IES N1 Gijón

Principado de Asturias
Consejería de Educación

UNIDAD DE TRABAJO 4

Servicios Web

Celia C. Pellico Llera

Cofinanciado por la Unión Europea

Contenidos

1. Introducción
2. Conceptos básicos
3. Funcionamiento de la web
4. URL
5. Páginas web, sitios web y aplicaciones web
6. Plataformas libres y propietarias
7. Protocolo HTTP
8. El estándar MIME
9.Seguridad
10. Alojamiento virtual de sitios web

1. Introducción

Una **página web** no es más que un conjunto de ficheros relacionados entre sí a través de enlaces o **hipervínculos**.

Para que esta página web pueda ser accedida por cualquier equipo conectado a una red, estos ficheros han de permanecer guardados en un ordenador que cumpla el rol de **servidor web**.

Un proveedor de **alojamiento web** es una empresa que alquila espacio web y ancho de banda para la publicación de sitios web. Se trata un ordenador remoto donde se guardan los archivos que forman parte de la web. La comunicación entre el ordenador donde se crea la página web con el servidor web se realiza frecuentemente a través de **FTP**.

Para decidir que tipo de alojamiento web se necesita, es necesario saber los requerimientos de la página web. No es lo mismo alojar una página web personal que la página web de una gran empresa. En este último caso, el volumen de información y la cantidad de accesos al servidor será mucho mayor. Por tanto, el servidor que maneje dicha información deberá disponer de más recursos.



¿Qué es un hosting o alojamiento web? Tipos de hosting

Existen diferentes tipos de alojamiento web.

- **Alojamiento compartido gratuito**: normalmente dispone de recursos limitados, poca funcionalidad y estabilidad. Generalmente agregan publicidad.
- **Alojamiento compartido**: es el más frecuente. En este caso el sitio web está en un servidor que aloja muchos otros sitios web lo que reduce el coste.
- **Servidor dedicado**: un solo cliente dispone de todos los recursos de la máquina. El cuidado y conectividad de la máquina está a cargo del proveedor.
- **Servidor virtual (Virtual Private Server)**: la empresa ofrece un sitio dedicado pero a través de una máquina virtual.
- **Alojamiento en la nube (Cloud hosting):** está basado en las tecnologías que permiten a un gran número de máquinas actuar como un sistema conectadas a un grupo de medios de almacenamiento. Tiene ventajas como el uso de recursos y que la seguridad de un sitio web alojado en la "nube" está garantizada por numerosos servidores en lugar de solamente uno



2. Conceptos básicos



La **WWW**, o la web, se compone de un sistema de distribución de información basado en hipervínculos que permiten su visualización en un ordenador conectado a la red **Internet.**
Con un navegador, un usuario o usuaria visualiza sitios web compuestos de páginas web y navega a través de ellas usando **hiperenlaces**.

Sitio web: conjunto de páginas web relacionadas y accesibles a partir de un mismo nombre de dominio DNS.

Servidor 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. Por defecto escuchan peticiones HTTP por el puerto **80/TCP**.

Cliente: cualquier aplicación que sirve para solicitar páginas web, como por ejemplo, los **navegadores**.

Navegador web: software instalado en el ordenador, que visualiza documentos en formato web (HTML) y permite el funcionamiento de los hiperenlaces. Los navegadores más utilizados son Mozilla Firefox, Chrome, Opera e Internet Explorer.

Aplicación web es aquella donde el usuario interactúa con un navegador que accede a los servicios y recursos que ofrece un servidor web. Ejemplos pueden ser un buscador, un cliente de correo, etc.

URL: Uniform Resource Location (Localizador de recurso uniforme). Es la dirección textual de un servidor web para la apertura de un archivo HTML.

IP: Internet Protocol (Protocolo de Internet). Es un código identificativo de un equipo informático en Internet formado por cuatro números, separados por puntos, cuyo valor no puede exceder de **255**.

DNS: Domain Name System (Sistema de nombres de dominio). Es una base de datos que relaciona una dirección textual de un servidor con su dirección IP. Esta base de datos se encuentra en unos servidores DNS que deben ser especificados en nuestra conexión a Internet.

HTTP: Hypertext Transfer Protocol (Protocolo de transferencia de hipertexto). Al encabezar una URL con este término, el servidor reconocerá que se le está asociando un documento web.

HTML: Hypertext Markup Language (Lenguaje de marcado de hipertextos). Los archivos creados en este formato tienen extensiones ***html*** o ***htm***.

Proxies web: o proxies http son programas intermediarios entre clientes y servidores web. Pueden actuar como cortafuegos y/o almacenar datos en ***cache*** para aumentar el rendimiento.

Tecnologías web: Son las utilizadas para desarrollar aplicaciones basadas en la web (*XHTML, Ajax, XQuery*, …)

Un poco de historia

HTTP fue desarrollado en 1989 por **Tim Berners-Lee** como parte del proyecto de la World Wide Web en el **CERN**. Inicialmente, HTTP funcionaba en su versión 0.9 con peticiones simples de solo texto.

Evolución

HTTP/1.0 introdujo cabeceras para mejorar la comunicación. HTTP/1.1 optimizó las conexiones persistentes y HTTP/2 mejoró la eficiencia con multiplexación, que permite enviar varias peticiones en paralelo, y la compresión de cabeceras. La versión más reciente, HTTP/3, utiliza el protocolo **QUIC** para reducir latencias y mejorar el rendimiento y la seguridad en redes modernas.

3. Funcionamiento de la web

Vamos a ver los pasos desde que el cliente solicita una página web en la barra de direcciones del navegador hasta que ésta se visualiza en la ventana del mismo navegador.

1.- Desde el ordenador ejecutamos el programa navegador de Internet y escribimos la dirección de una página web. La dirección que se inserta es la URL (por ejemplo, http://www.google.es).

2.- El servidor DNS que tengamos configurado en la conexión a Internet localiza la dirección IP que corresponde al ordenador servidor donde se encuentran los archivos web requeridos.

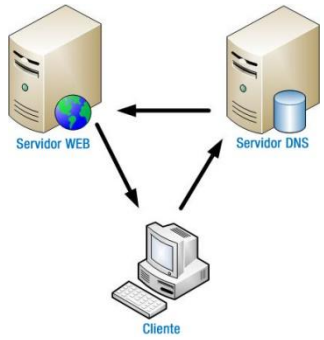
3.- Como la URL que hemos tecleado comienza por **http://**, el ordenador servidor reconoce que estamos solicitando algún documento web y nos envía un documento con extensión HTML o HTM.

Si en la dirección URL no se especifica un documento html concreto, el servidor buscará en la carpeta de la web y enviará por defecto el archivo nombrado como ***index.html****.*

4.- El navegador estudia el documento *html* recibido y solicita al servidor los archivos con los que esté enlazado, como imágenes, vídeos, animaciones, etc.

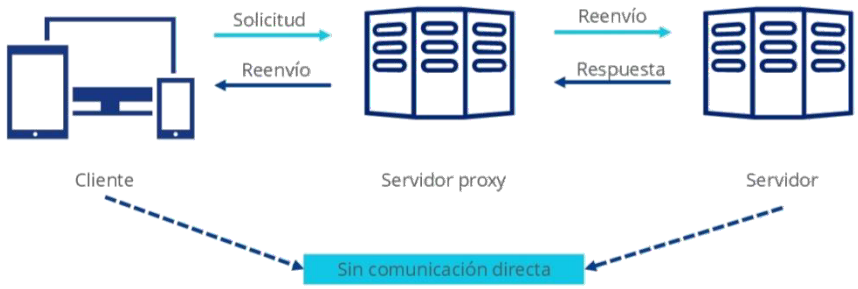
5.- Finalmente, el programa navegador compone la página (***renderiza*** el contenido) con los textos, enlaces, imágenes y animaciones.

Es importante que al elaborar una página web se compruebe si se visualiza correctamente en los distintos navegadores, ya que puede ocurrir que no se vea de igual manera en todos



Servicios Proxy

Un Proxy es un servidor —programa o dispositivo—, que hace de intermediario en las peticiones de recursos que realiza un cliente (**A**) a otro servidor (**C**). Por ejemplo, si una hipotética máquina A solicita un recurso a C, lo hará mediante una petición a **B** (el proxy), que a su vez trasladará la petición a C; de esta forma C no sabrá que la petición procedió originalmente de A.



IONOS

Esta situación estratégica de punto intermedio le permite ofrecer diversas funcionalidades como control de acceso, registro del tráfico, restricción a determinados tipos de tráfico, mejora de rendimiento, anonimato de la comunicación, caché web, etc.



Proxy ¿que es..?

Un servidor proxy gestiona las comunicaciones a nivel de transporte (nivel 4), de un mismo protocolo, realizando la tarea de petición del tráfico de red de ese protocolo en nombre de sus clientes. Por ejemplo, un proxy-caché de acceso a la www, recibe peticiones de tráfico de clientes web y las realiza en su nombre, copiando las respuestas en una memoria caché.
También es habitual que los servidores proxy puedan solicitar autenticación del usuario, así como filtrado de contenidos o servicio antivirus, anti-spam, etc.

4. URL

El acrónimo **URL** (del inglés ***U**niform **R**esource **L**ocator*) hace referencia al identificador único de cada recurso disponible en Internet.

Las URL son esenciales para crear los enlaces, pero también se utilizan en otros elementos HTML como las imágenes y los formularios.

Toda la parte de la URL que hace referencia a la máquina o host o no distingue mayúsculas o minúsculas. Pero, lo que corresponde a la ruta de directorios y nombre de archivo dentro de un ordenador, dependerá del sistema operativo en el que trabaje ese ordenador, y la mayoría de ellos distingue entre mayúsculas y minúsculas.

La URL de un recurso tiene dos objetivos principales:

- **Identificar de forma única a ese recurso**. Las URL permiten que cada página HTML publicada en Internet tenga un **nombre único** que permita diferenciarla de las demás. De esta forma es posible crear enlaces que apunten de forma inequívoca a una determinada página.
  Por ejemplo, la cadena de texto ***http://www.google.com*** (la dirección que muestra el navegador) es la URL completa de la página principal de Google. La URL de las páginas es imprescindible para crear los enlaces, ya que permite distinguir una página de otra.

- **Permitir localizar de forma eficiente ese recurso**. Para ello es necesario comprender las diferentes partes que forman las URL. Una URL sencilla siempre está formada por las mismas tres partes.

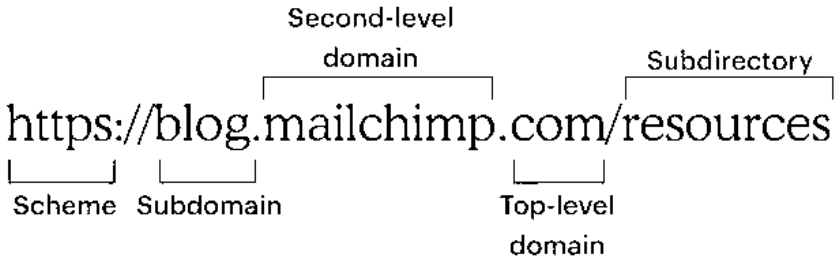
Si por ejemplo se considera la siguiente URL:

http://www.librosweb.es/xhtml/capitulo4.html

Las partes que componen la URL anterior son:

- **Protocolo** (*http://*): el mecanismo que debe utilizar el navegador para acceder a ese recurso. Todas las páginas web utilizan http://. Las páginas web *seguras* (por ejemplo, las de los bancos y las de los servicios de email) utilizan **https://** (se añade la letra **s**).
- **Servidor** (*www.librosweb.es*): simplificando mucho su explicación, se trata del ordenador en el que se encuentra guardada la página que se quiere acceder. Los navegadores son capaces de obtener la dirección de cada servidor a partir de su nombre.
- **Ruta** *(/xhtml/capitulo4.html*): *camino* que se debe seguir, una vez que se ha llegado al servidor, para localizar el recurso específico que se quiere acceder.

Por tanto, las URL no sólo identifican de forma única a cada recurso de Internet, sino que también proporcionan a los navegadores la información necesaria para poder llegar hasta ese recurso.



La mayoría de URL son tan sencillas como la URL mostrada anteriormente. No obstante, existen URL complejas formadas por más partes.

http://www.alistapart.com/comments/webstandards2008?page=5#42

Las cinco partes que forman esta URL son:

- **Protocolo** (*http://*)
- **Servidor** (*www.alistapart.com*)
- **Ruta** (*/comments/webstandards2008*)
- **Consulta** (*?page=5*): información adicional necesaria para que el servidor localice correctamente el recurso que se quiere acceder. Siempre comienza con el carácter **?** y contiene una sucesión de palabras separadas por **=** y **&**
- **Sección** (*#42*): permite que el navegador se posicione automáticamente en una sección de la página web. Siempre comienza con el carácter #.

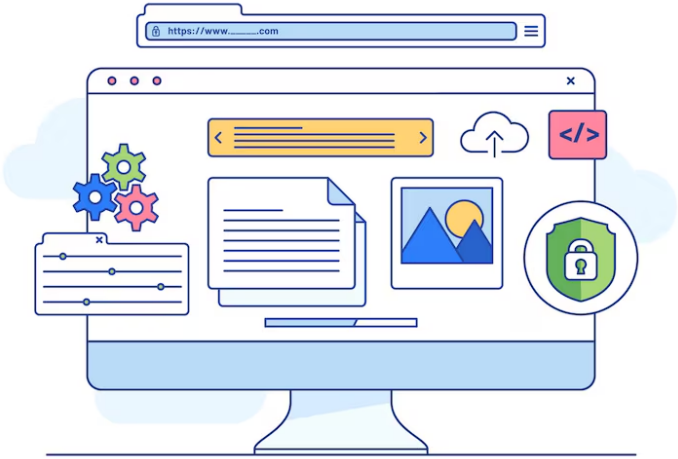
Como las URL utilizan los caracteres **:**, **=**, **&** y **/** para separar sus partes, estos caracteres están reservados y no se pueden utilizar libremente. Además, algunos caracteres no están reservados, pero pueden ser problemáticos si se utilizan en la propia URL.
Si es necesario incluir estos caracteres reservados y especiales en una URL, se sustituyen por combinaciones de caracteres seguros. Esta sustitución se denomina *codificación* de caracteres y el servidor realiza el proceso inverso (*decodificación*) cuando le llega una URL con los caracteres codificados.

El puerto normalmente no se especifica en la URL, el navegador hace la conversión por los puertos estándar de cada servicio, por ejemplo, HTTP es el 80, FTP es el 21, HTTPS es el 443, etc.

5. Páginas web, sitios web y aplicaciones web

Página web

Es un documento que normalmente contiene elementos como texto, audio, imágenes, etc. además de hiperenlaces a otras páginas web o recursos. Su contenido puede ser estático o dinámico. Están escritas en lenguajes que son interpretados por los navegadores (HTML, CSS, XML, JavaScript…)



Sitio web

Es el conjunto de páginas web relacionadas y accesibles a partir de un mismo nombre de dominio. A veces por extensión, se dice “página web” cuando en realidad nos estamos refiriendo al “sitio web”.

Aplicación Web

Es un programa que se codifica en un lenguaje interpretable por los navegadores web (PHP p.ej.) en el que se confía la ejecución al navegador.

¿Qué es un servidor Web?

Es un sistema de software que permite a un ordenador (al que también se denomina *servidor web*) alojar páginas web. Los servidores web entregan contenido estático (páginas html, imágenes, vídeos…) en respuesta a las solicitudes del protocolo http de un navegador web o una aplicación móvil. Según la configuración, un servidor web puede almacenar uno o más sitios web.

Se ejecuta de forma continua manteniéndose a la espera de peticiones por parte de un cliente (un navegador) y responde sirviendo una página web que será mostrada en el navegador o mostrando el mensaje correspondiente si se detectó algún error.

¿Qué es un servidor de Aplicaciones?

Es un servidor que aloja aplicaciones web, programas informáticos que se ejecutan dentro de un navegador. La función principal de un servidor de aplicaciones es proporcionar a los clientes acceso a la lógica de negocio, que es la parte de un programa informático que codifica las reglas para que el programa pueda crear, almacenar y manipular sus datos.

En algunos casos, las funciones y operaciones de los servidores web y de aplicaciones pueden solaparse.

Principales Servidores Web

Apache HTTP Server



- Tradicionalmente ha sido el servidor más utilizado.
- Es de fácil configuración y personalización.
- Es multiplataforma y su estructura modular permite emplear diversos lenguajes en el lado del servidor.
- Incorpora características para la compresión de datos, las conexiones seguras y la utilización de URLs amigables.
- El coste reducido es otro factor muy importante de su éxito.
- Actualizaciones y parches de seguridad regulares.
- Gran comunidad de desarrolladores.
- Compatibilidad con IPv6.

Microsoft Internet Information Services (IIS)



- Es el tercer servidor web más usado.
- Cuenta con un buen número de módulos.
- Cuenta con la limitación de funcionar únicamente en Windows.
- Ha conseguido solucionar la mala imagen dada por sus frecuentes vulnerabilidades.

NGINX (pronunciado engine X)



- Se ha convertido en el primero de la lista.
- Es un servidor Open Source ligero y muy rápido.
- Se ha hecho muy popular por su gran capacidad de carga y rendimiento con un coste de recursos muy bajo.
- Lo usan sitios tan famosos como Netflix o Hulu.
- Alto rendimiento
- Bajo consumo de memoria
- Compatibilidad con IPv6.

GWS, GOOGLE WEB SERVER



- Conocido como GWS es una gran incógnita.
- Se sabe que sobre él corren todas las infraestructuras web de Google.
- Google no publica apenas información sobre él y no se comercializa a nivel popular.

Principales Servidores de Aplicaciones

Como consecuencia del éxito del lenguaje de programación Java, el término servidor de aplicaciones usualmente hace referencia a un servidor de aplicaciones **Java EE**.

Entre los servidores de aplicación Java EE **privativos** más conocidos se encuentran:

- **WebLogic** de ***Oracle***
- **WebSphere** de ***IBM***.
- **Microsoft IIS** para el despliegue de aplicaciones basadas en tecnologías de Microsoft, como **.NET.**

Entre los servidores de aplicaciones **libres** se encuentran, entre otros:

- **WildFly**, anteriormente conocido como **JBoss Application Server,** al estar basado en Java, puede ser utilizado en cualquier sistema operativo para el que esté disponible la máquina virtual de Java. Es **software libre y de código abierto**. El proyecto se nutre de una red mundial de colaboradores.
- **GlassFish** es software de código abierto desarrollado por Sun Microsystems para la plataforma Java EE.
- **JOnAs** es un software libre indicado para servidores de aplicaciones desarrollado por el consorcio ObjectWeb.
- **Apache Tomcat** funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages de Oracle Corporation



6. Plataformas web libres y propietarias

Una plataforma web es el entorno de desarrollo de software empleado para diseñar y ejecutar un sitio web.

En términos generales, una plataforma web consta de cuatro componentes básicos:

1. El **sistema operativo,** bajo el cual opera el equipo donde se hospedan las páginas web y que representa la base misma del funcionamiento del computador. En ocasiones limita la elección de otros componentes.
2. El **servidor web** es el software que maneja las peticiones desde equipos remotos a través de la Internet. En el caso de páginas estáticas, el servidor web simplemente provee el archivo solicitado, el cual se muestra en el navegador. En el caso de sitios dinámicos, el servidor web se encarga de pasar las solicitudes a otros programas que puedan gestionarlas adecuadamente.
3. El **gestor de bases de datos** se encarga de almacenar sistemáticamente un conjunto de registros de datos relacionados para ser usados posteriormente.
4. **Un lenguaje de programación interpretado** que controla las aplicaciones de software que corren en el sitio web.

Diferentes combinaciones de los cuatro componentes señalados, basadas en las distintas opciones de software disponibles en el mercado, dan lugar a numerosas plataformas web, aunque, sin duda, hay dos que sobresalen del resto por su popularidad y difusión: **LAMP** y **WISA**.

PLATAFORMA LAMP

Trabaja enteramente con componentes de **software libre** y no está sujeta a restricciones propietarias. El nombre **LAMP** surge de las iniciales de los componentes de software que la integran: **L**inux, **A**pache, **M**ySQL y **P**HP (a veces se sustituye por ***Perl*** o ***Python***).

PLATAFORMA WISA

Está basada en tecnologías desarrolladas por la compañía Microsoft; se trata, por lo tanto, de **software propietario.** La componen los siguientes elementos: **W**indows, **I**nternet Information Services (IIS), **S**QL Server y **A**SP o ASP.NET

OTRAS SON:

WAMP: Usa las siguientes herramientas: **W**indows, **A**pache, **M**ySQL y **P**HP (generalmente), **P**erl, o **P**ython.

MAMP: La componen los siguientes elementos: **M**ac OS X, **A**pache, **M**ySQL y **P**HP, **P**erl o **P**ython.

XAMPP

Es un servidor independiente de plataforma. El programa está liberado bajo la licencia **GNU** y actúa como un servidor web libre, fácil de usar y capaz de interpretar páginas dinámicas. Está compuesto por:

- **X**: Cualquier Sistema operativo.
- **Apache:** Servidor web.
- **M**ySQL/MariaDB: Gestor de bases de datos.
- **P**HP y **P**erl: Lenguaje para scripting del lado del servidor

7. Protocolo HTTP

HTTP (HyperText Transfer Protocol) es el protocolo de comunicación en la web. Define las reglas que utilizan los componentes software (clientes, servidores, proxies) para comunicarse. Fue desarrollado por el **W**orld **W**ide **W**eb **C**onsortium y la Internet Engineering Task Force.



Es un protocolo orientado a transacciones que sigue el esquema petición-respuesta entre un **cliente** y un **servidor**.
Al cliente que efectúa la petición (un navegador web p.ej.) se le conoce como "***user agent***" (agente del usuario).

A la información transmitida se la llama ***recurso*** y es identificada mediante un localizador uniforme de recursos (URL). El resultado de la petición es la ejecución de un programa, una consulta a una base de datos, la traducción automática de un documento, etc.

Los recursos "*viven*" en los servidores web que "*hablan*" el protocolo HTTP y cuando un cliente pide parte de los recursos al servidor web, ***request***, éste se los devuelve en respuestas ***HTTP Response***.



Estos recursos pueden ser archivos **estáticos** que se almacenan en los discos duros, como archivos de **texto**, **html**, **pdf**, etc.
Sin embargo este contenido puede ser también **dinámico**, generado por programas. Por ejemplo, si hacemos compras on-line consultamos productos, envíos, etc.

Funcionamiento

HTTP es un protocolo **sin estado** (no guarda ninguna información sobre conexiones anteriores) 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.

Es un protocolo **cliente-servidor** cuyo funcionamiento está basado en el envío de *mensajes*.

1. El usuario especifica en el cliente web (o navegador) la dirección de la página que quiere consultar según un formato de URL válido

2. El cliente web descodifica la información de la URL diferenciando el protocolo de acceso, la IP o nombre de dominio del servidor, el puerto, etc.

3. El cliente web establece una conexión (**socket**) con el servidor web y solicita la página (mensaje request).

4. El servidor envía dicha página (si no existe, envía un código de error) y el cliente web interpreta el código HTML recibido (mensaje respuesta del servidor). En ambos casos, y al tratarse de un protocolo sin estado, la conexión se libera al finalizar . Solo dura el tiempo necesario para la transmisión del documento solicitado.

Para cada objeto que se transfiere por la red, se realiza una conexión independiente. Para resolver estas situaciones de «*falta de memoria*», además de almacenarse la información de la sesión actual se utilizan las ***cookies***.



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:

- Mensajes de **petición**.
- Mensajes de **respuesta**.

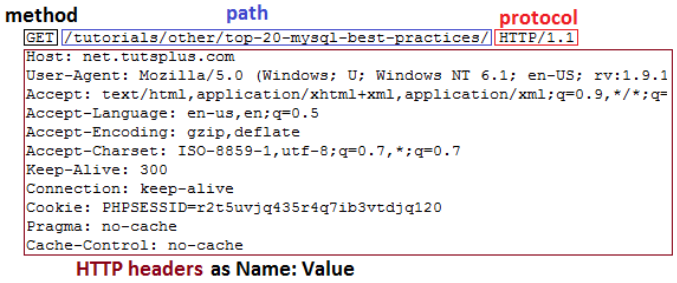
Los **mensajes de petición** están compuestos por tres partes:

- **Línea inicial de petición**, que incluye el **método** utilizado, la parte de la URL relativa al servidor y la versión del protocolo utilizada.
- **Líneas de cabecera**, que incluye un conjunto de pares ***nombre/valor*** denominados ***cabeceras***. Si no hay cabeceras se envía un **0**. Determinan cómo será procesada la petición por parte del servidor.
- **Cuerpo del mensaje**.

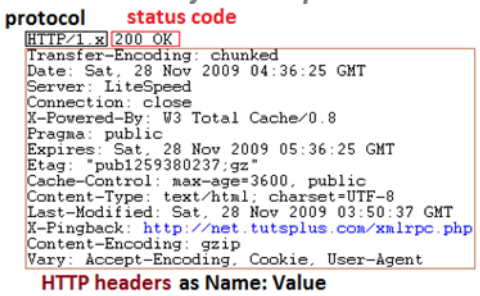
Los **mensajes de respuesta** están compuestos por tres partes:

- **Línea inicial de respuesta**, que incluye la versión HTTP utilizada, un código de estado o código de error y un texto explicativo del código de estado.
- **Líneas de cabecera**, que incluye un conjunto de pares ***nombre/valor*** denominados ***cabeceras***. Si no hay cabeceras se envía un **0**. Describen los datos y la forma en que son enviados al cliente.
- **Cuerpo del mensaje**.

Mensaje de petición



Mensaje de respuesta



Métodos HTTP

Los métodos de petición especifican la operación que quiere realizar el cliente en el servidor sobre el recurso solicitado.

Algunos de los más importantes son:

- **GET:** Solicita datos del servidor sin modificar el recurso.
- **POST:** Envía datos al servidor para ser procesados.
- **PUT:** Actualiza un recurso en el servidor o lo crea si no existe.
- **DELETE:** Elimina un recurso en el servidor.
- **HEAD:** Similar a GET, pero solo obtiene las cabeceras sin el contenido.
- **OPTIONS:** Permite conocer los métodos soportados por un servidor.
- **PATCH:** Modifica parcialmente un recurso en el servidor.

Generalmente los que más conocemos son los dos primeros por escribir formularios HTML.



Haz clic en la imagen para más información

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 un gran número de tipos de cabeceras definidas por HTTP y se pueden clasificar como:

- Generales.
- De petición (o de cliente)
- De respuesta (o de servidor)
- De entidad.

HTTP/1.1 define 46 cabeceras (la cabecera Host es obligatoria en las peticiones para permitir el uso de sitios o servidores virtuales).

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 una cabecera no estándar.

EJEMPLOS DE CAMPOS DE CABECERA

Request Headers (cabeceras de solicitud)

- **User-Agent**: Indica el navegador y sistema operativo del cliente.
- **Accept**: Define qué tipos de contenido puede aceptar el cliente.
- **Cookie**: Permite el envío de datos almacenados en el navegador.

Response Headers (cabeceras de respuesta):

- **Content-Type**: Indica el formato de los datos enviados (ej. text/html, application/json).
- **Set-Cookie**: Define cookies en el cliente.
- **Cache-Control**: Gestiona la caché del contenido.

Códigos de estado y error

Son 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 3 cifras que se clasifican en 5 grupos en función del primer dígito:

100 – **1**99 **Informativo** (*Informational*)

Indican que la solicitud se está procesando. Por ejemplo, ***100 Continue***.

200 – **2**99 **Éxito** (*Successful*)

La solicitud fue recibida y procesada correctamente. Por ejemplo, ***200 OK*** o ***201 Created***.

300 – **3**99 **Redirección** (*Redirection*)

Indican que el recurso ha cambiado de ubicación. Por ejemplo, ***301 Moved Permanently*** o ***302 Found***.

400 – **4**99 **Errores de cliente** (*Client Error*)

Indican que la solicitud es inválida o el recurso no existe. Por ejemplo, ***401 Unauthorized*** , ***403 Forbidden*** o ***404 Not Found***.

500 – **5**99 **Errores en el servidor** (*Server Error*)

Indican fallos en el servidor . Por ejemplo, ***500 Internal Server Error*** o ***503 Service Unavailable***.

Almacenamiento en caché (caching)

Los servidores web contienen información que no cambia durante periodos de tiempo muy largos. HTTP soporta almacenamiento en cache para aumentar el rendimiento y evitar tráfico innecesario ¿Por qué descargarse 15 veces la misma imagen si se accede 15 veces a ella el mismo día? Los navegadores y los *proxies* pueden almacenar localmente recursos durante un tiempo.

Se definen cabeceras (*Cache-Control, Last-Modified, Expires, Age, Etag, If-Match, IF-Modified-Since*,..) que permiten controlar qué se almacena en cache, cuánto tiempo, qué no se puede almacenar, si la información se puede "cachear" en un proxy o no, si hay que actualizar algo que está en cache porque se ha modificado, etc.

Redirecciones

HTTP permite a los servidores y a los *proxies* redirigir las peticiones a otras localizaciones. Algunas situaciones en las que puede ser útil son las siguientes:

- 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.

Compresión

Es posible que los servidores compriman 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.

Cookies

Como ya comentamos, HTTP es un **protocolo sin estado**, es decir, que no guarda ninguna información sobre conexiones anteriores. Sin embargo, el desarrollo de aplicaciones web necesita frecuentemente mantener estado. Para esto se usan las *cookies*, que es información que un servidor puede almacenar en el sistema cliente.

Las cookies son utilizadas por los servidores web para diferenciar usuarios y conexiones y actuar en consecuencia. Esto les permite a las aplicaciones web tener la noción de "***sesión***", y también permite rastrear usuarios ya que las cookies pueden guardarse en el cliente por tiempo indeterminado.

Una cookie es un fragmento de información que envía un servidor 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 donde indican el nombre, contenido o valor, fecha de expiración, ruta y dominio.



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*).

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

Autenticación

HTTP soporta el uso de mecanismos de autentificación 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 ***Authoritation***. Los navegadores muestran entonces al usuario un cuadro de dialogo para que se identifique.

Algunos mecanismos son:

- **Basic**: El cliente envía un usuario y una clave codificados con el algoritmo *base64*. Método no seguro.
- **Digest**: El cliente envía un usuario y una función *hash* (resumen) de la clave del 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 actuales, la autenticación se basa en el uso de formularios XHTML en los que se introduce usuario/clave y que son enviados al servidor, donde son trasladados por alguna aplicación escrita en un lenguaje como PHP, JSP, ASP, etc., o en el uso de certificados digitales y **HTTPS**.

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/1.1 se introducen las conexiones persistentes por defecto.

8. El estándar MIME

MIME (***M**ultipurpose **I**nternet **M**ail **E**xtensions)* son una serie de convenciones o especificaciones dirigidas al intercambio a través de Internet de todo tipo de archivos (texto, audio, vídeo, etc.) de forma transparente para el usuario.

Los servidores web ponen una etiqueta MIME a todo el contenido para que, cuando un navegador haga una petición a los recursos, éste sepa qué tipo de archivos son y con qué aplicaciones o *plugins* puede manipularlos. Inicialmente, fueron usadas en el correo electrónico pero en la actualidad se ha generalizado a otros servicios.

Entre otros aspectos definen:

- Una serie de tipos y subtipos 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 como 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-Enconding*. Como se codifican los datos binarios usando ASCII.
  - *Content-Type*. Tipo y subtip

Cada identificador de tipo MIME consta de dos partes:

- La categoría general a la que pertenece el archivo como, por ejemplo, "text".
- El identificador que detalla el tipo de archivo específico como, por ejemplo, "html".

Un identificador de tipo MIME "*text/html*", por ejemplo, indica que el archivo es una página web estándar.

Algunos ejemplos de tipos MIME son:

Extensión	Tipo de documento	Tipo de MIME
.avi	AVI: Audio Video Intercalado	video/x-msvideo
.css	Hojas de estilo (CSS)	text/css
.csv	Valores separados por coma (CSV)	text/csv
.doc	Microsoft Word	application/msword
.epub	Publicación Electrónica (EPUB)	application/epub+zip
.gif	Graphics Interchange Format (GIF)	image/gif
.htm/.html	Hipertexto (HTML)	text/html
.ico	Formato Icon	image/x-icon
.jar	Archivo Java (JAR)	application/java-archive
.jpeg/.jpg	Imágenes JPEG	image/jpeg
.js	JavaScript (ECMAScript)	application/javascript
.json	Formato JSON	application/json
.mp3	Audio mp3	audio/mpeg
.mpeg	Video MPEG	video/mpeg
.pdf	Adobe Portable Document Format (PDF)	application/pdf
.ppt	Microsoft PowerPoint	application/vnd.ms-powerpoint
.rar	Archivo RAR	application/x-rar-compressed
.rtf	Formato de Texto Enriquecido (RTF)	application/rtf
.svg	Gráficos Vectoriales (SVG)	image/svg+xml
.wav	Formato de audio de forma de onda (WAV)	audio/x-wav
.xhtml	XHTML	application/xhtml+xml
.xls	Microsoft Excel	application/vnd.ms-excel
.xml	XML	application/xml
.xul	XUL	application/vnd.mozilla.xul+xml
.zip	Archivo ZIP	application/zip

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 que 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***.

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

- **El servidor** debe estar capacitado y habilitado para manejar diversos tipos MIME.
- En el código de **la página web** se referencia tipos MIME constantemente en etiquetas *link*, *script*, *object*, *form*, *meta*,… Por ejemplo:
  - El enlace a un archivo hoja de estilo CSS:
    <link href="./miarchivo.css" rel="stylesheet" type="text/css">
  - El enlace a un archivo código javascript:
    <script language="JavaScript" type="text/javascript" src="scripts/mijavascript.js">
  - Con las etiquetas meta podemos hacer que la página participe en el diálogo servidor/cliente, especificando datos MIME:
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

- **El navegador** del cliente también participa; además de estar capacitado para interpretar el tipo concreto MIME que el servidor le envía, también puede, en el diálogo previo al envío de datos, informar que tipos MIME puede aceptar la cabecera **http_accept**, así por ejemplo una cabecera http_accept tipo de un navegador sería:

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

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

¿Por qué son importantes los tipos correctos de MIME?

Si el servidor o la aplicación web indican un tipo incorrecto de MIME para el contenido, un navegador no tiene forma de saberlo, de acuerdo a la especificación HTTP, más si tenemos en cuenta que el autor especificó el contenido para ser procesado y mostrado en una forma diferente que la impuesta por el tipo MIME indicado.

Algunos navegadores de red intentan determinar el tipo de MIME adecuado en servidores mal configurados. Esto protege a muchos administradores de red de sus propios errores, pues el navegador continúa procesando el contenido aunque, por ejemplo, una imagen haya sido informada como texto plano.

Sin embargo, los navegadores no deberían suponer tipos MIME, pues aparte de la violación de la especificación HTTP, es una mala estrategia para los navegadores por dos razones:

- **Pérdida del control**: los administradores de red y los autores dejarán de tener el control sobre cómo sus contenidos serán procesados.
- **Seguridad**: algunos tipos de contenidos, como los programas ejecutables, son inherentemente inseguros, por lo que esos tipos MIME son generalmente restringidos en términos de qué acciones tomará el navegador de red al recibirlos.

9. Seguridad

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 objeto constante de atacantes.

El protocolo 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, suplantarle en webs (bancos, webmails, redes locales,..)

Protocolo HTTPS

El protocolo **HTTPS** permite que la información viaje de forma segura entre el cliente y el servidor.

HTTPS (**H**yper **T**ext **T**ransfer **P**rotocol **S**ecure) es un protocolo que utiliza **SSL** (**S**ecure **S**ockets **L**ayer) / **TLS** (**T**ransport **L**ayer **S**ecurity) para encapsular mensajes HTTP, cifrando la comunicación y asegurando la confidencialidad de los datos transmitidos.

Ventajas de HTTPS

- Los clientes utilizan **https:**// en las URIs (o URLs).
- Los servidores web por defecto escuchan peticiones **HTTPS** en el puerto **443/TCP**.
- Mayor seguridad y privacidad.
- Mayor confianza de los usuarios.
- Mejora el posicionamiento SEO en buscadores como Google.

En la actualidad todos los sitios web que manejan información personal y/o confidencial de los usuarios deberían usar HTTPS.

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.

10. Alojamiento virtual de sitios web

El alojamiento virtual de sitios Web (***Web virtual hosting***) consiste en simular que existen varias máquinas (Host) con sus respectivos sitios Web sobre un servidor Web, es decir, alojar varios sitios Web en un mismo servidor.

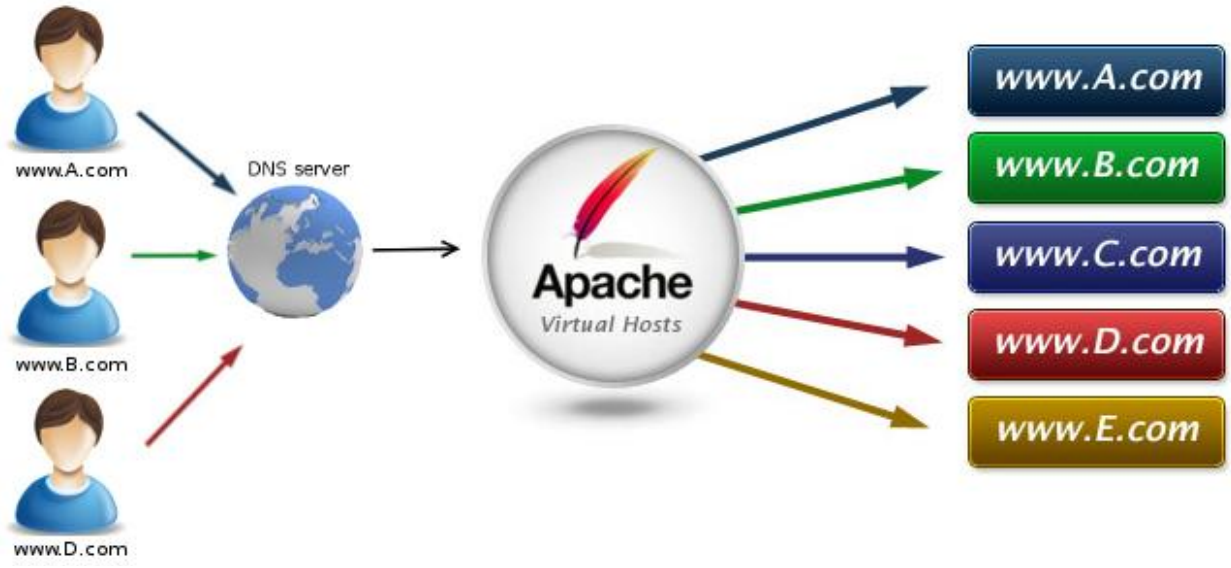
También se usan los términos *Host virtuales*, *servidores virtuales*, ***Virtualhost*** y *sitios virtuales* para referirse a este tipo de configuración.

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

Tener todo un servidor para atender sólo un sitio web es una pérdida de recursos. Un servidor es capaz de atender desde una sola máquina a todo un conjunto de sitios web. Es decir, podemos servir al mismo tiempo peticiones para *educastur.es*, *www.apache.org*, *www.kernel.org*...

Cada dominio que es configurado apuntará al visitante a una carpeta específica que contiene la información del sitio; nunca indicará que el mismo servidor es responsable de otros sitios.

Una gran ventaja en el uso de *hosts virtuales* en Apache, es que permite heredar la configuración del sitio principal por lo que no habrá que reconfigurar todas las directivas, solo las que cambien.



Existen tres formas diferentes de crear hosts virtuales:

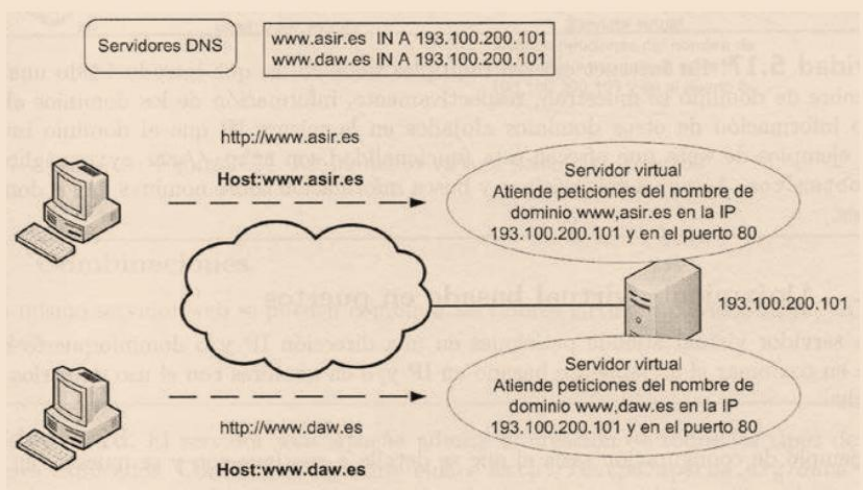
- Alojamiento virtual basado en **nombres**
- Alojamiento virtual basado en **IPs**
- Alojamiento virtual basado en **puertos**

ALOJAMIENTO VIRTUAL BASADO EN NOMBRES (NAME-BASED)

El servidor permite alojar varios nombres de dominio 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.

El servidor Web es capaz de diferenciar el nombre de dominio por el que pregunta el cliente porque en el mensaje de petición HTTP se envía la *cabecera* ***Host*** con el nombre de dominio.
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. Este método hace muy fácil migrar nuestro servidor a otra dirección IP.

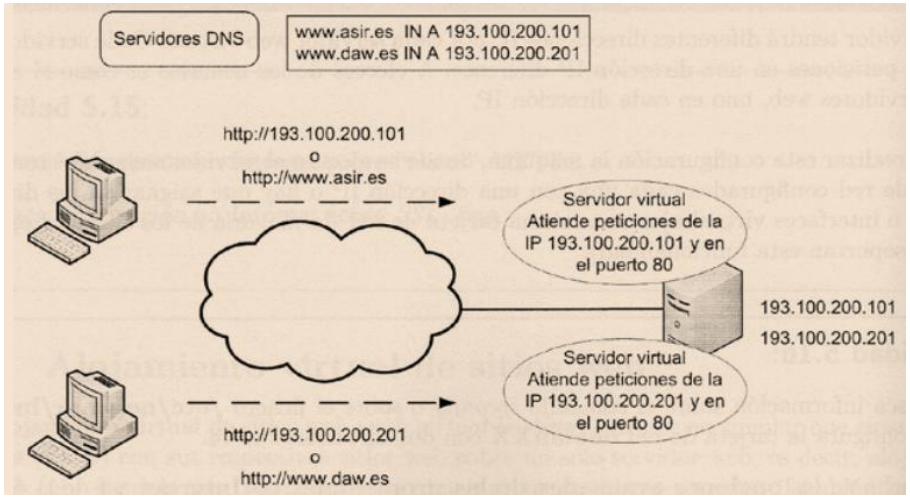


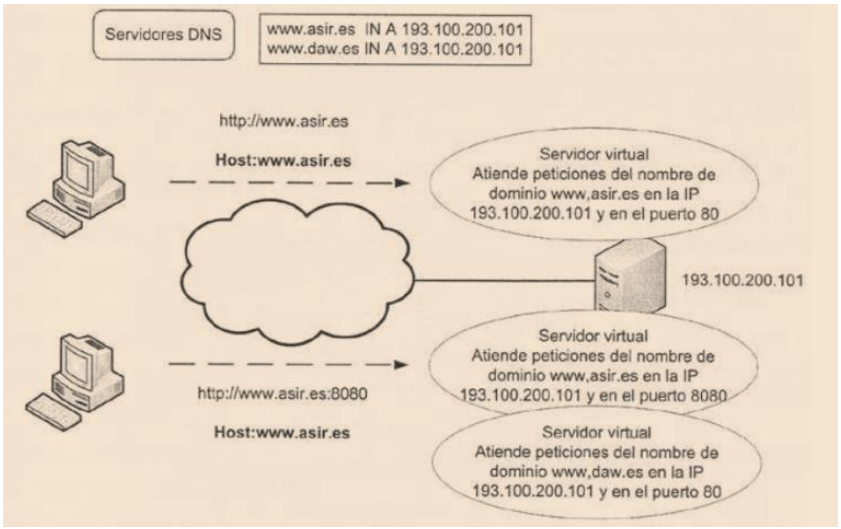
Haz clic aquí para consultar varios ejemplos de configuración de ***VirtualHost*** *en* ***Apache****.*

ALOJAMIENTO VIRTUAL BASADO EN IPS (IP-BASED)

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, 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.





ALOJAMIENTO VIRTUAL BASADO EN PUERTOS (PORT-BASED)

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 la escucha.

El uso de estas alternativas no es excluyente; en un mismo servidor Web se pueden combinar servidores virtuales basados en IP, en nombres y en puertos.

REFERENCIAS

Bibliografía

Servicios de red e internet. García Sánchez, Álvaro. Sanz Rodríguez, Javier. González Sotillo, Álvaro. Enamorado Sarmiento, Luis. Garceta Grupo Editorial.

Servicios en Red. Andreu Gómez, Joaquín . Editorial Editex

Webgrafía

Wikipedia

www.freepik.es

recursostic.educacion.es

Otros…

