/html/basic a los usuarios creados en el paso anterior. Solo queda reiniciar el servidor.

```
<Directory /var/www/html/basic>
        Options Indexes FollowSymLinks
        AuthType Basic
        AuthName "Acceso privado"
        AuthUserFile "/etc/apache2/passwd"
        Require user alu1 alu2
        # Sólo los usuarios alu1 alu2 podrán autentificarse. Otros posibles usuarios dados de alta en el archivo
        # /etc/apache2/passwd no podrían acceder.
        # Para que todos los usuarios dados de alta en /etc/apache2/passwd pudieran acceder se debe usar la
        # directiva Require valid-user
</Directory>
```

11.8. Los archivos .htaccess.

Los archivos .htaccess son archivos de configuración de directorios que permiten realizar cambios en la configuración del servidor, afectando al directorio donde esté colocado (con sus respectivos subdirectorios), sin necesidad de que el administrador tenga que editar el archivo de configuración de Apache, ni reiniciar el servicio.

Por defecto, en el archivo /etc/apache2/apache2.conf, donde se establece la configuración general de Apache, se indica mediante la directiva AccessFileName que el nombre del archivo donde se puede escribir estas directivas es .htaccess aunque pudiera ser otro cualquiera, como por ejemplo .htconfiguracion.

```
AccessFileName .htconfiguracion
```

Cuando un cliente realiza una petición de un recurso, el servidor busca en la ruta del recurso solicitado el archivo .htaccess y si existe, aplica las directivas contenidas en él. Como el archivo .htaccess se lee en cada petición, cualquier cambio o modificación se aplica de forma inmediata.

¿Qué cambios de configuración para un directorio se pueden realizar mediante el archivo .htaccess?. Pues va a depender del valor que tenga la directiva AllowOverride incluida en la sección <Directory> de configuración de dicho directorio. Así, cuando el servidor encuentra un fichero .htaccess necesita saber qué directivas presentes en ese fichero pueden prevalecer sobre las directivas de configuración previas. Por ejemplo, cuando el valor de esta directiva es AllowOverride None, entonces los ficheros .htaccess son ignorados completamente. En ese caso, el servidor ni siquiera intentará leer los archivos .htaccess existentes.

Los valores que puede tomar la directiva AllowOverride son:

Valor	Descripción (qué directivas se pueden modificar)
All	Permite todas las directivas.
None	No permite ninguna directiva.
AuthConfig	Permite directivas de autentificación de usuarios.
FileInfo	Permite directivas de control del tipo de documentos.
Indexes	Permite directivas de indexado de directorios.
Limit	Permite directivas que controlan el acceso por dirección IP o por nombre de hosts del cliente.
Options	Permite directivas que controlan funcionalidades de los directorios.

Por ejemplo si establecemos AllowOverride All podremos usar cualquier directiva dentro de los archivos .htaccess.

Hay que tener muy en cuenta que las directivas son aplicadas en el orden en que son encontradas por Apache. De esta forma, un archivo .htaccess de un directorio puede sobrescribir las directivas encontradas en un archivo situado en un nivel superior, y así sucesivamente. Apache busca directivas en cualquier archivo .htaccess del directorio particular que ha sido llamado y en los que están más arriba, para determinar las directivas que utilizará cuando procese la petición de llamada de la página de su sitio.

Todo esto es muy útil sobre todo para configurar sitios que estén hospedados en proveedores de sitios web. La mayoría de los *hostings* permiten directivas .htaccess, aunque depende de cada proveedor.

El uso de estos archivos trae consigo una reducción del rendimiento del servidor, ya que Apache buscará en cada uno de los directorios que forman parte de la web la presencia de archivos .htaccess para determinar las directivas que se utilizan en él.

Son muchas las posibilidades que ofrecen este tipo de archivos, pero habitualmente se usan para especificar restricciones de seguridad para un determinado directorio, reescribir URLs largas y complejas en otras más sencilla o bloquear a usuarios por su dirección IP, tal y como vimos anteriormente.

En el archivo de configuración general de Apache hay una directiva para que los archivos que comiencen con .ht no sean accesibles por los clientes. Por eso se nombran comenzando por .ht.

```
<Files ~ "^\.ht">
        Require all denied
</Files>
```

Ejemplo. Supongamos que Apache está configurado para que en el directorio /home/manolo/wiki se ubiquen los archivos a servir cuando accedamos mediante http://localhost/wiki mediante un alias. Manolo quiere tener cierto control sobre el acceso a los contenidos de su carpeta /home/manolo/wiki; en concreto quiere que solo pueda acceder el usuario *wiki* mediante autentificación *Basic* desde la red 192.168.210.0/24.

Esta petición se la realiza al administrador de Apache, el cual como administrador edita /etc/apache2/sites-available/000-default.conf, creando el alias y mediante AllowOverride All indica que se permitirá sobrescribir todas las directivas (All) en el directorio /home/manolo/wiki; (eso es lo que quiere Manolo). El administrador reinicia el servidor para que los cambios tengan efecto.

```
Alias /wiki /home/manolo/wiki
<Directory /home/manolo/wiki>
        AllowOverride All
</Directory>
```

Ahora Manolo inicia sesión como usuario *manolo* en la máquina del servidor Apache, crea el directorio /home/manolo/wiki, un archivo para servir por defecto llamado indexwiki.html, y un archivo de claves /home/manolo/wiki/.htclaves al cual le añade el usuario *wiki*. No tiene problemas de crear archivos ya que lo hace en su carpeta personal.

```
$ htpasswd  -c /home/manolo/wiki/.htclaves wiki
```

Por último Manolo crea el archivo /home/manolo/wiki/.htaccess con las directivas necesarias para la configuración que desea para el directorio /home/manolo/wiki.

```
DirectoryIndex indexwiki.html
Require ip 192.168.210.0/24
AuthType Basic
AuthName "Acceso privado a la wiki"
AuthUserFile "/home/manolo/wiki/.htclaves"
Require user wiki
```

Sin necesidad de reiniciar Apache, Manolo ha conseguido su propósito con esta nueva configuración.

11.9. Configuración de servidores virtuales.

Apache permite servir varios sitios web mediante un único software servidor, mediante la creación de diferentes dominios virtuales distinguibles por su IP, nombre de dominio o puerto. Esta capacidad simplifica enormemente la administración de diferentes sitios web.

Los servidores virtuales por nombre son los más empleados ya que por una misma IP y puerto se puede tener varios sitios web distintos. Cuando Apache recibe la petición, la cabecera host de la petición contiene el nombre de dominio del servidor solicitado, lo que le permite conocer el sitio a servir.

Aunque no es necesario, es conveniente por simplicidad en la configuración de los servidores virtuales, usar archivos de configuración independientes para cada uno de ellos. Por ejemplo, si queremos configurar dos servidores virtuales por nombre asir.aula.izv y bd.aula.izv crearíamos dos archivos de configuración (por ejemplo asir-web.conf y bd-web.conf) en el directorio /etc/apache2/sites-available.

Para habilitar un servidor virtual es necesario crear un enlace en /etc/apache2/sites-enabled al archivo de configuración creado previamente en /etc/apache2/sites-available/. Esta operación se realiza mediante la orden a2ensite. Por ejemplo, las órdenes:

```
/etc/apache2/sites-available$ sudo a2ensite asir-web.conf
/etc/apache2/sites-available$ sudo a2ensite bd-web.conf
```

habilitaría los servidores cuyos archivos de configuración se denominan asir-web.conf y bd-web.conf.

Los servidores virtuales se pueden deshabilitar rompiendo el enlace existente mediante la orden

```
a2dissite archivo_configuracion
```

11.9.1. Servidores virtuales por nombre.

Para atender a varios servidores virtuales por nombre utilizando una misma dirección IP y puerto para todos ellos, se debe emplear la directiva contenedora <VirtualHost IP:puerto> para cada uno de ellos y colocar dentro de ésta las directivas que indiquen como se define cada uno de los servidores virtuales que estarán disponibles por dicha IP y puerto.

En principio si desde un navegador usamos la IP, (o el nombre del dominio del equipo si un servidor DNS lo resuelve), Apache no sabría que sitio web servir si dispone de varios sitios definidos en dicha IP y puerto. Lo soluciona sirviendo el sitio cuyo archivo de configuración es el primero en una ordenación alfabética. Por eso sirve el sitio web por defecto ya que el archivo de configuración de dicho sitio se denomina 000-default.conf.

La única manera de romper esta ambigüedad es indicando en la configuración de cada uno de los sitios, un nombre de dominio distinto para cada uno de ellos. Para ello se usa la directiva ServerName nombre_dominio. Ahora, cuando Apache reciba la petición de un sitio, en la cabecera host de la petición recibirá dicho nombre, y por lo tanto podrá distinguir un sitio de otro y servir el correspondiente.

Por ejemplo, si queremos configurar dos servidores virtuales por nombre accediendo a cada uno de ellos mediante los nombres de dominio asir.aula.izv y bd.aula.izv, y los dos definidos sobre la misma IP y puerto, crearíamos los dos archivos de configuración siguientes:

```
# Archivo /etc/apache2/sites-available/asir-web.conf

<VirtualHost *:80>
        ServerName asir.aula.izv
        DocumentRoot /var/www/asir

        <Directory /var/www/asir>
                DirectoryIndex index.html
                Options Indexes FollowSymLinks
                AllowOverride None
                Require all granted
        </Directory>

        ErrorLog ${APACHE_LOG_DIR}/asir.error.log
        LogLevel warn
        CustomLog ${APACHE_LOG_DIR}/asir.access.log combined
</VirtualHost>

# Archivo /etc/apache2/sites-available/bd-web.conf

<VirtualHost *:80>
        ServerName bd.aula.izv
        DocumentRoot /var/www/bd

        <Directory /var/www/bd>
                DirectoryIndex index.html
                Options Indexes FollowSymLinks
                AllowOverride None
                Require all granted
        </Directory>
```

```
    ErrorLog ${APACHE_LOG_DIR}/bd.error.log
    LogLevel warn
    CustomLog ${APACHE_LOG_DIR}/bd.access.log combined
</VirtualHost>
```

El ejemplo define dos servidores web por cualquier IP y puerto 80, pero cada uno de ellos con un nombre de dominio diferente establecido mediante la directiva ServerName y con su propio directorio raíz indicado en DocumentRoot. Además cada uno de ellos dispone sus archivos de *logs* de errores y accesos.

IMPORTANTE. Por supuesto siempre debe haber un servidor DNS o una tabla *hosts* que asocie los nombres de dominio asir.aula.izv y bd.aula.izv a la misma dirección IP donde escucha el servidor.

En nuestro ejemplo, el servidor virtual por defecto seguirá respondiendo cuando usemos en la URL de la petición directamente la dirección IP, o bien un nombre de dominio que se resuelva con nuestra IP, pero siempre que dicho nombre no se corresponda con ningún nombre de dominio previamente usado en la directiva ServerName de nuestros servidores virtuales.

Es muy común usar la directiva ServerAlias *hostname* [*hostname*] ... para indicar otros nombres que se pueden usar para acceder al sitio. Se pueden usar * y ? en *hostname* para establecer patrones de nombres.

11.9.2. Servidores virtuales por dirección IP.

Desde que los servidores web implementaron la posibilidad de usar servidores virtuales por nombre, los servidores virtuales por dirección IP dejaron prácticamente de usarse. Principalmente por la escasez de direcciones IP y la complejidad de usar y mantener varias interfaces de red en el servidor.

Si no tenemos como mínimo dos tarjetas de red instaladas en la máquina, cada una con su IP, deberemos usar la única tarjeta disponible configurándola mediante *IP aliasing*. Esta técnica permite asignar más de una IP a una misma interfaz de red.

Para añadir la segunda interfaz de red 192.168.210.222/24, podemos hacerlo usando los comandos:

```
$ sudo ip addr add 192.168.210.222/24 dev enp0s3
$ sudo ipdown enp0s3
$ sudo ipup enp0s3
```

Esta configuración permanecerá válida hasta el siguiente reinicio del equipo. Para que sea permanente deberemos activarla en los archivos de configuración del servicio de red.

Si el servicio de red se configura mediante el archivo /etc/network/interfaces, lo editaremos y le añadimos la segunda IP a la interfaz:

```
# the loopback network interface
auto lo
iface lo inet loopback

# the primary network interface
auto enp0s3
iface enp0s3 inet static
address 192.168.210.212
netmask 255.255.255.0
gateway 192.168.210.254
broadcast 192.168.210.255
address 192.168.210.222
netmask 255.255.255.0
```

Luego deberemos reiniciar el servicio de red:

```
$ sudo  systemctl restart networking
```

Si el servicio de red se configura mediante la aplicación *NetworkManager*, sólamente deberemos editar la conexión y añadir la nueva IP en la pestaña de *Ajustes de IPv4*.

Para definir servidores virtuales sólo por IP, en la definición de cada sitio web tendríamos que definir los sitios web de forma que cada uno de ellos usara una IP distinta. Por ejemplo:

```
<VirtualHost IP1:80>
    ...
</VirtualHost>
<VirtualHost IP2:80>
    ...
</VirtualHost>
```

```
<VirtualHost IP3:80>
    ...
</VirtualHost>
```

Ahora Apache servirá el sitio correspondiente según sea la IP indicada en la petición. Ahora bien, los usuarios en las peticiones de un sitio web no usamos la IP del sitio, sino un nombre de dominio. Por lo tanto si queremos asociar a cada IP un nombre de dominio distinto, deberemos en el servidor DNS asociar dichos nombres de dominios con las IP donde se sirven los sitios web.

También cabe la posibilidad de combinar el uso de servidores virtuales por IP y por nombre. Si por cada IP, en vez de definir como en el ejemplo anterior un solo sitio web, queremos tener varios sitios web por esa misma IP, en las configuración de cada sitio, se incluye la directiva ServerName para que Apache sepa distinguir cual sitio a servir cuando reciba la petición por una IP concreta. Por ejemplo:

```
<VirtualHost IP1:80>
    ServerName s1.dominio
    ...
</VirtualHost>
<VirtualHost IP1:80>
    ServerName s2.dominio
    ...
</VirtualHost>
<VirtualHost IP2:80>
    ServerName s3.dominio
    ...
</VirtualHost>
<VirtualHost IP2:80>
    ServerName s4.dominio
    ...
</VirtualHost>
```

Además, para poder usar en las peticiones los nombres de dominio indicados en las directivas ServerName, un servidor DNS tendrá que asociar los nombres de dominio s1.dominio y s2.dominio a la IP1, y los nombres de dominio s3.dominio y s4.dominio a la IP2.

11.9.3. Servidores virtuales por puerto.

Para definir servidores virtuales sólo por puerto, tendríamos que indicar mediante la directiva Listen los puertos de escucha, y en la definición de cada sitio web, tendríamos que definirlos de forma que cada uno de ellos usara un puerto distinto. Por ejemplo:

```
Listen 80
Listen 81
Listen 82
<VirtualHost IP:80>
    ...
</VirtualHost>
<VirtualHost IP:81>
    ...
</VirtualHost>
<VirtualHost IP:82>
    ...
</VirtualHost>
```

Ahora Apache servirá el sitio correspondiente según sea el puerto indicado en la URL de la petición. Si no se indica un número de puerto en la URL de petición, se supone el puerto 80. Por ejemplo:

```
http://IP:81
```

Como siempre, si queremos usar en la petición nombres de dominio en vez de la IP, un servidor DNS tiene que proporcionar el mecanismo de resolución del nombre del dominio.

11.9.4. Combinando diferentes tipos de servidores virtuales

La técnica de usar servidores virtuales por diferentes criterios se puede combinar. Por ejemplo, el mecanismo de definición de servidores virtuales por puerto se puede combinar con el mecanismo de definición de servidores virtuales por IP, y también con el mecanismo de definición de servidores virtuales por nombre; incluso combinado las tres técnicas. Supongamos la siguiente configuración:

```
Listen 192.168.1.201:80
Listen 192.168.1.201:81
Listen 192.168.1.202:81
Listen 192.168.1.202:82
# Se definien varios servidores virtuales por nombre usando el socket 192.168.1.201:80. (s1.dominio.izv
#  y s2.dominio.izv)
<VirtualHost 192.168.1.201:80>
        ServerName s1.dominio.izv
        ...
</VirtualHost>

<VirtualHost 192.168.1.201:80>
        ServerName s2.dominio.izv
        ...
</VirtualHost>

# Se definien varios servidores virtuales por nombre usando el socket 192.168.1.201:81. (s3.dominio.izv
# y s4.dominio.izv)
<VirtualHost 192.168.1.201:81>
        ServerName s3.dominio.izv
        ...
</VirtualHost>

<VirtualHost 192.168.1.201:81>
        ServerName s4.dominio.izv
        ...
</VirtualHost>

# Se definien varios servidores virtuales por puerto usando los sockets 192.168.1.202:81 y  192.168.1.202:82.
<VirtualHost 192.168.1.202:81>
        ...
</VirtualHost>

<VirtualHost 192.168.1.202:82>
        ...
</VirtualHost>
```

Supongamos que disponemos de un servidor DNS autorizado para la zona dominio.izv y que la máquina donde corre el servidor Apache dispone de dos interfaces de red: 192.168.1.201 e 192.168.1.202. Con este archivo de zona:

```
dominio.izv.        IN      NS      dns.dominio.izv.
dns.dominio.izv.    IN      A       IP del servidor DNS
. . .
s1.dominio.izv.     IN      A       192.168.1.201
s2.dominio.izv.     IN      CNAME   s1.dominio.izv.
s3.dominio.izv.     IN      CNAME   s1.dominio.izv.
s4.dominio.izv.     IN      CNAME   s1.dominio.izv.
s5.dominio.izv.     IN      A       192.168.1.202
s6.dominio.izv.     IN      CNAME   s5.dominio.izv.
```

Para solicitar cada uno de los sitios web definidos anteriormente, las peticiones deberían ser:

http://s1.dominio.izv
http://s2.dominio.izv
http://s3.dominio.izv:81
http://s4.dominio.izv:81
http://s5.dominio.izv:81
http://s6.dominio.izv:82 o bien http://s5.dominio.izv:82

Recordar que cualquier petición realizada al servidor usando directamente la IP o un nombre de dominio que no coincida con ninguno de los especificados con la directiva ServerName, hará que Apache sirva el servidor que considere con más prioridad en su definición. Así por ejemplo, si estuviera habilitado el servidor virtual por defecto, éste sería el que serviría.

11.10. Configuración de un servidor virtual seguro (https).

Si necesitamos configurar un servidor web seguro en Linux, tendremos que definir en el servidor las carpetas a servir que mediante el protocolo SSL/TLS proporcionen el cifrado de los datos que se intercambian entre el servidor web y el cliente.

Como ya comentamos, debemos disponer de un certificado de seguridad que puede ser expedido por una AC con el consiguiente coste económico, o bien crear y utilizar nuestra propia AC que expedirá certificados válidos en nuestro ámbito de actuación.

El paquete *OpenSSL* para Linux se encarga de la implementación del protocolo SSL para llevar a cabo transferencias seguras y de la creación de certificados. La creación de certificados autofirmados con OpenSSL es análoga a como lo hicimos para Windows, pero normalmente en Ubuntu viene instalado el paquete ssl-cert que permite mediante el comando make-ssl-cert generar certificados de una forma más simple que usando las utilidades de OpenSSL.

El módulo de Apache que da soporte para SSL/TLS es ssl. En el archivo de configuración de dicho módulo ssl.conf, viene por defecto habilitados todos los protocolos. Ya no existe soporte para SSLv2, pero por seguridad deberíamos también deshabilitar el soporte para SSLv3 modificando la línea:

```
SSLProtocol all -SSLv3
```

Debido al funcionamiento del protocolo SSL, hasta la versión 2.2.12 de Apache no era posible definir servidores virtuales por nombre que utilizaran SSL. La razón es que cuando SSL establece la negociación, no conoce el nombre del servidor virtual con el que establecer la conexión, por lo que negociará siempre con el primer servidor virtual las claves, certificados, etc. quedando el resto de servidores virtuales sin posibilidad de acceso.

A partir de la versión 2.2.12 de Apache, se dispone para SSL de la extensión SNI (*Server Name Indication*), la cual permite, antes de negociar con el servidor la conexión SSL, el envío del nombre del servidor a conectar. De esta forma el servidor conoce previamente el nombre del sitio con el que se va a desear establecer la comunicación y, por tanto, enviarle el certificado correspondiente.

El uso de SNI viene desactivado por defecto en el archivo ssl.conf ya que la directiva

```
SSLStrictSNIVHostCheck On
```

está comentada, por lo que le quitaremos el comentario para que podamos usar servidores virtuales por nombre seguros. Hay que tener en cuenta que sólo las últimas versiones de los navegadores permiten el uso de la extensión SNI de SSL.

11.10.1. Servidor virtual HTTPS por defecto.

Apache dispone de un servidor virtual seguro por defecto. Para usarlo deberemos realizar una serie de tareas.

- Añadiremos el módulo que da soporte para SSL:
  ```
  $ sudo a2enmod ssl
  ```
- Consultamos el archivo /etc/apache2/ports.conf y comprobamos que si se habilita el módulo ssl, el servidor escuchará en el puerto 443. Podemos verificarlo con netstat –ltn.
- Comprobamos que en el directorio /etc/apache2/sites-availables existe el archivo default-ssl.conf que contiene la configuración por defecto de un servidor HTTPS.
- Habilitamos el servidor virtual ssl por defecto de Apache:
  ```
  $ sudo a2ensite default-ssl.conf
  ```
- Reiniciamos el servidor para que tengan efecto los cambios.
- Si editamos el archivo /etc/apache2/sites-availables/default-ssl.conf podremos observar en su configuración cuáles son las directivas que habilitan SSL e indican el nombre del archivo que contiene el certificado digital del servidor.
  - SSLEngine. Activa/desactiva el protocolo SSL/TLS en un servidor virtual. (Por defecto está deshabilitado tanto para el servidor principal como para todos los servidores virtuales.)
  - SSLCertificateFile. Indica el nombre del archivo que contiene el certificado del servidor
  - SSLCertificateKeyFile. Indica el nombre del archivo que contiene la clave privada del servidor.

Como podemos comprobar, el servidor virtual seguro por defecto usa un certificado autofirmado y una clave privada que se crearon cuando se realizó la instalación de Apache.

- Para probar el servidor debemos usar el protocolo HTTPS en la URL del navegador de la forma https://localhost o un nombre de dominio si lo resuelve un servidor DNS.

  El navegador nos presentará un mensaje de aviso diciendo que la conexión no es segura ya que la identidad del servidor no puede ser verificada. Esto ya la sabemos, porque el certificado no está emitido por una AC de confianza para el navegador (no tiene instalado su certificado raíz). Tendremos que añadir una excepción y confirmarla para poder acceder al sitio web.

11.10.2. Servidores virtuales HTTPS.

Vamos a crear un servidor virtual seguro con nombre de dominio webseg.aula.izv. Necesitaremos definir el servidor virtual y generar un certificado digital. En nuestro caso generaremos un certificado autofirmado con la utilidad make-ssl-cert que proporciona *OpenSSL*.

Los pasos para crear un servidor virtual seguro son:

- Si queremos disponer de varios servidores virtuales seguros por nombre, deberemos comprobar que la extensión SNI de SSL está activada. Además deberemos comprobar en el archivo /etc/apache2/ports.conf que el puerto 443 está a la escucha.
- Crearemos un certificado autofirmado que guardaremos por ejemplo en /etc/apache2/ssl (crear antes el subdirectorio ssl …).Como ya comentamos podemos generar un certificado autofirmado ejecutando el comando:

  ```
  $ make-ssl-cert /usr/share/ssl-cert/ssleay.cnf /etc/apache2/ssl/aula.pem
  ```

  Durante la ejecución del comando *make-ssl-cert*, nos preguntará por el nombre del servidor a certificar, el país donde se encuentra, la organización a la que pertenece, etc. creándose el archivo /etc/apache2/ssl/aula.pem conteniendo el certificado de servidor.

  Para ver el contenido del certificado podemos usar el comando

  ```
  $ sudo openssl x509 -in /etc/apache2/ssl/aula.pem -noout -text
  ```

  Informándonos de quién certifica, la validez del certificado, qué se certifica, y la firma del certificado.
- Creamos el directorio raíz del servidor seguro (por ejemplo /var/www/websegura) y dentro un archivo index.html con el contenido de la página principal.
- Creamos en /etc/apache2/sites-available el archivo de configuración del servidor, al cual le llamaremos websegura.conf, con el contenido siguiente:

```
# Archivo /etc/apache2/sites-available/websegura
<IfModule mod_ssl.c>
    <VirtualHost *:443>
        ServerName webseg.aula.izv
        DocumentRoot /var/www/websegura

        <Directory /var/www/websegura>
            SSLRequireSSL
            DirectoryIndex index.html
            Options Indexes FollowSymLinks
            AllowOverride None
            Require all granted
        </Directory>

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

        SSLEngine On
        SSLCertificateFile /etc/apache2/ssl/aula.pem
    </VirtualHost>
</IfModule>
```

El significado de la directiva SSLRequireSSL es prohibir el acceso al directorio salvo que se use el protocolo HTTPS.

- Habilitamos el servidor web seguro y reiniciamos el servicio

```
$ sudo a2ensite websegura.conf
$ sudo systemctl restart apache2
```

- Para poder resolver el nombre de dominio del servidor, se necesitará registrar dicho nombre de dominio en el servidor DNS y reiniciar posteriormente el servicio DNS.
- Para probar el servidor debemos usar el protocolo HTTPS en la URL del navegador de la forma https://webseg.aula.izv.

  Ya sabemos que el navegador nos presentará un mensaje de aviso diciendo que la conexión no es segura ya que la identidad del servidor no puede ser verificada.

Anexo 1. Certificados autofirmados mediante OpenSSL

En el caso de que usemos paso a paso las órdenes que proporciona *OpenSSL* para crear nuestro certificado autofirmado, haríamos el mismo procedimiento que vimos cuando lo hicimos en Windows Server, excepto el último paso en que teníamos que convertir el certificado obtenido a formato *pfx*.

1. Generación de una clave privada.

Para firmar digitalmente necesitamos generar una clave privada. Crearemos una de tipo RSA de 1024 bits y guardada en formato PEM (legible). En este caso, diferencia a como hicimos con Windows Server, no la cifraremos con la opción -des. De esta forma cada vez que la usemos no se nos pedirá una contraseña de protección. (Así evitamos proporcionársela a Apache cada vez que reiniciemos el servidor cuando use un certificado)

```
alumno@ubu:/etc/apache2/ssl$ openssl genrsa  -out servidor.key 1024
```

2. Generar un CSR (Solicitud de firma de certificado).

Para realizar la petición de un certificado hay que generar un archivo con información referente a quien realiza la petición del certificado y firmado con la clave privada del solicitante.

Para ello ejecutamos:

```
alumno@ubu:/etc/apache2/ssl$ openssl req -new -key servidor.key  -out servidor.csr
```

En este comando nos pedirá los datos que queremos que aparezcan en el certificado. Entre los datos, lo más importante es indicar correctamente el nombre del servidor web para el cual generaremos el certificado (Common Name).

3. Crear un certificado autofirmado.

La petición de certificado generada anteriormente la firmaremos nosotros mismos (por eso es autofirmado…), usando para firmar nuestra clave privada. El certificado generado lo guardaremos en el archivo certificado.crt. Para ello ejecutamos el comando:

```
alumno@ubu:/etc/apache2/ssl$ openssl x509 -req -days 365 -in servidor.csr -signkey servidor.key
  -out certificado.crt
```

Sólo queda indicar en la configuración del sitio web seguro, cuál es nuestro archivo con el certificado del servidor, y cuál el archivo que contiene la clave privada del servidor. Esto último es necesario porque este certificado no incluye la clave privada, a diferencia de los certificados generados con make-ssl-cer que si la incluyen.

```
…
SSLEngine On
SSLCertificateFile /etc/apache2/ssl/certificado.crt
SSLCertificateKeyFile /etc/apache2/ssl/servidor.key
…
```

Anexo 2. Control de acceso por IP y nombres de dominio en Apache 2.2

Se puede definir desde qué direcciones IP y/o nombres de dominios se puede acceder a un servidor web a través de las directivas Order, Allow y Deny.

Cuando se realiza una conexión, el servidor comprueba si el acceso al recurso está autorizado, y si no lo está, devuelve un código de error del tipo *403 Forbbiden*.

Allow permite especificar quién tendrá autorización para acceder a un recurso. Se pueden especificar direcciones IP, máscaras de red, nombres de dominio, fragmentos de nombre de dominio. También existe la palabra clave "all" que indica todos los clientes.

Deny indica a quién no le permitiremos el acceso a un recurso y usa las mismas opciones que Allow.

Order permite afinar el funcionamiento de las anteriores directivas. Existen 2 opciones:

- Order Allow,Deny. Por defecto se deniega el acceso y sólo podrán acceder aquellos clientes que cumplan las especificaciones de Allow y no cumplan las especificaciones de Deny.
- Order Deny,Allow. Por defecto se permite el acceso y sólo podrán acceder los clientes que no cumplan las especificaciones de deny o sí cumplan las especificaciones de Allow.

Ejemplo 1: Para poder acceder al sitio web sólo desde el equipo local y desde el equipo 192.168.109.212, escribiríamos en el archivo de configuración:

```
Order Allow,Deny
Allow from 127.0.0.1 localhost
Allow from 192.168.210.212
```

Ejemplo 2: Para permitir el acceso a todos los equipos pertenecientes al dominio midominio.com y denegar el acceso al resto de equipos:

```
Order Deny,Allow
Deny from all
Allow from midomio.com
```

Ejemplo 3: Para poder acceder sólo desde los equipos de determinadas redes:

```
Order Allow,Deny
Allow from 192.168.210.0/24
Allow from 192.168.209.0/24
```

Anexo 3. HTTP/2

La velocidad de carga de una web en el navegador es muy importante, tanto que Google lo tiene en cuenta a la hora del posicionamiento en su buscador.

En las conexiones de red un factor clave es la latencia, el tiempo que tarda en transmitirse un paquete dentro de la red. En las conexiones a Internet, dependiendo de la latencia de nuestra conexión, el tiempo de carga de una web será mayor o menor.

Los factores que influyen en la latencia de una conexión son: la tecnología de acceso a la red (ADSL, fibra óptica, etc.), la distancia al servidor, la carga del servidor así como la capacidad de proceso del dispositivo usado en la conexión. Estos factores en la mayoría de las ocasiones no podemos cambiarlos, así que un modo generalizado que nos permita disminuir los tiempos de transmisión es mejorar el protocolo de comunicación.

Desde hace quince años el protocolo usado para la transferencia desde el servidor al navegador del HTML, CSS, JavaScript, imágenes y demás recursos de la página ha sido HTTP. Con el objetivo de disminuir la latencia Google diseñó un protocolo complementario al HTTP denominado SPDY, el cual sirvió de base para que en mayo de 2015 se presentará el nuevo protocolo HTTP/2.

HTTP/2 no modifica la semántica del protocolo HTTP, de forma que los mensajes HTTP, los códigos de estado, URIs y cabeceras siguen siendo los mismos. Algunas de las mejoras que ofrece HTTP/2 son:

- Multiplexación y concurrencia. Ahora solo es necesaria una conexión TCP para enviar todos los recursos de la página, además la transmisión de varios recursos es posible que sea simultánea sin necesidad de que termine una para iniciar otra.
- Compresión de cabeceras. Para reducir la sobrecarga de las cabeceras en cada petición se puede emplear distintos mecanismos. Uno de ellos es usar un algoritmo de compresión y otro es evitar reenviar las cabeceras que anteriormente han sido transmitidas.
- Envíos desde el servidor de una forma proactiva, en el sentido que se pueden enviar recursos similares al cliente para que sean cacheados por él, aunque el cliente no los haya solicitado de forma explícita.

Una diferencia que lo hace incompatible con HTTP es que HTTP/2 es binario, No obstante se pueden usar ambos de forma conjunta, ya que cuando se establece la conexión, se negocia el protocolo a elegir en función de cuál es el mejor según las características de ambas partes. Los protocolos binarios son más eficientes de procesar, más compactos y menos propensos a errores comparados con los basados en texto.

Es muy importante tener en cuenta que aunque para usar HTTP/2 no es necesario usar cifrado TLS, actualmente los navegadores Firefox, Microsoft Edge y Google Chrome solo lo soportan si se usa con cifrado TLS.

A la hora de implementar HTTP/2 en el servidor Apache, sólo podremos hacerlo a partir de la versión 2.4.17 y posteriores. Para conocer la versión instalada de Apache, podemos usar el comando:

```
alumno@ubu:~$ sudo apachectl -v
```

En Apache, el protocolo HTTP/2 se implementa activando el módulo mod_http2. Este módulo incluye soporte HTTP/2 sobre texto plano (http:), así como sobre conexiones seguras (https:). La variante de texto plano se denomina 'h2c', y la segura 'h2'. Para activar este módulo haremos:

```
alumno@ubu:~$ sudo a2enmod http2
```

Para añadir soporte HTTP/2 para un sitio web concreto, deberemos indicarlo en la configuración de dicho sitio web usando la directiva Protocols. Así por ejemplo:

```
Protocols h2 h2c http/1.1;
```

Habilitaría tanto las dos variantes de HTTP/2, como HTTP en donde el orden de preferencia queda indicado de izquierda (mayor prioridad) a derecha (menor prioridad). En este caso como el sitio web ofrece HTTP/2 en el modo seguro (h2), dicho sitio debe estar configurado previamente como un sitio web seguro.

Recordar que algunos navegadores como Firefox o Chrome únicamente soportan HTTP/2 sobre TLS. Así que sólo veremos actuar HTTP/2 al entrar en sitios con https:// y que ofrezcan soporte HTTP/2.

No existe ningún elemento en la interfaz de los navegadores que nos diga que se está usando HTTP/2. Un modo de averiguarlo en Firefox es activar "Herramientas"-> “Desarrollador Web"->"Red” y seleccionar una petición para comprobar cuales son las cabeceras de respuesta que está enviando el servidor. La respuesta será entonces “HTTP/2.0”. Firefox inserta su propia cabecera denominada “X-Firefox-Spdy:” para indicarlo.

Las cabeceras que se muestran en las herramientas de red de los navegadores al utilizar HTTP/2, han sido convertidas desde el formato binario para parecerse al estilo clásico de las cabeceras de HTTP.

Para Firefox y en Chrome existe un *plugin* denominado HTTP/2 Indicator que muestra un icono en la barra de direcciones con forma de rayo en el caso de que se use HTTP/2 en la conexión.