

Práctica 3.11: Servidor DNS <i>BIND</i> en <i>Linux</i> . Configuración del servidor como secundario (esclavo) para una zona de resolución directa y una zona de resolución inversa . . . . .	144	Práctica 4.13: Configuración de un servidor FTP para que acepte conexiones en modo pasivo detrás de un Cortafuegos/NATP . . . . .	184
Práctica 3.12: Servidor DNS en <i>Zentyal</i> . Configuración del servidor primario para una zona de resolución directa y una zona de resolución inversa . . . . .	144	Práctica 4.14: Servidor SSH/SFTP . . . . .	185
Práctica 3.13: Servidor DNS de <i>Microsoft</i> en <i>Windows 2012</i> . Creación de subdominios . . . . .	145	4.7. Resumen . . . . .	186
Práctica 3.14: Servidor DNS de <i>Microsoft</i> en <i>Windows 2012Server</i> . Delegación de subdominios . . . . .	145	4.8. Test de repaso . . . . .	187
Práctica 3.15: Servidor DNS <i>BIND</i> en <i>Linux</i> . Creación de subdominios . . . . .	145	4.9. Comprueba tu aprendizaje . . . . .	188
Práctica 3.16: Servidor DNS <i>BIND</i> en <i>Linux</i> . Delegación de subdominios . . . . .	146	<b>5. Servidores Web (HTTP)</b> . . . . .	189
Práctica 3.17: Servidor DHCP y DNS en <i>Zentyal</i> . Actualizaciones DNS Dinámicas . . . . .	146	5.1. Introducción . . . . .	190
Práctica 3.18: Servidor DNS <i>BIND</i> en <i>Linux</i> . Opciones de seguridad básicas . . . . .	146	5.2. WWW . . . . .	190
Práctica 3.19: Servidor <i>Dnsmasq</i> en <i>Linux</i> . . . . .	146	5.3. W3C y estándares Web . . . . .	191
Práctica 3.20: Herramientas de consulta a servidores DNS. Consultas recursivas e iterativas . . . . .	146	5.4. Componentes y funcionamiento . . . . .	191
3.20. Resumen . . . . .	148	5.5. Nombres y direcciones (URIs y URLs) . . . . .	192
3.21. Test de repaso . . . . .	149	5.6. Páginas web, sitios web y aplicaciones web . . . . .	192
3.22. Comprueba tu aprendizaje . . . . .	150	5.7. Servidores web . . . . .	193
<b>4. Servicios de transferencia de ficheros</b> . . . . .	151	5.8. Clientes web (navegadores) . . . . .	194
4.1. Introducción . . . . .	152	5.9. Proxies web . . . . .	196
4.2. Servicio FTP . . . . .	152	5.10. Protocolo HTTP . . . . .	197
4.2.1. Características . . . . .	152	5.10.1. Funcionamiento básico . . . . .	197
4.2.2. Componentes y funcionamiento . . . . .	152	5.10.2. Mensajes HTTP . . . . .	198
4.2.3. Servidores FTP . . . . .	153	5.10.3. Métodos de petición . . . . .	200
4.2.4. Clientes FTP . . . . .	154	5.10.4. Cabeceras . . . . .	204
4.2.5. Protocolo FTP . . . . .	155	5.10.5. Códigos de estado y error . . . . .	205
4.2.6. Tipos de acceso . . . . .	156	5.10.6. Almacenamiento en caché ( <i>caching</i> ) . . . . .	206
4.2.7. Conexiones y modos . . . . .	156	5.10.7. Redirecciones . . . . .	206
4.2.8. Tipos de transferencia de archivos . . . . .	162	5.10.8. Compresión . . . . .	207
4.2.9. Seguridad . . . . .	162	5.10.9. Cookies . . . . .	207
4.2.10. FTPS (o FTP/SSL) . . . . .	163	5.10.10. Autenticación . . . . .	209
4.2.11. Protocolo FXP . . . . .	163	5.10.11. Conexiones persistentes . . . . .	209
4.3. Servicio TFTP . . . . .	164	5.11. MIME . . . . .	209
4.4. Servicios SFTP/SCP . . . . .	165	5.12. Seguridad . . . . .	211
4.5. Prácticas resueltas . . . . .	167	5.13. Protocolo HTTPS . . . . .	211
Práctica 4.1: Clientes FTP . . . . .	167	5.14. Alojamiento virtual de sitios web . . . . .	212
Práctica 4.2: Instalación del servidor FTP <i>Filezilla</i> en <i>Windows</i> . . . . .	170	5.14.1. Alojamiento virtual basado en IPs . . . . .	213
Práctica 4.3: Configuración del servidor FTP <i>Filezilla</i> en <i>Windows</i> . . . . .	171	5.14.2. Alojamiento virtual basado en nombres . . . . .	214
Práctica 4.4: Instalación y configuración por defecto del servidor <i>vsftpd</i> en <i>Linux</i> . . . . .	176	5.14.3. Alojamiento virtual basado en puertos . . . . .	216
Práctica 4.5: Configuración básica del servidor <i>vsftpd</i> en <i>Linux</i> . . . . .	179	5.14.4. Combinaciones . . . . .	217
4.6. Prácticas propuestas . . . . .	182	5.15. Prácticas resueltas . . . . .	218
Práctica 4.6: Otros clientes FTP . . . . .	182	Práctica 5.1: Clientes web (Navegadores) . . . . .	218
Práctica 4.7: Clientes FTP (Modos activo y pasivo) . . . . .	182	Práctica 5.2: Protocolo HTTP . . . . .	219
Práctica 4.8: Configuración avanzada del servidor <i>vsftpd</i> en <i>Linux</i> . . . . .	182	Práctica 5.3: Instalación y configuración por defecto del servidor web <i>Apache 2.4</i> en <i>Linux</i> . . . . .	219
Práctica 4.9: Instalación y configuración por defecto del servidor FTP de IIS 7.5 en <i>Windows 7</i> . . . . .	183	Práctica 5.4: Configuración del servidor <i>Apache 2.4</i> en <i>Linux</i> . Parte 1 . . . . .	225
Práctica 4.10: Configuración del servidor FTP de IIS 7.5 en <i>Windows 7</i> . Parte 1 . . . . .	183	Práctica 5.5: Configuración del servidor <i>Apache 2.4</i> en <i>Linux</i> . Parte 2 . . . . .	228
Práctica 4.11: Configuración del servidor FTP de IIS 7.5 en <i>Windows 7</i> . Parte 2 . . . . .	183	Práctica 5.6: Configuración del servidor <i>Apache 2.4</i> en <i>Linux</i> . Parte 3 . . . . .	229
Práctica 4.12: Configuración de servidores FTPS (FTPS Explícito) . . . . .	184	Práctica 5.7: Configuración del servidor <i>Apache 2.4</i> en <i>Linux</i> . Parte 4 . . . . .	231

## 5.1. Introducción

HTTP (*Hyper Text Transfer Protocol*) es un protocolo de capa de aplicación que facilita a los usuarios, de una forma sencilla e intuitiva, el acceso a la información hipertextual remota de sistemas conectados a una red TCP/IP. El modelo cliente/servidor y el protocolo HTTP son la base de la WWW (*World Wide Web* o simplemente *Web*).

## 5.2. WWW

La WWW (*World Wide Web*) es un servicio de distribución de información que permite acceder a millones de recursos electrónicos (documentos, imágenes, audio, vídeo...) y aplicaciones distribuidos en servidores por todo Internet e identificados y localizados por direcciones (URIs o URLs). Estos recursos están conectados entre sí a través de hiperenlaces (o hipervínculos) lo que permite "navegar" de unos a otros fácilmente.

**¿Sabías que ...?** Un hiperenlace (también llamado enlace, vínculo o hipervínculo) es un elemento de un documento electrónico que hace referencia a otro recurso, por ejemplo, otro documento o un punto específico del mismo o de otro documento. (Fuente: Wikipedia). Los hiperenlaces son muy importantes para aparecer en las primeras páginas de los resultados de los buscadores de Internet. Entre muchos otros aspectos, los buscadores consultan el número de enlaces de otras webs (y su importancia) hacia una web determinada como criterio para posicionarla en sus resultados.

- **Actividad 5.1:** Busca información en Internet sobre posicionamiento en buscadores o posicionamiento web (SEO, *Search Engine Optimization*). Investiga sobre el *Pagerank* de Google ¿cómo puedes saber el *Pagerank* de una web?

## 5.3. W3C y estándares Web

La WWW fue desarrollada por el CERN (Centro Europeo de Investigación Nuclear) en el año 1989 y actualmente su desarrollo está controlado por el W3C (*World Wide Web Consortium*) (<http://www.w3.org/>).

- **Actividad 5.2:**

- Consulta la web [www.w3.org](http://www.w3.org) del W3C y observa los estándares que desarrolla. Pincha en el hiperenlace **W3C A to Z** para tener una visión general de todos los estándares de la Web.
- Accede a la web en español [www.w3c.es](http://www.w3c.es). Consulta el hiperenlace **Documentos y guías** que muestra el material en español basado en las especificaciones técnicas del W3C, las **Guías breves** y el **glosario de la A a la Z**.



## 5.4. Componentes y funcionamiento

El servicio que ofrece la Web se basa en el modelo cliente/servidor y está formado por los siguientes componentes.

- **Recursos.** Documentos, vídeos, imágenes, audio, aplicaciones, buzones de correo, etc., accesibles a través de servidores web y conectados por hiperenlaces.
- **Nombres y direcciones (URIs y URLs).** Sistema de nombres basado en cadenas de caracteres que identifican y localizan inequívocamente a los recursos en la Web.
- **Clientes web (clientes HTTP o navegadores).** Permiten a los usuarios acceder a los recursos disponibles en servidores web. Establecen conexiones con los servidores web, dialogan con ellos e interpretan la información que obtienen mostrándosela a los usuarios.
- **Servidores web (o servidores HTTP).** Atienden las peticiones de los clientes y les envían los recursos solicitados.
- **Proxies web (o proxies HTTP).** Programas intermediarios entre clientes y servidores web. Pueden actuar como cortafuegos y/o almacenar datos en *cache* para aumentar el rendimiento.
- **Protocolo HTTP.** Conjunto de normas y reglas en base a las cuales "dialogan" los clientes, los servidores web y los *proxies*. Usa TCP como protocolo de transporte.
- **Tecnologías web.** Utilizadas para desarrollar aplicaciones basadas en la Web (XHTML, CSS, XML, Ajax, XQuery, XPath, RDF, etc., <http://www.w3c.es/Divulgacion/GuiasBreves/>)

Estos componentes interactúan entre sí para crear sitios web que ofrecen a los usuarios páginas web y aplicaciones web, Figura 5.1.

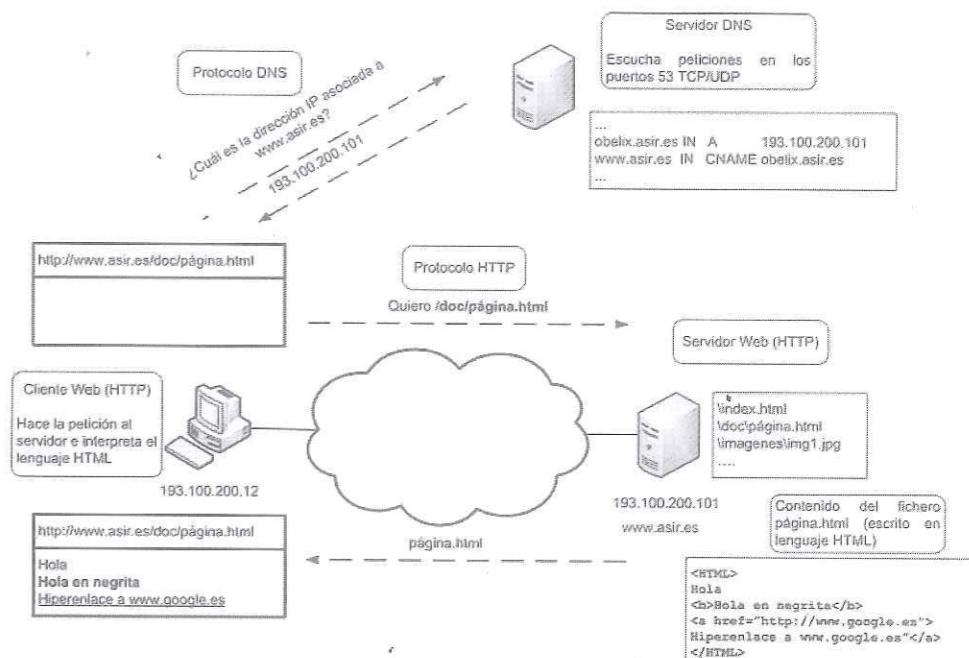


Figura 5.1: Componentes Web

## 5.5. Nombres y direcciones (URIs y URLs)

La Web se puede ver como un conjunto de recursos distribuidos y conectados entre sí. Para poder localizar los recursos y acceder a ellos se utilizan cadenas de caracteres (por ejemplo, `http://www.asir.net/pagina.html`) que los identifican inequívocamente. Estas cadenas de caracteres se denominan identificadores uniformes de recursos (URI, *Uniform Resource Identifier*) y permiten acceder a los recursos disponibles usando una gran variedad de sistemas de denominación y métodos de acceso, llamados esquemas, como HTTP, FTP, etc. Los localizadores universales de recursos (URL, *Universal Resource Locator*) son tipos de URIs.

◦ **Actividad 5.3:** Existe cierta confusión en la comunidad web sobre la relación y de los términos URI, URL y URN (*Uniform Resource Name*). Se debe a la incompatibilidad de dos visiones diferentes de la estructura de una URI, denominados punto de vista clásico y contemporáneo. Puedes ampliar información, profundizar sobre los formatos de URIs, URLs y URNs e intentar aclarar la confusión existente consultando la web <http://www.w3.org/Addressing/> de la W3C.

Para simplificar en el libro usaremos el término URL como sinónimo de URI, aunque técnicamente los URLs son parte de los URI.

No es el objetivo del capítulo explicar en profundidad los diferentes formatos de URI (o URL). A continuación se describen algunos ejemplos significativos que permiten tener una visión práctica de su significado y utilidad en la Web:

- `http://193.168.200.101/pagina.html`
  - http. Esquema o protocolo que se utiliza para acceder al recurso.
  - 193.168.200.101. Dirección IP de la máquina donde está el recurso.
  - /pagina.html. Ruta del recurso solicitado relativa al directorio raíz del servidor web.
- `http://aula.asir.es:8080/datos/practica1.pdf`
  - http. Esquema o protocolo que se utiliza para acceder al recurso.
  - aula.asir.es. Nombre DNS donde está el recurso.
  - 8080. Puerto en el que escucha peticiones el servidor http.
  - /datos/practica1.pdf. Ruta del recurso solicitado relativa al directorio raíz del servidor web.
- `http://obelix.asir.es/buscarLibros.php?id=2&tema=Historia`
  - http. Esquema o protocolo que se utiliza para acceder al recurso.
  - obelix.asir.es. Nombre DNS donde está el recurso.
  - /buscarLibros.php. Ruta del recurso solicitado relativa al directorio raíz del servidor web.
  - ?id=2&tema=Historia. Parámetros y su valor enviados al servidor.

## 5.6. Páginas web, sitios web y aplicaciones web

Una página web es un documento hipermedia o conjunto de información electrónica relacionada (texto, audio, imágenes, vídeo, etc.) que normalmente contiene hiperenlaces a otras páginas web o recursos. Las páginas web están escritas en lenguajes que son interpretados y/o ejecutados por los navegadores (XHTML, CSS, Java Script, Flash...). Su contenido puede ser estático (almacenado en navegadores) o dinámico (se genera en el servidor web al ejecutar un conjunto de instrucciones de un determinado lenguaje).

Un sitio web es un conjunto de páginas web relacionadas y accesibles a partir de un mismo nombre de dominio DNS. El conjunto de sitios web de Internet constituyen la WWW. Los sitios web se pueden clasificar según múltiple criterios, uno de los más empleados es el tema o contenidos que ofrecen (sitios de redes sociales, sitios de periódicos, sitios de buscadores, etc.).

Una aplicación web es aquella donde el usuario interactúa con un navegador que accede a los servicios y recursos que ofrece un servidor web (por ejemplo, un buscador, una tienda electrónica, un cliente de correo web...).

**¿Sabías que ...?** Computación en la nube (*Cloud computing*) es un modelo que consiste en ofrecer servicios a través de Internet ("la nube"). En la actualidad muchas empresas y organizaciones están apostando por ofrecer cada vez más servicios en "la nube" (por ejemplo *Google Maps*, *Google Docs*, *Dropbox*...) basados en aplicaciones web. Incluso se está empezando a apostar por sistemas operativos que se ejecutan en "la nube" y a los que se accede a través de un navegador (visita <http://www.eyeos.com/>).



## 5.7. Servidores web

Los servidores web, también llamados servidores HTTP, son programas que atienden peticiones HTTP, procesan e interpretan código escrito en diferentes lenguajes y envían a los clientes los recursos solicitados.

Los recursos pueden estar localizados en el mismo equipo donde se ejecuta el servidor o en otros equipos de la red a los que el servidor puede acceder usando protocolos adicionales. El servidor puede enviar tanto contenido estático (archivos en diferentes formatos) como dinámico (el resultado de ejecutar programas).

Ofrecen múltiples opciones de configuración y suelen tener una arquitectura modular que permite ampliar o quitar funcionalidades fácilmente.

Por defecto escuchan peticiones HTTP en el puerto 80/TCP.

Existen múltiples servidores web tanto para sistemas libres como para sistemas propietarios. Algunos de los más utilizados son:

- Apache HTTP server (<http://httpd.apache.org/>).
- IIS (*Internet Information Server*) de Microsoft (<http://www.iis.net/>).
- Nginx (<http://nginx.org/>).
- Servidor Web de Google.
- Lighttpd (<http://www.lighttpd.net/>).

**¿Sabías que ...?** Un servidor de aplicaciones es un producto que ofrece diferentes contenedores en los que se ejecutan distintos componentes que interactúan entre sí para proporcionar un servicio a una empresa. Entre los contenedores siempre suele existir un contenedor Web que recibe peticiones HTTP, FTP, SMP, etc. Además suelen existir más contenedores que ejecutan otros componentes, basados en la mayoría de los casos en la tecnología Java EE (*Java Platform, Enterprise Edition*, conocida hasta la versión 1.4 como J2EE).

Un ejemplo típico consistiría en un servidor de aplicaciones formado por: un componente que recibe las peticiones vía HTTP y genera los eventos asociados, un componente que representa la lógica de negocio de la empresa, que recibe los eventos y realiza las operaciones necesarias y otro componente que genera las respuestas que son enviadas al usuario. Ejemplos de servidores de aplicaciones son *GlassFish* (<http://glassfish.java.net>), *JBoss* (<http://www.jboss.com/>), *WebSphere* (<http://www-03.ibm.com/software/products/en/was-overview>), *Geronimo* (<http://geronimo.apache.org/>), etc.

**¿Sabías que ...?** Un descriptor de implementación de aplicación web es un fichero que permite establecer valores de configuración de la aplicación, que también pueden ser utilizados por el servidor web para responder a las peticiones que recibe. En aplicaciones Java se utiliza el fichero *web.xml* (<https://cloud.google.com/appengine/docs/java/config/webxml>) y en aplicaciones ASP .Net se utiliza el fichero *web.config* ([http://msdn.microsoft.com/es-es/library/ms181993\(v=vs.80\).aspx](http://msdn.microsoft.com/es-es/library/ms181993(v=vs.80).aspx)).

◊ **Actividad 5.4:** *Netcraft* (<http://www.netcraft.com/>) es una compañía que, entre otros servicios, ofrece análisis y comparativas de la Web. Visita su página web.

- En la parte izquierda superior de la página inicial de *Netcraft* hay un formulario “*What's that site running?*” que permite consultar información sobre el servidor web en el que se ejecuta o aloja un sitio web determinado. Divírtete averiguando qué servidor web utilizan los sitios que visitas habitualmente.
- Busca estadísticas de uso de servidores web (*web survey*). Por ejemplo, en la dirección: <http://news.netcraft.com/archives/2014/12/18/december-2014-web-server-survey.html>

Se muestran estadísticas del uso de servidores web en Internet entre Agosto de 1995 y Diciembre del 2014.

## 5.8. Clientes web (navegadores)

Los clientes web o navegadores son programas con los que interactúa el usuario y que le permiten, entre otras funciones, introducir URIs (o URLs) para acceder a recursos disponibles en la red.

Pueden actuar como clientes de diferentes protocolos aunque su función principal es ejercer como clientes HTTP.

Reciben recursos de los servidores web, los procesan y muestran los resultados al usuario, permitiéndole interactuar si es necesario. Si el recurso que recibe el navegador no puede ser interpretado por él, puede redirigirlo a una aplicación externa capaz de gestionarlo o preguntar al usuario qué quiere hacer.

Mantienen una memoria cache en la que almacenan durante un tiempo las direcciones a las que han accedido (histórico), los recursos procesados, las contraseñas introducidas por el usuario en las aplicaciones, etc.

Los navegadores ofrecen múltiples opciones de configuración y personalización. Además, permiten ampliar su funcionalidad con la instalación de plantillas, idiomas, extensiones y complementos.

**¿Sabías que ...?** Aunque la W3C crea estándares para tecnologías web, los navegadores, en determinados aspectos, no cumplen completamente los estándares. Esto supone un problema para los desarrolladores de aplicaciones web, que tienen que adaptarlas para que se muestren y funcionen correctamente en varios navegadores, incluso entre diferentes versiones de un mismo navegador. También es un problema para los usuarios porque accediendo al mismo sitio web con distintos navegadores pueden obtener resultados diferentes.

Existen múltiples navegadores tanto para sistemas libres como para sistemas propietarios. Algunos de los más utilizados son:

- *Internet Explorer* (<http://windows.microsoft.com/es-es/internet-explorer>).
- *Mozilla Firefox* (<https://www.mozilla.org/es-ES/firefox/desktop/>).
- *Google Chrome* (<http://www.google.com/chrome/>).
- *Chromium* (<http://www.chromium.org/Home>).
- *Safari* (<http://www.apple.com/es/safari/>).
- *Opera* (<http://www.opera.com/>).
- Navegadores en modo texto: *Lynx*, *Links*, etc.

◊ **Actividad 5.5:** Inicia sesión en *ubuntuXX*. Inicia *Firefox* y accede en la barra de menú a herramientas, complementos. Observa las extensiones, temas, idiomas y *plugins* instalados. Haz clic en obtener complementos y verás una lista de complementos disponibles para descargar e instalar.

**¿Sabías que ...?** Hay una fuerte competencia entre las empresas que desarrollan navegadores por aumentar su cuota de mercado. Mantienen una lucha constante para mejorar la rapidez, la seguridad, la usabilidad y la funcionalidad de sus productos.

## 5.9. Proxies web

En el ámbito de las redes, el término *proxy* hace referencia a un programa que actúa como intermediario entre otros. Un *proxy web* o *proxy HTTP* hace de intermediario entre un cliente web y un servidor web. Según el comportamiento y la función que realizan pueden ser *proxies directos* y *proxies inversos*:

- **Proxy directo (forward proxy).** Recibe la petición iniciada por un cliente web y se la traslada al servidor web. La solicitud del cliente es hacia el servidor web (la URI que utiliza es la del servidor) no hacia el *proxy*, este solo hace de intermediario o representante del cliente. Usados habitualmente para optimizar y controlar el acceso a redes externas, como Internet, de los clientes de una organización o empresa, Figura 5.2.

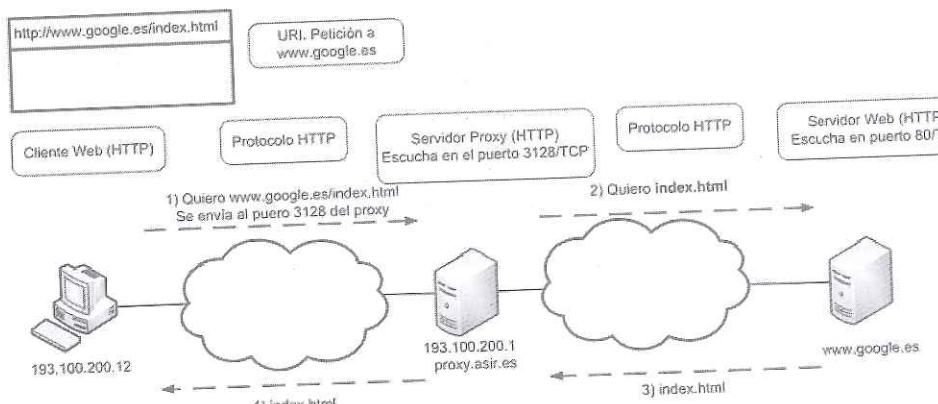


Figura 5.2: Proxy web directo

- **Proxy inverso (reverse proxy).** Igualmente reciben la petición de un cliente web y la reenvían a uno o varios servidores web. En este caso, la solicitud del cliente (la URI que utiliza es la del proxy) es hacia el *proxy* (para los clientes es un servidor web normal). Usados habitualmente para ofrecer acceso a servidores web que están detrás de un cortafuegos y no son accesibles directamente, para balancear la carga entre varios servidores web, para aumentar los accesos a múltiples servidores web en la misma URI, Figura 5.3.

La mayoría de los *proxies* web tiene una *cache* donde almacenan los datos que han obtenido recientemente para reducir el tráfico de red y aumentar la velocidad de acceso a los recursos. Por eso, se utiliza el término *proxy-cache* para referirse a ellos.

Además, al ser un punto central por el que pasa el tráfico web, se pueden configurar como cortafuegos filtrando tráfico no deseado, controlando accesos por usuarios, priorizando contenidos,

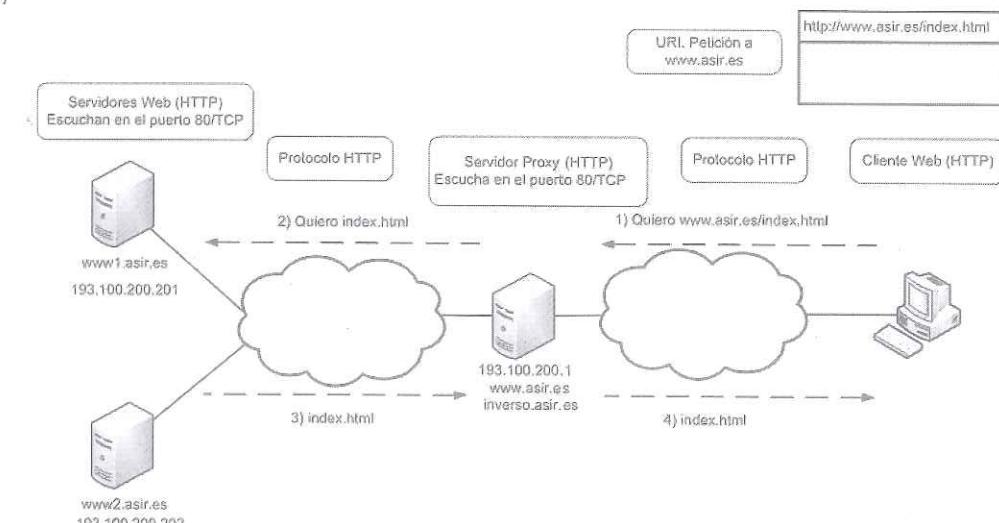


Figura 5.3: Proxy web inverso

etc.

No es el objetivo del capítulo profundizar sobre los *proxies* web. Recomendamos aplicar los contenidos explicados con documentación adicional.

**¿Sabías que ...?** Existen *proxies* específicos para otros servicios como por ejemplo FTP y SMTP. Hacemos referencia en el libro a los *proxies* web porque son los más utilizados.

## 5.10. Protocolo HTTP

HTTP (*HyperText Transfer Protocol*) es de hecho el protocolo de comunicación en la Web. Define las reglas que utilizan los componentes software (clientes, servidores y *proxies*) para comunicarse.

Es un protocolo sin estado que utiliza TCP como protocolo de transporte y determina los tipos de peticiones que los clientes pueden enviar, así como el formato y estructura de las respuestas. También define una estructura de metadatos, en forma de cabeceras que se envían tanto en las peticiones como en las respuestas.

Ha pasado por varias versiones HTTP/0.9 (Obsoleta), HTTP/1.0 y HTTP/1.1 (versión actual). En el sitio web <https://http2.github.io/> mantiene información sobre la siguiente versión prevista, la HTTP/2

### 5.10.1. Funcionamiento básico

A continuación se describe un ejemplo básico y esquemático de interacción entre un cliente y un servidor para explicar a partir de él las principales características del protocolo.



- El usuario introduce una URI (o URL) en la barra de direcciones del navegador o hace clic sobre un hiperenlace.
- El navegador analiza la URL y establece una conexión TCP con el servidor web (si se ha utilizado un nombre DNS previamente invoca al resolver para que le resuelva el nombre) (si no se indica el número de puerto se conectará al puerto 80 del servidor).
- Cuando se ha establecido la conexión TCP, el navegador envía un mensaje HTTP de petición que depende del la URI (o URL).
- El servidor envía un mensaje de respuesta que depende de la petición enviada y del estado del servidor.
- Se cierra la conexión TCP.

◦ **Actividad 5.6:** Inicia sesión en **ubuntuXX** o en **W7XX** con un usuario con privilegios de administrador. Inicia una captura con **Wireshark**, abre el navegador web e introduce la dirección de la web que deseas. Observa el tráfico capturado, analiza el inicio de conexión TCP y los mensajes de petición y respuesta intercambiados entre el cliente y el servidor. Puedes situarte sobre un mensaje HTTP, hacer clic con el botón derecho del ratón y seleccionar **Follow TCP Stream** para analizar el intercambio de mensajes HTTP.

### 5.10.2. Mensajes HTTP

Los mensajes que utiliza HTTP se componen de líneas escritas en texto plano (ASCII) que contienen las órdenes y parámetros con la sintaxis definida por el protocolo. Pueden ser de dos tipos: mensajes de petición y mensajes de respuesta.

En los siguientes apartados se describen en primer lugar la estructura de los mensajes de petición y respuesta HTTP y posteriormente, se explican las características y la utilidad de los diferentes elementos que componen estos mensajes.

#### 5.10.2.1. Mensajes de petición

Los mensajes de petición, Figura 5.4, están formados por tres partes:

- Línea inicial de petición. Incluye:
  - Método utilizado (GET, POST...) (se explica en apartados posteriores).
  - La parte relativa al servidor de la URL o la URL completa si la conexión se establece con un servidor *proxy*.
  - Versión del protocolo utilizada (opcional). Los clientes y servidores actuales usan HTTP/1.1.
- Línea/s de cabecera
  - Conjunto de pares nombre/valor denominados cabeceras (se explican en apartados posteriores) que determinan cómo será procesada la petición por parte del servidor. Por ejemplo, la cabecera *Accept: text/html* indica que el navegador es capaz de procesar código HTML.
  - Si no hay cabeceras se envía un 0.
  - Cada cabecera se muestra en una línea, es decir, las cabeceras se terminan con CR-LF.

- Detrás de la última cabecera se envía una línea en blanco.
- Cuerpo del mensaje (opcional). Contiene parámetros o ficheros a enviar al servidor.

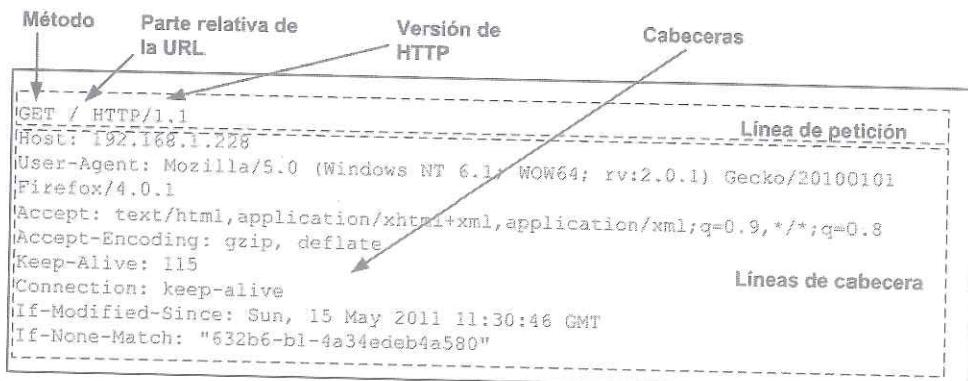


Figura 5.4: Mensaje de petición HTTP

**¿Sabías que ... ?** La utilidad NetCat (<http://netcat.sourceforge.net/>) permite realizar conexiones TCP, tanto entrantes como salientes, desde la línea de comandos. Está incluida en la mayoría de distribuciones GNU/Linux, y puede utilizarse para simular un servidor web y así inspeccionar las cabeceras que envía el navegador. Por ejemplo, puede ejecutarse NetCat para que admita conexiones entrantes por el puerto 8081 ejecutando en un terminal *nc -l 8081*.

Al utilizar un navegador desde el mismo ordenador que visite la dirección *http://localhost:8081*, los mensajes enviados saldrán por el terminal, y los caracteres introducidos en la terminal serán la respuesta recibida por el navegador.

#### 5.10.2.2. Mensajes de respuesta

Los mensajes de respuesta, Figura 5.5, están formados por tres partes:

- Línea inicial de respuesta (línea de estado)
  - Versión HTTP utilizada.
  - Código de estado o código de error que informa al cliente de cómo ha sido procesada la petición. Por ejemplo, el código 200 indica que la petición se ha procesado correctamente y que el recurso correspondiente se envía al cliente.
  - Texto explicativo del código de estado.
- Línea/s de cabecera
  - Conjunto de pares nombre/valor denominados cabeceras (se explican en apartados posteriores) que describen los datos y la forma en que son enviados al cliente. Por ejemplo, la cabecera *Content-Type:text/html* indica que se envía código HTML que deberá interpretar el navegador.
  - Si no hay cabeceras se envía un 0.

- Cada cabecera se muestra en una línea, es decir, las cabeceras se terminan con CR-LF.
- Detrás de la última cabecera se envía una línea en blanco.
- Cuerpo del mensaje (opcional). Queda determinado por el tipo de recursos solicitado.

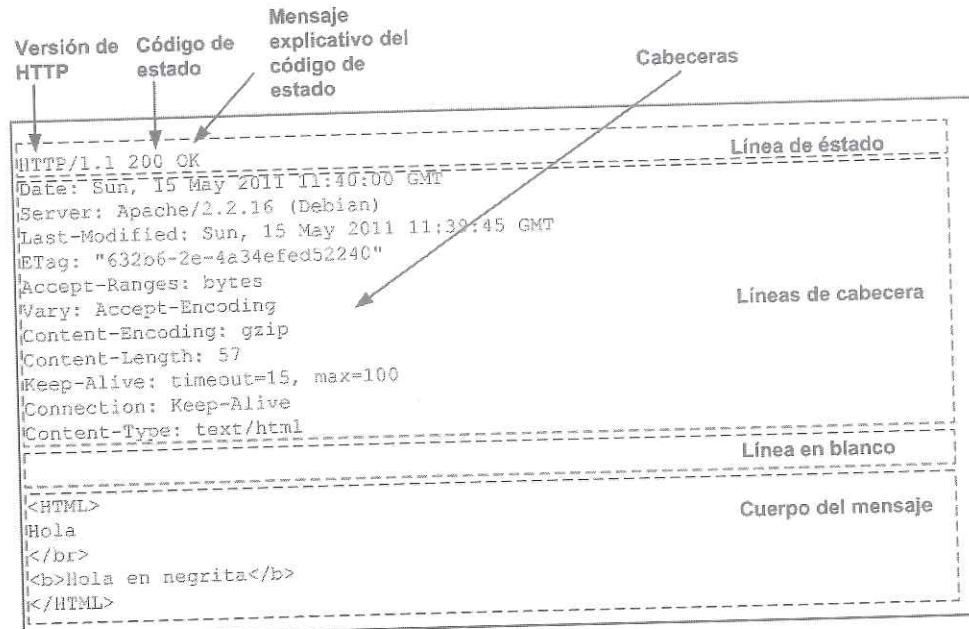


Figura 5.5: Mensaje de respuesta HTTP

**¿Sabías que ...?** Las imágenes no se envían directamente en el código HTML. Para hacer referencia a una imagen se utiliza la etiqueta *img*. Por ejemplo:



Cuando un navegador recibe una etiqueta *img* comprueba si tiene una copia válida en su caché, y si no es el caso envía otra petición HTTP al servidor. Por lo tanto, una misma página web puede generar múltiples peticiones HTTP.

Este comportamiento puede ocurrir con muchos otros elementos usados en lenguajes para crear páginas y aplicaciones web. Por lo tanto, cuando el navegador recibe un recurso del servidor puede darse el caso de que internamente (sin intervención del usuario) se realicen múltiples peticiones HTTP a otros recursos.

### 5.10.3. Métodos de petición

Los métodos de petición especifican la operación que quiere realizar el cliente en el servidor. La versión HTTP 1.1 contempla siete métodos de petición que describimos brevemente a continuación:

- **Método GET**, Figuras 5.6, 5.7 y 5.8
  - Es el método más utilizado y se emplea para obtener cualquier tipo de información del servidor.

- Se invoca cuando se introduce una URL en el navegador, cuando se pincha sobre un hiperenlace o cuando se envía un formulario GET.
- Permite enviar parámetros al servidor en la URI (o URL) (conocidos como *Query String*)
  - Para codificar los parámetros como parte de la URI
  - Se añaden a la URI detrás del nombre del recurso, separados de este por un carácter ?.
  - Los diferentes parámetros se separan entre sí por el carácter &.
  - Los espacios se sustituyen por +.
  - Ejemplo: www.asir.es/datos.php?nombre=Juan&edad=28
  - Los parámetros se incluyen a la línea inicial de petición.
- Las peticiones GET no envían cuerpo de mensaje.
- No se puede usar para subir archivos o realizar otras operaciones que requieran enviar una gran cantidad de datos al servidor.

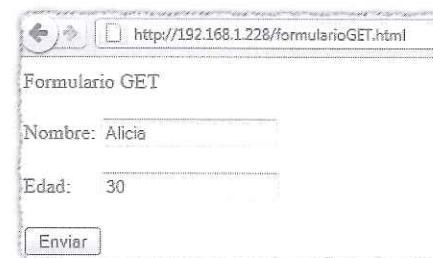


Figura 5.6: Método GET. Formulario HTML

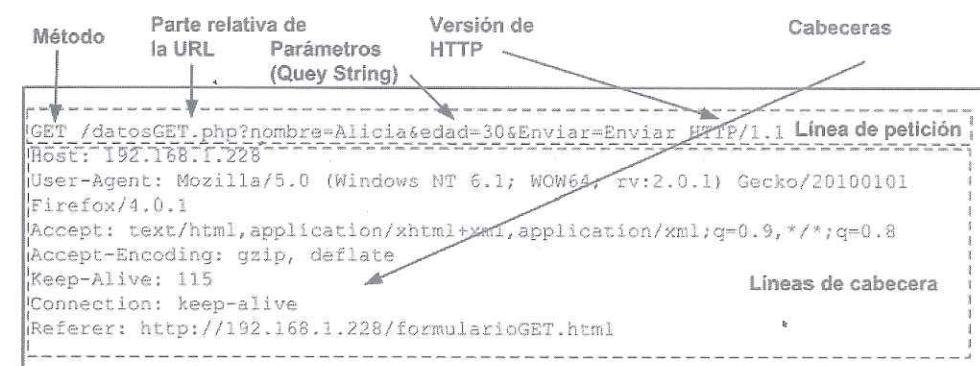


Figura 5.7: Método GET. Petición

### ■ Método POST, Figuras 5.9, 5.10, 5.11 y 5.12

- Se emplea para solicitar al servidor que acepte información que se envía adjunta en una petición.
- Las peticiones POST envían el cuerpo de mensaje y en él se incluyen los parámetros y los datos que se consideren. Los parámetros no son visibles por lo tanto en la URL.
- Se invoca normalmente como consecuencia de enviar un formulario HTML POST.

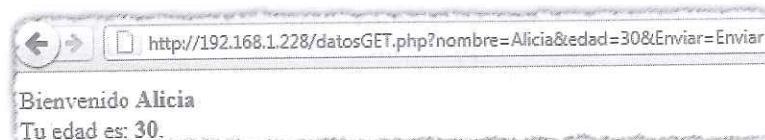


Figura 5.8: Método GET. Respuesta de página PHP

- Se utiliza en operaciones que no deberían ser repetidas porque pueden tener efectos negativos (por ejemplo, ordenar una transferencia bancaria). Los navegadores muestran mensajes de advertencia cuando se refresca una página con una petición POST.
- No hay límite en la cantidad de datos que se pueden enviar. Se utiliza, por lo tanto, para enviar ficheros al servidor.
- No es recomendable (y no se hace por defecto) cachear las respuestas obtenidas ante una petición POST.

Figura 5.9: Método POST. Formulario POST

Diagrama que muestra la estructura de una petición HTTP POST:

```

    graph TD
      subgraph Request [Petición]
        direction TB
        A[Método] --> B[Parte relativa de la URL]
        B --> C[Versión de HTTP]
        C --> D[Línea de petición]
        D --> E[Líneas de cabecera]
        E --> F[Cuerpo del mensaje]
      end
  
```

Detalles de la petición:

Línea de petición:

```
POST /datosPOST.php HTTP/1.1
```

Líneas de cabecera:

```

HOST: 192.168.1.228
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:2.0.1) Gecko/20100101
Firefox/4.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Keep-Alive: 115
Connection: keep-alive
Referer: http://192.168.1.228/formularioPOST.html
Content-Type: application/x-www-form-urlencoded
Content-Length: 36
  
```

Cuerpo del mensaje:

```
nombre=Antonio&edad=20&Enviar=Enviar
```

Figura 5.10: Método POST. Petición

#### Método OPTIONS

- Para solicitar al servidor información sobre las opciones de comunicación disponibles de un recurso determinado.

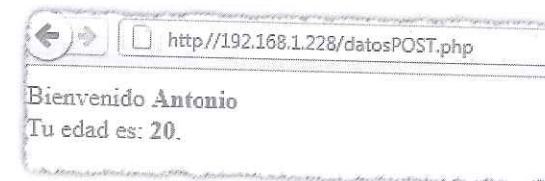


Figura 5.11: Método POST. Respuesta de página PHP

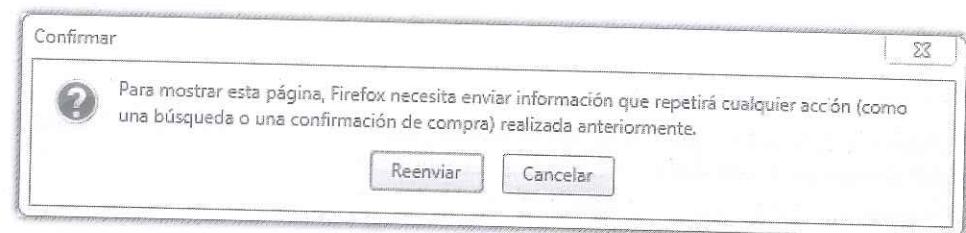


Figura 5.12: Método POST. Mensaje de advertencia al refrescar una petición POST

#### Método HEAD

- Para recuperar las cabeceras de una página web.
- Similar a GET pero el servidor devuelve cabeceras (por ejemplo HEAD /index.html HTTP/1.1).
- Usado para implementar *cachés* de navegadores, informar al usuario del tamaño del recurso antes de intentar recuperarlo, etc.

#### Método PUT

- Para enviar recursos (subir) al servidor.
- Por seguridad no es habitual que los servidores web permitan subir recursos usando el método PUT.

#### Método DELETE

- Para eliminar recursos del servidor.
- Por seguridad no es habitual que los servidores web permitan eliminar recursos usando el método DELETE.

#### Método TRACE

- Para trazar la ruta de una petición a través de *proxies* y cortafuegos.
- Usado para depurar errores en redes complejas.

**¿Sabías que ... ?** Los parámetros enviados a un servidor Web son procesados usando lenguajes como PHP, JSP, ASP, Ruby, etc., los cuales permiten generar contenido dinámico en función de los valores enviados.

#### 5.10.4. Cabeceras

Las cabeceras son pares de nombre/valor que se pueden incluir en los mensajes de petición y respuesta HTTP. Definen información (metadatos) sobre los datos que se intercambian los clientes y servidores, sobre los propios clientes y servidores, y sobre la propia transferencia de información. Tienen el mismo formato que las cabeceras de los protocolos usados en los servicios de correo electrónico y noticias.

Existen una gran número de tipos de cabeceras definidas por HTTP y se pueden clasificar como se muestra a continuación.

##### ■ Generales

- Definen información que puede ser utilizada tanto por clientes como por servidores, ya que se aplican a una sesión completa de comunicación.
- Proporcionan información sobre:
  - Control de caché (por ejemplo *Caché-Control*).
  - Fechas (por ejemplo *Date*).
  - Codificación de la transferencia (por ejemplo, *Transfer-Encoding*).
  - etc.

##### ■ De petición (o de cliente)

- Empleadas por los clientes para enviar información al servidor.
- Proporcionan información sobre:
  - El propio navegador (por ejemplo *User-Agent*).
  - Nombre y puerto del servidor al que se dirige la petición (*Host*).
  - Tipos MIME, compresión, mapas de caracteres, idiomas que está dispuesto a aceptar el navegador, etc. (Por ejemplo *Accept*).
  - Cookies (por ejemplo *Cookies*).
  - etc.

##### ■ De respuesta (o de servidor)

- Empleadas por el servidor para enviar información añadida al cliente.
- Proporcionan información sobre:
  - La edad de la respuesta (por ejemplo *Age*).
  - Si un recurso no está disponible, fecha en la que se espera que esté disponible (por ejemplo *Retry-After*).
  - El tipo servidor (por ejemplo *Server*).
  - Si servidor necesita autorización para acceder al recurso pedido (por ejemplo *WWW-Authenticate*).
  - etc.

##### ■ De entidad

- Información relacionada directamente con el recurso que se le va a proporcionar al cliente.
- Proporcionan información sobre:
  - Codificación (por ejemplo *Content-Encoding*).
  - Idioma (por ejemplo *Content-Language*).
  - Longitud (por ejemplo *Content-Length*).

- Tipo MIME de los recursos (por ejemplo *Content-Type*).
- etc.

HTTP/1.0 define 16 cabeceras (ninguna obligatoria) y HTTP/1.1 define 46 cabeceras (la cabecera *Host* es obligatoria en las peticiones para permitir el uso de sitios o servidores virtuales, se explican posteriormente en este capítulo).

Las aplicaciones web pueden definir sus propias cabeceras para añadir información adicional en los mensajes HTTP. Como convenio, se ha establecido que se utilice el prefijo X- para indicar que es una cabecera no estándar (por ejemplo *X-asir*).

- **Actividad 5.7:** Consulta la web <http://www.w3.org/Protocols/rfc2616/rfc2616.html> y busca información sobre las cabeceras HTTP/1.1 disponibles. También puedes consultar Wikipedia [http://en.wikipedia.org/wiki/List\\_of\\_HTTP\\_headers](http://en.wikipedia.org/wiki/List_of_HTTP_headers).

#### 5.10.5. Códigos de estado y error

Códigos que envían los servidores en las respuestas HTTP y que informan al cliente de cómo ha sido procesada la petición. Se acompañan de un texto explicativo.

Son números de tres cifras que se clasifican en 5 grupos en función del primer dígito:

- **100 - 199 (Informativo, *Informational*)**
  - Indican que el servidor ha recibido la petición pero no ha finalizado de procesarla.
- **200 - 299 (Éxito, *Successful*)**
  - Usados cuando la petición ha sido procesada satisfactoriamente.
  - Ejemplo: 200 “OK”.
- **300 - 399 (Redirección, *Redirection*)**
  - Indican que la petición ha sido procesada y redirigida a otra localización.
  - Ejemplos:
    - 302 “*Found*”. El recurso solicitado ha sido movido temporalmente a otra localización. El navegador debería buscarlo en la localización indicada en la cabecera *Location* de la respuesta.
    - 304 “*Not Modified*”. El recurso solicitado no se ha modificado y el navegador debería leerlo de su caché local.
- **400 - 499 (Errores del cliente, *Client Error*)**
  - El servidor indica que hay un algún error en la petición del cliente o que no se puede conceder.
  - Ejemplos:
    - 400 “*Bad request*”. Error de sintaxis en la petición.
    - 401 “*Unauthorized*”. Un usuario anónimo no está autorizado para acceder al recurso solicitado.
    - 403 “*Forbidden*”. El servidor no acepta la petición.
    - 404 “*Not found*”. El recurso solicitado no está disponible en el servidor, Figura 5.13.





Figura 5.13: Error 404

#### ■ 500 - 599 (Errores en el servidor, *Server Error*)

- El servidor no puede atender una petición porque ha existido algún problema.
- Ejemplos:
  - 500 “*Internal Server Error*”. Error interno en el servidor, por ejemplo, por un fallo de configuración.
  - 503 “*Service Unavailable*”. El servidor no puede responder en ese momento, por ejemplo, porque está sobrecargado.

◦ **Actividad 5.8:** Consulta la web <http://www.w3.org/Protocols/rfc2616/rfc2616.html> y busca información sobre las códigos de error y estado disponibles. También puedes consultar Wikipedia [http://en.wikipedia.org/wiki/List\\_of\\_HTTP\\_status\\_codes](http://en.wikipedia.org/wiki/List_of_HTTP_status_codes).

#### 5.10.6. Almacenamiento en caché (*caching*)

Los servidores web contienen información que no cambia durante períodos de tiempo muy largos. HTTP soporta almacenamiento en caché para aumentar el rendimiento y evitar tráfico innecesario ¿por qué descargarse 20 veces la misma imagen si se accede 20 veces a ella en mismo día? Los navegadores y los *proxies* pueden almacenar localmente recursos durante un tiempo.

Se definen cabeceras (*Caché-Control*, *Last-Modified*, *Expires*, *Age*, *Etag*, *If-Match*, *If-Modified-Sinze...*) que permiten controlar qué se almacena en caché, cuánto tiempo, qué no se puede almacenar (¿por qué almacenar el resultado de un partido de baloncesto que cambia constantemente?), si la información se puede “cachear” en un *proxy* o no, si hay que actualizar algo que está en caché porque se ha modificado, etc.

#### 5.10.7. Redirecciones

HTTP permite a los servidores y a los *proxies* redirigir las peticiones a otras localizaciones. Como ya sabemos, el servidor le indica la redirección al cliente mediante un código 3XX.

Algunas situaciones en las que puede ser útil:

- El contenido de un servidor se ha movido a una URI (o URL) diferente.
- Convertir una solicitud POST en una solicitud GET.
- Redirigir la petición a otro *proxy*.
- Redirigir peticiones a *Servlets*, *JSPs*, *ASPs*, etc.

#### 5.10.8. Compresión

Es posible que los servidores compriman (por ejemplo usando *gzip*) los recursos solicitados antes de enviárselos a los clientes para reducir el tráfico de red. Los clientes usan cabeceras en los mensajes de petición (*Accept-Encoding*) para indicar que soportan compresión y los servidores para hacerle saber al cliente (*Content-Encoding*) que envían datos comprimidos.

#### 5.10.9. Cookies

Una *cookie* es un fragmento de información que envía un servidor web en una respuesta HTTP y es almacenada, si su configuración lo permite, por el navegador para su uso futuro. El navegador puede enviar la *cookie* en solicitudes posteriores al mismo servidor.

- Los servidores envían las *cookies* usando cabeceras (*Cookies* y *Set-Cookie*) y pueden incluir los siguientes detalles:
  - Nombre (*Name*). Nombre de la *cookie*.
  - Contenido o valor (*Value*). Texto con el valor de la *cookie*.
  - Fecha de expiración (*Expires*). Fecha/hora en la que la *cookie* deberá ser descartada por el navegador.
  - Ruta (*Path*). Lugar donde la *cookie* será guardada por el navegador.
  - Dominio (*Domain*). Nombre de dominio de donde “viene” la *cookie*.
- Cuando un navegador realiza una solicitud a un servidor HTTP consulta las *cookies* que tiene almacenadas. Si existen *cookies* no caducadas cuya ruta y dominio coincidan con los de la petición, se las envía al servidor usando cabeceras (*Cookies* y *Set-Cookie*).

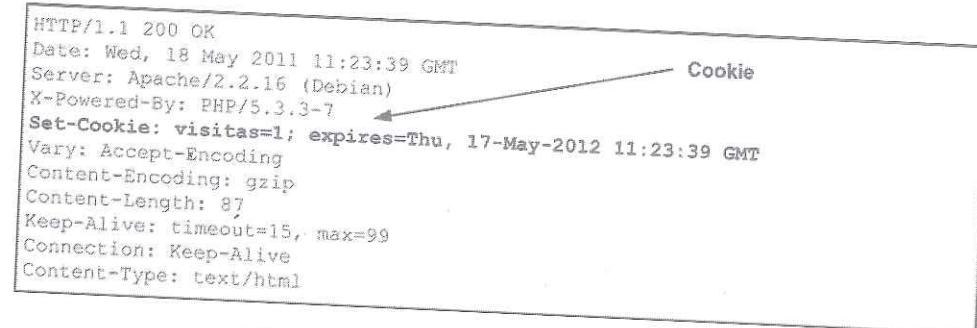


Figura 5.14: Cookies en una respuesta HTTP

Las *cookies* son utilizadas por los servidores web para diferenciar usuarios y conexiones y actuar en consecuencia. Recuerda que HTTP es un protocolo “sin estado”, cada transferencia de datos es independiente de la anterior sin ninguna relación entre ellas. ¿Cómo es posible que cuando accedes a una web, en la que introduces tu nombre de usuario y contraseña, el servidor web recuerde que eres tú mientras navegas por diferentes páginas, que implican varias peticiones HTTP (es lo que se conoce como sesión)? ¿Por qué determinadas páginas te muestran anuncios, noticias, productos... personalizados? Estas y otras cuestiones se pueden resolver mediante el uso de *cookies*.

Las *cookies* pueden ser aceptadas, bloqueadas o borradas según se configure el navegador.

- **Actividad 5.9:** Accede a un navegador de tu equipo. Visita varias páginas web de Internet. Accede a las propiedades de configuración del navegador y busca cómo puedes consultar las *cookies* que tienes almacenadas. Comprueba qué *cookies* han enviado las páginas que has visitado al iniciar la actividad, Figura 5.15. Investiga cómo puedes configurar si el navegador acepta *cookies*, qué tipo acepta, de qué webs, etc.

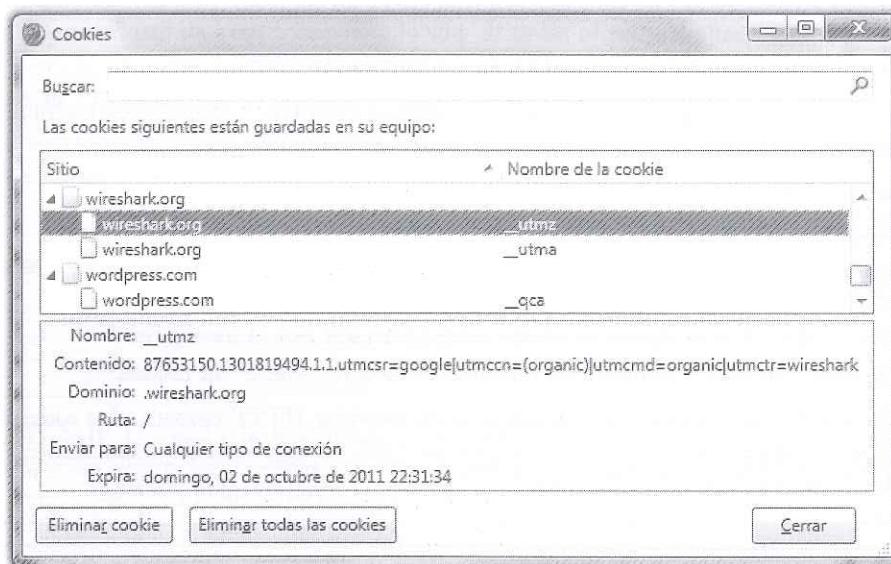


Figura 5.15: Cookies almacenadas en el navegador

- **Actividad 5.10:** Las *cookies* en sí mismas no son un código malicioso, solo son texto, pero pueden ser utilizadas con fines malintencionados para obtener datos privados de usuarios como hábitos de navegación y sitios visitados, pueden ser robadas y/o falsificadas para realizar ataques de seguridad, etc. Consulta la página de Wikipedia sobre las *cookies* para obtener más información.

#### ◦ Actividad 5.11:

- Busca información en Internet sobre las nuevas normativas Europeas de cómo se controla y rastrea a los usuarios en Internet. ¿Cómo se relaciona con las *cookies*?
- ¿En qué consiste la opción *do-not-track* que incorporan las nuevas versiones de los navegadores?
- Investiga qué formas existen para realizar un *tracking* de los usuarios sin necesidad de *cookies*, como el uso de la cabecera *ETag*, o el *canvas fingerprinting*.

#### 5.10.10. Autenticación

HTTP soporta el uso de mecanismos de autenticación para controlar el acceso a los recursos que ofrece el servidor. Estos mecanismos están basados en el uso del código de estado 401 y en las cabeceras *WWW-Authenticate* y *Authorization*. Los navegadores muestran al usuario un cuadro de diálogo para que se identifique.

Algunos mecanismos son:

- **Basic.** El cliente envía un usuario y una clave codificados con el algoritmo *base64*. Método no seguro, es trivial capturar y obtener la clave enviada.
- **Digest.** El cliente envía un usuario y una función *hash* (resumen) de la clave al servidor (se utiliza el algoritmo MD5). Es más seguro que el método *Basic* pero es vulnerable a varios ataques y por lo tanto también es inseguro.

La autenticación que ofrece HTTP no es segura y por ello la autenticación se traslada a las aplicaciones web. En las aplicaciones web actuales la autenticación se basa en el uso de formularios XHTML en los que se introducen usuario y clave que son enviados al servidor (usando parámetros de los métodos GET o POST), donde son tratados por alguna aplicación escrita en un lenguaje como PHP, JSP, ASP, Ruby, etc., o en el uso de certificados digitales y HTTPS (*Hypertext Transfer Protocol Secure*) (se explica en apartados posteriores del capítulo).

#### 5.10.11. Conexiones persistentes

El uso de conexiones persistentes en HTTP consiste en que varias peticiones y respuestas sean transferidas usando la misma conexión TCP. Su uso reduce el número de conexiones TCP, lo que repercute en un menor gasto de CPU/memoria y una reducción de los tiempos de respuesta.

En HTTP/1.0 el comportamiento por defecto es crear una conexión TCP por cada interacción de petición respuesta. Existe la posibilidad de utilizar conexiones persistentes mediante la cabecera *Connection: Keep-Alive* pero implica una serie de problemas.

En HTTP/1.1 se introducen las conexiones persistentes por defecto. Se definen varias cabeceras (*Connection*, *Content-Length...*) y temporizadores en servidores y clientes para controlar cuándo hay que cerrar las conexiones TCP.

- **Actividad 5.12:** Inicia sesión en *ubuntuXX* o en *w7XX* con un usuario con privilegios de administrador. Inicia una captura con *Wireshark*, abre el navegador web e introduce la dirección de la web que deseas. Sitúate sobre un mensaje HTTP, haz clic con el botón derecho del ratón y selecciona *Follow TCP Stream* para analizar el intercambio de mensajes HTTP. Observa que en la misma conexión TCP hay varias peticiones y respuestas HTTP.

#### 5.11. MIME

MIME (*Multipurpose Internet Mail Extensions*) consiste en una serie de especificaciones orientadas a intercambiar en Internet, usando protocolos como HTTP y SMTP, todo tipo de recursos (texto, audio, vídeo, imágenes...) de forma transparente a los usuarios. Inicialmente, fueron usadas en el correo electrónico pero en la actualidad se ha generalizado su utilización a

otros servicios.

Entre otros aspectos definen:

- Una serie de tipos y subtipos (por ejemplo *text/html*, *image/gif*, *text/css*, *audio/x-mpeg*, *multipart/form-data*, *video/mpeg-2*,...) que determinan el contenido de los recursos enviados a través de la red. Son utilizados por los clientes y servidores para saber el tipo de recurso que reciben y cómo tienen que manejarlo.
- Un conjunto de reglas para codificar mensajes no ASCII.
- Un conjunto de cabeceras que son utilizadas por los protocolos para informar a los clientes y servidores sobre los recursos transmitidos:
  - *MIME-Version*. Versión de MIME usada.
  - *Content-Description*. Texto que describe el contenido.
  - *Content-Id*. Identificador único.
  - *Content-Transfer-Encoding*. Cómo se codifican los datos binarios usando ASCII.
  - *Content-Type*. Tipo y subtipo.

En HTTP se utilizan las cabeceras y tipos MIME en:

- Los mensajes de respuesta que envía el servidor para informar al cliente de los recursos que le envía. El navegador en función del tipo MIME recibido visualiza o reproduce el recurso, lo abre con una aplicación externa o le pregunta al usuario qué hacer.
- Los mensajes de petición que envían los clientes para informar al servidor de los tipos MIME que acepta.
- Para encapsular uno o más recursos en el cuerpo de mensaje usando un tipo MIME especial denominado *multipart*.

◊ **Actividad 5.13:** Accede las webs <http://www.iana.org/assignments/media-types/index.html> y <http://www.iana.org/assignments/transfer-encodings/transfer-encodings.xml> y observa los tipos MIME y códigos de transferencia existentes.

```
HTTP/1.1 200 OK
Date: Wed, 18 May 2011 11:23:39 GMT
Server: Apache/2.2.16 (Debian)
X-Powered-By: PHP/5.3.3-7
Set-Cookie: visitas=1; expires=Thu, 17-May-2012 11:23:39 GMT
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 87
Keep-Alive: timeout=15, max=99
Connection: Keep-Alive
Content-Type: text/html
```

Cabecera, tipo y subtipo MIME

Figura 5.16: MIME

## 5.12. Seguridad

HTTP no es un protocolo seguro:

- El intercambio de información se realiza en texto plano. Es vulnerable a ataques de análisis de tráfico de red (*sniffing*).
- Los mecanismos de autenticación como *Basic* y *Digest* no son seguros.
- No se usan mecanismos para garantizar que los equipos involucrados en la transferencia son quienes dicen ser. Es vulnerable a ataques de suplantación de identidad (*spoofing* y *man-in-the-middle*).
- Existen ataques que se basan en el robo o falsificación de *cookies* y/o parámetros enviados en la URL o en el contenido de los mensajes, y que permiten al atacante "robar la identidad de un usuario" y suplantarle en webs (bancos, *webmails*, redes sociales...).

Los clientes y servidores web tienen vulnerabilidades que pueden ser aprovechadas por potenciales atacantes para comprometer los datos y los sistemas donde se ejecutan.

Las aplicaciones web también tienen vulnerabilidades usadas para comprometer la propia aplicación, los datos de sus usuarios y los sistemas y redes que intervienen en su funcionamiento.

Actualmente, existen una gran cantidad de aplicaciones y servicios basados en la Web (redes sociales, correo electrónico, comercio electrónico, banca por Internet, *wikis*, *blogs*, etc.) con una gran repercusión económica y social. Por ello, todos los componentes que intervienen en la arquitectura de la Web (protocolos, clientes, servidores, aplicaciones...) son un objetivo constante de atacantes.

◊ **Actividad 5.14:** Busca en información en Internet sobre dos ataques web muy extendidos, XSS y *SQL Injection*.

## 5.13. Protocolo HTTPS

HTTPS (*Hyper Text Transfer Protocol Secure*) es un protocolo que utiliza SSL (*Secure Sockets Layer*) o en TLS (*Transport Layer Security*) para encapsular mensajes HTTP. Gracias a la utilización de algoritmos criptográficos y certificados digitales se puede garantizar, la confidencialidad y la integridad de la información transmitida, así como la autenticidad de los servidores.

- Los clientes utilizan <https://> en las URIs (o URLs).
- Los servidores web por defecto escuchan peticiones HTTPS en el puerto 443/TCP.

En la actualidad todos los sitios web que manejen información personal y/o confidencial de los usuarios (correo electrónico, redes sociales, bancos, comercio electrónico...) deberían usar HTTPS.

**¿Sabías que...?** Existen varios "mitos" o información errónea y/o incompleta sobre HTTPS que pueden hacer que los usuarios se relajen cuando acceden a sitios web. Por ejemplo "si un sitio web usa HTTP es completamente seguro" o "si se muestra un candado en el navegador estoy accediendo y enviado mis datos a un sitio seguro". Observa la Figura 5.17 y haz la siguiente actividad para ampliar información sobre estos y otros "mitos" de HTTPS.

#### ◦ Actividad 5.15:

- Busca información en Internet sobre "mitos" HTTPS.
- Busca información en Internet sobre *SSL Trip*.

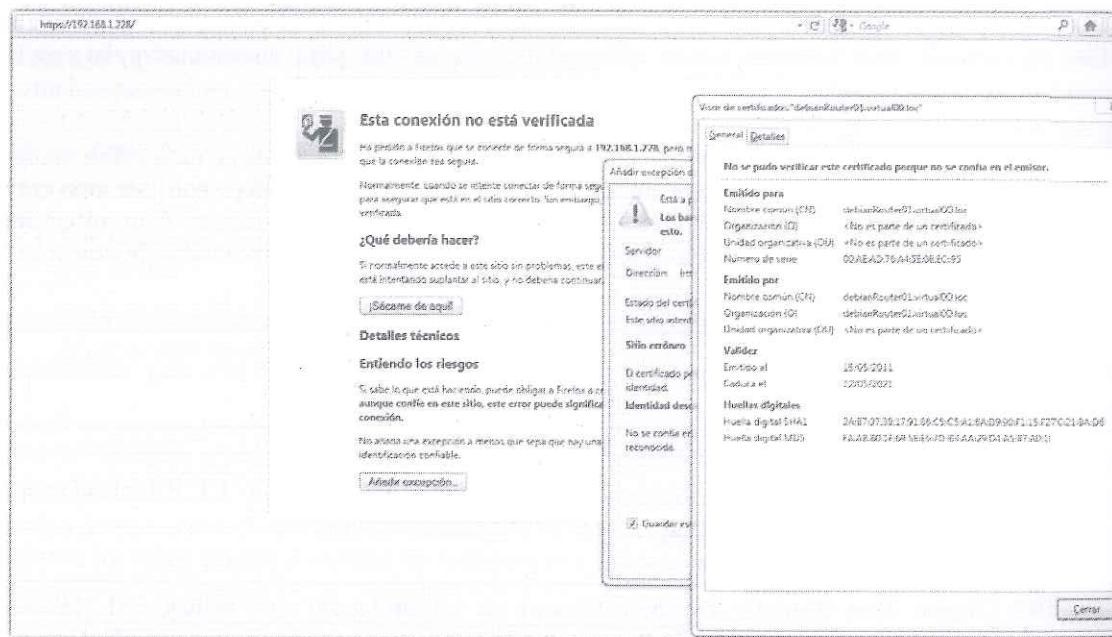


Figura 5.17: HTTPS. Certificado digital no verificado por el navegador

## 5.14. Alojamiento virtual de sitios web

El alojamiento virtual de sitios web (*web virtual hosting*) consiste en simular que existen varias máquinas (*hosts*) con sus respectivos sitios web sobre un solo servidor web, es decir, alojar varios sitios web (por ejemplo [www.asir.es](http://www.asir.es) y [www.daw.es](http://www.daw.es)) en un mismo servidor. También se usan los términos *hosts* virtuales, servidores virtuales, y sitios virtuales para referirse a este tipo de configuración. El uso de servidores web virtuales permite reducir el número de máquinas físicas

necesarias para alojar los millones de sitios web que existen en Internet y al mismo tiempo aprovechar mejor los recursos (uso de CPU, memoria...) de los equipos.

La mayoría de los servidores web actuales (*Apache*, *IIS*...) soportan estas funcionalidades.

Se pueden diferenciar tres tipos de alojamiento virtual que explicamos a continuación.

### 5.14.1. Alojamiento virtual basado en IPs

El servidor tendrá diferentes direcciones IP por cada servidor web virtual. Cada servidor virtual atenderá peticiones en una dirección IP diferente. A efectos de los usuarios es como si existiesen varios servidores web, uno en cada dirección IP.

Para realizar esta configuración la máquina, donde se ejecuta el servidor web, debe tener varias tarjetas de red configuradas cada una con una dirección IP o hay que asignar varias direcciones IP (alias o interfaces virtuales) a una misma tarjeta de red (la mayoría de los sistemas operativos actuales soportan esta funcionalidad).

#### ◦ Actividad 5.16:

- Busca información sobre el comando *ipconfig* o sobre el fichero */etc/network/interfaces* y configura la tarjeta de red *debianXX* con dos direcciones IPv4.
- Pincha en las opciones avanzadas de las propiedades de Internet v4 de la tarjeta de red de *w7XX* y configura dos direcciones IPv4.

Un ejemplo de configuración sería el siguiente, mostrado en la Figura 5.18:

- Equipo con una tarjeta de red.
- Dos direcciones IP configuradas en la tarjeta (193.100.200.101 y 193.100.200.101).
- Se configura el servidor web para que responda de forma distinta a cada IP:
  - <http://193.100.200.101>
  - <http://193.100.200.201>
- Si se configura un servidor DNS:
  - [www.asir.es](http://www.asir.es) IN A 193.100.200.101
  - [www.daw.es](http://www.daw.es) IN A 193.100.200.201
- Se podrán realizar las peticiones:
  - <http://www.asir.es>
  - <http://www.daw.es>
- Para el usuario existen dos servidores web distintos pero realmente es un servidor web con dos servidores virtuales.

Comparado con el alojamiento virtual basado en nombres, que se explica a continuación, supone un derroche de direcciones IP. Son necesarias tantas direcciones IP como servidores web virtuales.

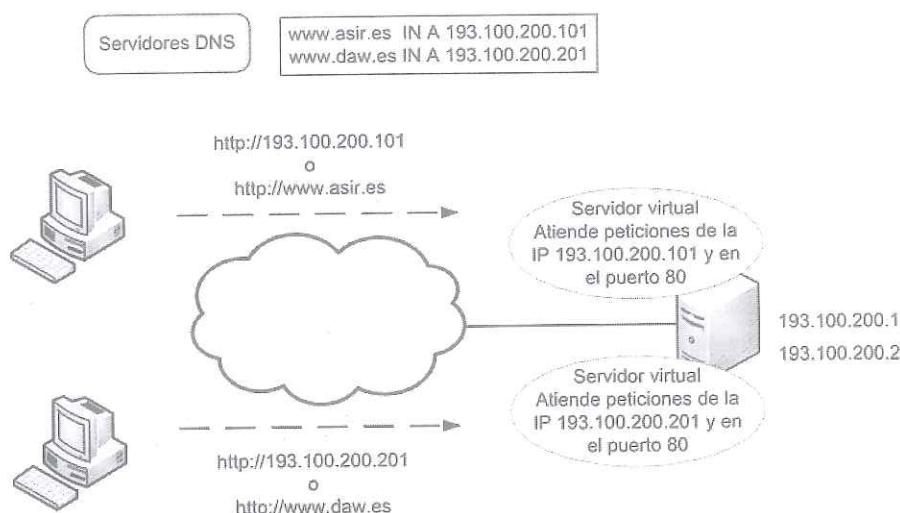


Figura 5.18: Alojamiento virtual basado en IPs

#### 5.14.2. Alojamiento virtual basado en nombres

El servidor permite alojar varios nombres de dominio (por ejemplo `www.asir.es` y `www.daw.es`) sobre la misma dirección IP. Cada servidor virtual atiende las peticiones de un nombre de dominio.

Hay que configurar un servidor DNS que asocie los nombres de dominio con la misma dirección IP.

Un ejemplo de configuración sería el siguiente, mostrado en la Figura 5.19:

- Equipo con una tarjeta de red.
- Una dirección IP configurada en la tarjeta (192.168.200.101).
- Se configura un servidor DNS
  - `www.asir.es` IN A 193.100.200.101
  - `www.daw.es` IN A 193.100.200.101
- Se configura el servidor para que responda a las peticiones
  - `http://www.asir.es`
  - `http://www.daw.es`
- Para el usuario existen dos servidores web distintos pero realmente es un servidor web con dos servidores virtuales.

Este tipo de alojamiento se definió a partir de la versión HTTP/1.1. El servidor web es capaz de diferenciar el nombre de dominio al que se dirige el cliente, ya que en el mensaje de petición HTTP se envía la cabecera `Host` con el nombre de dominio, Figura 5.20.

Es la forma de alojamiento más utilizada. Al permitir alojar varios dominios en un único equipo y dirección IP, se ahorra tanto en número de equipos como en direcciones IP. Además simplifica y facilita la administración centralizada de los servidores.

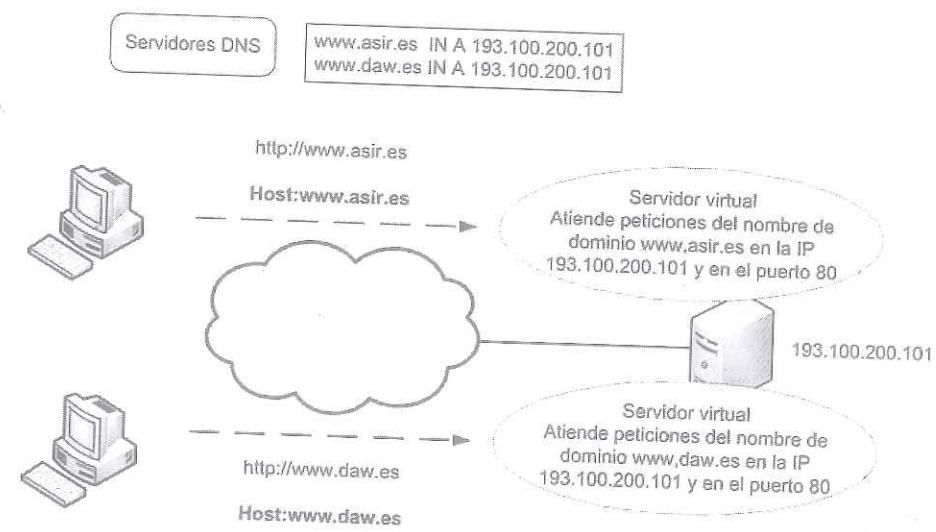


Figura 5.19: Alojamiento virtual basado en nombres

```

GET / HTTP/1.1
Host: www.todofp.es
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:2.0.1) Gecko/20100101
Firefox/4.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Keep-Alive: 115
Connection: keep-alive
Cookie: _utma=41836455.1669009742.1292858328.1303907907.1305452850.5;
_utmr=41836455.1305452850.5.5._utmcst=iesgranacapitan.org!utmccn=(referral)!utm
cmd=referral!utmccr=/blog04
  
```

Cabecera Host. Usada por el servidor web para saber qué servidor virtual (si está configurado) tiene que atender la petición

Figura 5.20: Cabecera Host

**¿Sabías que ... ?** La mayoría de los planes que nos ofrecen las empresas de alojamiento (*hosting*) en Internet se basan en alojamiento virtual basado en nombres. Nos dan espacio y recursos en un servidor web virtual que asocian con nuestro nombre de dominio. En una misma IP se pueden alojar cientos de dominios.

◊ **Actividad 5.17:** En Internet existen múltiples webs en las que introduciendo una dirección IP o nombre de dominio se muestran, respectivamente, información de los dominios alojados en esa IP o información de otros dominios alojados en la misma IP que el dominio introducido. Algunos ejemplos de webs que ofrecen esta funcionalidad son <http://www.myipneighbors.net> y [www.robtex.com](http://www.robtex.com). Accede a esas páginas y busca información sobre nombres de los dominios que consideres.

### 5.14.3. Alojamiento virtual basado en puertos

Cada servidor virtual atiende peticiones en una dirección IP y/o dominio:puerto diferentes. Consiste en combinar el alojamiento basado en IP y/o en nombres con el uso de varios puertos a las escucha.

Un ejemplo de configuración sería el que se detalla a continuación y se muestra en la Figura 5.21.

- Equipo con una tarjeta de red.
- Una dirección IP configurada en la tarjeta (193.100.200.101).
- Se configura un servidor DNS.
  - www.asir.es IN A 193.100.200.101
  - www.daw.es IN A 193.100.200.101
- Se configura el servidor para que responda a las peticiones
  - http://www.asir.es:80
  - http://www.asir.es:8080
  - http://www.daw.es:80
- Para el usuario existen tres servidores web distintos pero realmente es un servidor web con tres servidores virtuales.

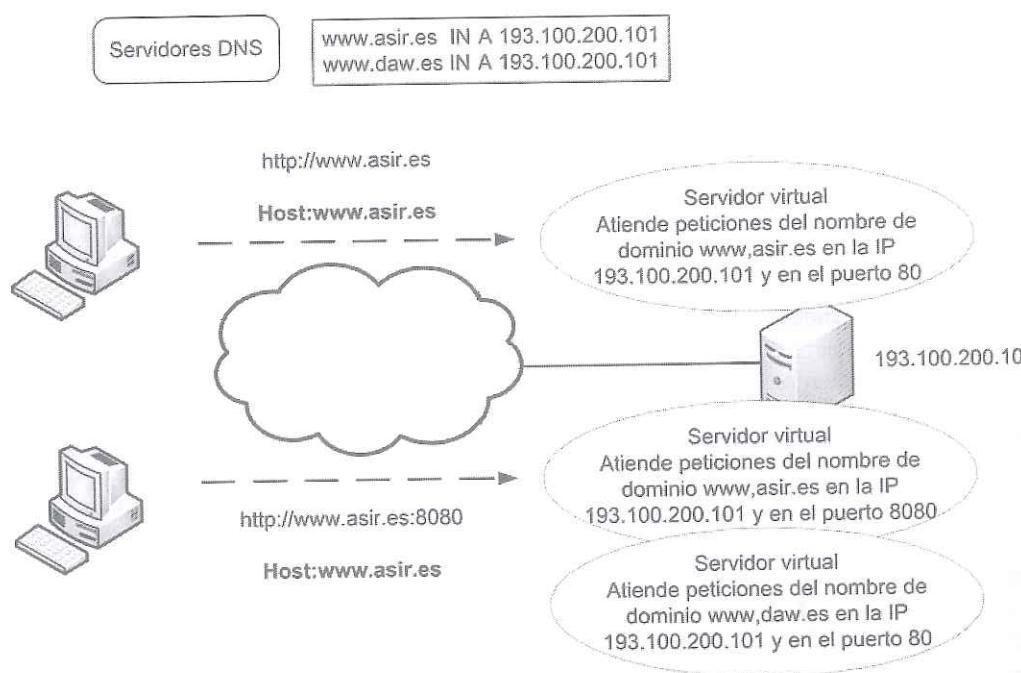


Figura 5.21: Alojamiento virtual basado en puertos

### 5.14.4. Combinaciones

En un mismo servidor web se pueden combinar servidores virtuales basados en IP, en nombres y en puertos.

- ◊ **Actividad 5.18:** El servidor web Apache admite la creación de todos los tipos de servidores virtuales explicados. Consulta el siguiente enlace <http://httpd.apache.org/docs/2.4/es/vhosts/examples.html> y observa varios ejemplos de configuración del servidor. Aunque no entiendas todas las opciones, puedes hacerte una idea de cómo se configuran servidores virtuales en Apache y relacionarlo con las explicaciones anteriores.

