**HTTP**

**y**

**Apache HTTPD**

Material creado para **Desarrollo Web en Entorno Servidor**

* Desarrollo de Aplicaciones Web II -



Índice

[1. Conceptos previos 2](#_crqz4ps68771)

[1.1 HTTP 2](#_qmnp7t52e1bm)

[1.2 Transacciones 3](#_7g6tuiw0udem)

[1.3 HTTPS 4](#_1gv4gfsg4tto)

[2. Funcionamiento 5](#_fwv1pjsl5775)

[2.1 Características básicas: 5](#_kxphpahh4dgh)

[2.2 Arquitectura de los sistemas basados en HTTP 5](#_4d9hobdtip8h)

[2.3 Flujo de HTTP 7](#_uf3bwkpw8xxx)

[3. Comandos HTML 8](#_5hoo6dq9dck5)

[4. Servidores Web. APACHE 11](#_ro7q1p64l42f)

[4.1 Servidor Web: Apache 11](#_ya37ew1bd8c5)

[4.2 Obtener Apache 11](#_w14o3es7fww4)

[4.3 XAMPP 11](#_azmhl3iy6rei)

[4.4 Iniciar y cerrar XAMPP 12](#_qmw4w2v89k30)

[4.3 Ficheros de configuración 13](#_905wa9f9l2a8)

[Bibliografía 14](#_kaaxubetrxy6)

# Conceptos previos

Los exploradores se comunican con los servidores usando el Protocolo de Transporte de Hyper Texto (HyperText Transport Protocol - HTTP).

El explorador envía una petición HTTP al servidor web. La petición incluye un URL que identifica el recurso afectado, un método que define la acción requerida (por ejemplo, obtener, borrar o publicar el recurso), y puede incluir información adicional codificada en parámetros en el URL (los pares campo-valor enviados en una cadena de consulta (query string).

Los servidores web esperan los mensajes de petición de los clientes, los procesan y responden al explorador web con un mensaje de respuesta HTTP.

## 1.1 HTTP

El HTTP (HyperText Transfer Protocol - Protocolo de Transferencia de Hiper Textos) es el protocolo de transmisión de información (a través de archivos XML, HTML,...) en la World Wide Web, es decir, el código que se establece para que el computador solicitante y el que contiene la información solicitada puedan “hablar” un mismo idioma a la hora de transmitir información por la red. Fue diseñado para la comunicación entre los navegadores y servidores web.

Establece criterios de sintaxis y semántica informática (forma y significado) para el establecimiento de la comunicación entre los diferentes elementos que constituyen la arquitectura web.

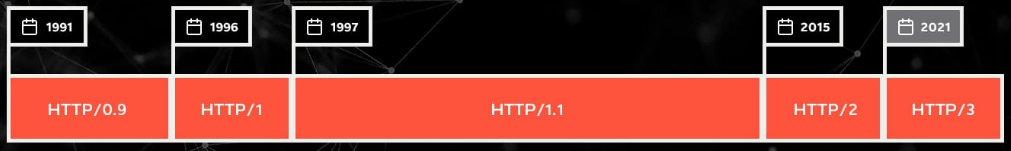
Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor (protocolo cliente-servidor). Al cliente que efectúa la petición se lo conoce como "user agent" (agente usuario). A la información transmitida se la llama recurso y se la identifica mediante un localizador uniforme de recursos (URL). Los recursos pueden ser archivos, el resultado de la ejecución de un programa, una consulta a una base de datos, la traducción automática de un documento, etc.

Un cliente establece una conexión con el servidor, realiza una petición y espera hasta que recibe una respuesta del mismo.

Clientes y servidores se comunican intercambiando mensajes individuales. Los mensajes que envía el cliente, normalmente un navegador Web, se llaman peticiones, y los mensajes enviados por el servidor se llaman respuestas.

Además, es un protocolo “sin estado”, que no guarda información sobre conexiones anteriores, sino que siempre empieza de nuevo. Para poder mantener la información relativa a visitas previas se utilizan las “cookies”, que se almacenan en el cliente.

Este protocolo tiene cierta flexibilidad para incorporar nuevas peticiones y funcionalidades, en especial a medida que se avanza en sus versiones.



## 1.2 Transacciones

Una transacción HTTP está formada por un encabezado seguido, opcionalmente, por una línea en blanco y algún dato.

El encabezado especificará cosas como la acción requerida del servidor, o el tipo de dato retornado, o el código de estado.

Un encabezado es un bloque de datos que precede a la información propiamente dicha, por lo que muchas veces se hace referencia a él como metadato -porque tiene datos sobre los datos-.

El servidor envía al cliente:

* Un código de estado que indica si la petición fue correcta o no. Los códigos de error típicos indican que el archivo solicitado no se encontró, que la petición no se realizó de forma correcta o que se requiere autenticación para acceder al archivo.
* La información propiamente dicha. Como HTTP permite enviar documentos de todo tipo y formato, es ideal para transmitir multimedia, como gráficos, audio y video.
* Información sobre el objeto que se retorna.

## 1.3 HTTPS

El HTTPs (HyperText Transfer Procotol Secure - Protocolo Seguro de Transferencia de Hipertexto), es la versión segura del http.

El protocolo HTTPS impide que otros usuarios puedan interceptar la información confidencial que se transfiere entre el cliente y el servidor web a través de Internet.

Para que una página web funcione bajo el protocolo HTTPS es necesario instalar un Certificado SSL (Secure Sockets Layer) en el servidor en el que está alojada.

Este certificado de seguridad es el encargado de cifrar las conexiones entre el navegador y servidor web impidiendo que nadie pueda interceptar la información que se transfiere entre ambos.

# Funcionamiento

Su funcionamiento se basa en operaciones de solicitud/respuesta.

Un cliente establece una conexión con un servidor y envía un mensaje con los datos de dicha solicitud.

El servidor responde con un mensaje en el que indica el estado de la operación y su posible resultado.

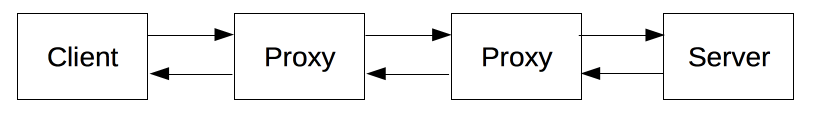
## 2.1 Características básicas:

* Existen varios verbos para que un cliente pueda dialogar con el servidor. Los más utilizados son:
  + GET: para recoger un objeto
  + POST: para enviar información al servidor
  + HEAD: para solicitar las características de un objeto (por ejemplo, la fecha de modificación de un documento HTML).
* No mantiene estado. Cada petición de un cliente a un servidor no es influida por las transacciones anteriores. El servidor trata cada petición como una operación totalmente independiente del resto.
* Cada objeto al que se aplican los verbos del protocolo está identificado a través de un localizador uniforme de recurso (URL) único.

## 2.2 Arquitectura de los sistemas basados en HTTP

Las peticiones son enviadas por una entidad: el agente del usuario. La mayoría de las veces el agente del usuario (cliente) es un navegador Web, pero podría ser cualquier otro programa, como por ejemplo un programa-robot, que explore la Web, para adquirir datos de su estructura y contenido para uso de un buscador de Internet.

Cada petición individual se envía a un servidor, el cuál la gestiona y responde. Entre cada petición y respuesta, hay varios intermediarios (proxies). Un servidor proxy retransmite el tráfico entre el dispositivo (cliente o servidor) y la Web, con lo que evita que el navegador cliente esté en contacto directo con los sitios que visita. Sus solicitudes web pasan primero a través del servidor proxy



**Cliente:**

Cualquier herramienta que actué en representación del usuario. Suele ser el navegador Web, pero hay excepciones, como el caso de programas específicamente usados por desarrolladores para desarrollar y depurar sus aplicaciones.

Para poder mostrar una página Web, el navegador envía una petición de documento HTML al servidor. Entonces procesa este documento, y envía más peticiones para solicitar scripts, hojas de estilo (CSS), y otros datos que necesite (normalmente vídeos y/o imágenes). El navegador, une todos estos documentos y datos, y compone el resultado final: la página Web. Los scripts, los ejecuta también el navegador, y también pueden generar más peticiones de datos en el tiempo, y el navegador, gestionará y actualizará la página Web en consecuencia.

**Servidor Web:**

Le "sirve" los datos que ha pedido el cliente.

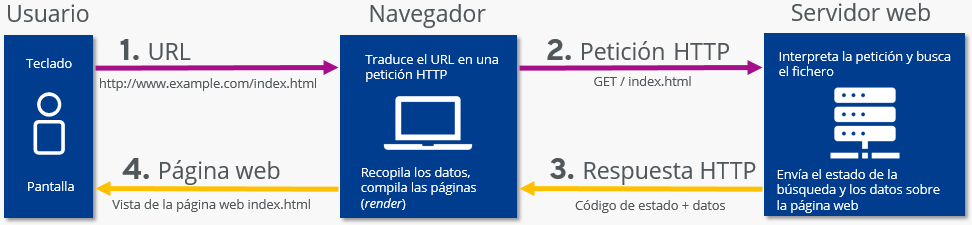
Un servidor no tiene que ser necesariamente un único equipo físico, varios servidores pueden estar funcionando en un único computador.

**Proxies:**

Entre el cliente y el servidor hay distintos dispositivos que gestionan los mensajes HTTP. La mayoria de estos dispositivos solamente gestionan estos mensajes en los niveles de protocolo inferiores: capa de transporte, capa de red o capa física, siendo así transparentes para la capa de comunicaciones de aplicación del HTTP, además esto aumenta el rendimiento de la comunicación. Los dispositivos, que sí operan procesando la capa de aplicación son conocidos como proxies. Estos pueden ser transparentes, o no (modificando las peticiones que pasan por ellos), y realizan varias funciones:

* caching (la caché puede ser pública o privada, como la caché de un navegador)
* filtrado (como un anti-virus, control parental, ...)
* balanceo de carga de peticiones (para permitir a varios servidores responder a la carga total de peticiones que reciben)
* autentificación (para el control al acceso de recursos y datos)
* registro de eventos (para tener un histórico de los eventos que se producen)

## 2.3 Flujo de HTTP



Cada vez que un cliente realiza una petición a un servidor, se siguen los siguientes pasos:

1. Un usuario accede a una URL, seleccionando un enlace de un documento HTML o introduciéndola directamente en el navegador.

2. El cliente web descodifica la URL, separando sus diferentes partes. Así identifica el protocolo de acceso, la dirección DNS o IP del servidor, el puerto (de carácter opcional; el valor por defecto es 80) y el objeto requerido del servidor.

3. Se abre una conexión TCP/IP con el servidor, llamando al puerto TCP correspondiente. La conexión se puede usar para hacer una petición o varias, y recibir la respuestas. El cliente puede abrir una conexión nueva, reusar una existente, o abrir varias a la vez hacia el servidor.

4. Se realiza la petición. Para ello, se envía el comando necesario (GET, POST, HEAD,...), la dirección del objeto requerido (el contenido de la URL que sigue a la dirección del servidor), la versión del protocolo HTTP empleada y un conjunto variable de información, que incluye datos sobre las capacidades del navegador, datos opcionales para el servidor, etc.

5. El servidor devuelve la respuesta al cliente. Consiste en un código de estado y el tipo de dato MIME de la información de retorno, seguido de la propia información.

6. Se cierra la conexión TCP. Este proceso se repite para cada acceso al servidor HTTP.

El diálogo con los servidores HTTP se establece a través de mensajes formados por líneas de texto, cada una de las cuales contiene los diferentes comandos y opciones del protocolo. Solo existen dos tipos de mensajes, uno para realizar peticiones y otro para devolver la correspondiente respuesta.

Ejemplo: descripción básica de cómo se abre una página web:

1.- En la barra de direcciones del navegador, el usuario teclea http://example.com/.

2.- El navegador envía esa solicitud, es decir, la petición HTTP, al servidor web que administre el dominio example.com. Normalmente, la solicitud del cliente dice algo así como “Envíame este archivo”, pero también puede ser “¿Tienes este archivo?”.

3.- El servidor web recibe la solicitud HTTP, busca el archivo en cuestión (en nuestro ejemplo, la página de inicio de example.com, que corresponde al archivo index.html) y envía en primer lugar una cabecera o header. Esta cabecera le comunica al cliente, mediante un código de estado, el resultado de la búsqueda.

4.- Si se ha encontrado el archivo solicitado y el cliente ha solicitado recibirlo (y no solo saber si existe), el servidor envía, tras el header, el message body o cuerpo del mensaje, es decir, el contenido solicitado: en nuestro ejemplo, el archivo index.html.

5.- El navegador recibe el archivo y lo abre en forma de página web.

# Comandos HTML

* GET: Sirve para recoger cualquier tipo de información del servidor. Se utiliza siempre que se pulsa sobre un enlace o se teclea directamente a una URL. Como resultado, el servidor HTTP envía el documento ubicado en la dirección especificada por dicha URL.
* HEAD: Es un comando similar a GET pero que pide solamente la cabecera del objeto. Lo utilizan principalmente los gestores de cachés de páginas o los servidores proxy para conocer cuándo es necesario actualizar la copia que se mantiene de un fichero.
* POST: Este comando envía datos de información al servidor, normalmenten procedentes de un formulario web, para que el servidor los administre o los añada a una base de datos.
* PUT: Almacena un objeto en la URL especificada. Si la dirección de destino ya contenía un objeto, se considera que se está enviando una versión actualizada del mismo.
* DELETE: Elimina el objeto especificado. Este comando es muy poco utilizado.
* TRACE: Realiza un eco de la solicitud recibida para que el cliente pueda conocer qué servidores intermedios están añadiendo información o modificando la petición.
* OPTIONS: Devuelve los métodos HTTP que soporta el cliente. Se suele utilizar para comprobar la funcionalidad de un servidor web.
* CONNECT: Se utiliza en los servidores proxy que puedan establecer un túnel dinámicamente (por ejemplo, un túnel SSL).

Ante cada transacción con un servidor HTTP, éste devuelve un código numérico en la primera línea del mensaje de respuesta que informa sobre el resultado de la operación. Estos códigos aparecen en algunos casos en la pantalla del cliente, cuando se produce un error. Están clasificados en cinco categorías.

* 1xx: Mensajes informativos.
* 2xx: Mensajes asociados con operaciones realizadas correctamente.
* 3xx: Mensajes de redirección, que informan de operaciones complementarias que se deben realizar para finalizar la operación.
* 4xx: Errores del cliente
* 5xx: Errores del servidor, que no ha podido llevar a cabo una solicitud.

Códigos más frecuentes:

| Código | Comentario | Descripción |
| --- | --- | --- |
| 200 | OK | Operación realizada satisfactoriamente. |
| 201 | Created | La operación ha sido realizada correctamente, y como resultado se ha creado un nuevo objeto, cuya URL de acceso se proporciona en el cuerpo de la respuesta. Este nuevo objeto ya está disponible. |
| 202 | Accepted | La operación ha sido realizada correctamente, y como resultado se ha creado un nuevo objeto, cuya URL de acceso se proporciona en el cuerpo de la respuesta. El nuevo objeto no está disponible por el momento. En el cuerpo de la respuesta se debe informar sobre la disponibilidad de la información. |
| 204 | No Content | La operación ha sido aceptada, pero no ha producido ningún resultado de interés. El cliente no deberá modificar el documento que está mostrando en este momento. |
| 301 | Moved  Permanently | El objeto al que se accede ha sido movido a otro lugar de forma permanente. El servidor proporciona, además, la nueva URL en el campo Location de la respuesta. |
| 302 | Found | El objeto al que se accede ha sido movido a otro lugar de forma temporal. El servidor proporciona, además, la nueva URL en el campo Location de la respuesta. El cliente no debe modificar ninguna de las referencias a la URL errónea. |
| 304 | Not Modified | Se devuelve cuando se hace un GET condicional y el documento no ha sido modificado. |
| 400 | Bad Request | La petición tiene un error de sintaxis y no es entendida por el servidor. |
| 401 | Unauthorized | La petición requiere una autorización especial, que  normalmente consiste en un nombre y clave que el servidor verificará. El campo WWW-Autenticate informa de los protocolos de autentificación aceptados para este recurso. |
| 403 | Forbidden | Está prohibido el acceso a este recurso. No es posible utilizar una clave para modificar la protección. |
| 404 | Not Found | La URL solicitada no existe. |
| 500 | Internal  Server Error | El servidor ha tenido un error interno, y no puede continuar con el procesamiento. |
| 501 | Not  Implemented | El servidor no tiene capacidad, por su diseño interno, para llevar a cabo el requerimiento del cliente. |
| 502 | Bad Gateway | El servidor, que está actuando como proxy o pasarela, ha encontrado un error al acceder al recurso que había solicitado el cliente. |
| 503 | Service  Unavailable | El servidor está actualmente deshabilitado y no es capaz de atender el requerimiento. |

# Servidores Web. APACHE

Para el desarrollo de aplicaciones web y sitios web con PHP se necesitan tres cosas: PHP, un servidor web y un navegador web.

## 4.1 Servidor Web: Apache

HTTPD, o Apache HTTP Server, es un software de servidor web que permite servir páginas web a clientes a través de Internet utilizando el Protocolo de transferencia de hipertexto (HTTP).

Se trata de un software gratuito y de código abierto.

Es multiplataforma.

Es modular. Tiene muchos módulos adicionales y se integra perfectamente con PHP y MySQL.

Tiene una fácil configuración y personalización.

Tiene una gran comunidad de desarrolladores

Es compatibile con IPv6

## 4.2 Obtener Apache

Apache suele venir incorporado en la mayoría de distribuciones de Linux, aunque el proceso en si no se llama Apache sino httpd y se inicia con el script /etc/init.d/httpd (en algunas distribuciones se sutiliza el nombre httpd2).

Para otros Sistemas Operativos que no lo llevan incluido, se puede descargar desde la página web [www.apache.org](http://www.apache.org).

## 4.3 XAMPP

AMP son las siglas de Apache, MySQL y PHP/Perl/Python. Las dos primeras siglas hacen referencia al servidor web (Apache) y al servidor de base de datos (MySQL). La última se corresponde con el lenguaje de programación utilizado, que puede ser PHP, Perl o Python.

Dependiendo del sistema operativo que se utilice para el servidor, se utilizan las siglas LAMP (para Linux), WAMP (para Windows) o MAMP (para Mac). También es posible usar otros componentes, como el gestor de bases de datos PostgreSQL en lugar de MySQL.

Todos los componentes de esta arquitectura son de código libre (open source). Su gran ventaja es la gran comunidad que la soporta y la multitud de aplicaciones de código libre disponibles.

Existen paquetes software que incluyen en una única instalación una plataforma AMP completa. El más conocido, y que usaremos nosotros en el aula, es XAMPP.

La X nos indica que se trata de un servidor independiente de la plataforma y PP nos indica que incluye los intérpretes para lenguajes PHP y Perl.

También incluye otros módulos como OpenSSL y phpMyAdmin así como un servidor para aplicaciones de Java Tomcat, un servidor de correo Mercury y un servidor FTP entre otras herramientas.

El programa está liberado bajo la licencia GNU y actúa como un servidor web libre, fácil de usar y capaz de interpretar páginas dinámicas. Actualmente XAMPP está disponible para Microsoft Windows, GNU/Linux, Solaris y Mac OS X.

## 4.4 Iniciar y cerrar XAMPP

Apache funciona como un servicio por lo que hay que ponerlo en marcha, iniciarlo.

En el aula utilizaremos XAMP (distribución de Apache que contiene MariaDB y PHP)

1-XAMPP Control panel

2-Apache->Start: puertos 80, 443 (Podemos ver que funciona poniendo localhost en el navegador o con el botón Admin)

3-MySQL-> Start: puerto 3306 (Podemos ver que funciona poniendo localhost/phpmyadmin/ en el navegador o con el botón Admin)

Al terminar de usarlo es importante cerrarlo correctamente:

1-cerrar navegador

2-MySQL->Stop

3-Apache->Stop

4-Quit

Si no se cerrado correctamente es posible que Mysql no arranque:

Z:\xampp\mysql\backup\mysql

Copiar todo el contenido de esa carpeta y vamos a:

Z:\xampp\mysql\data\mysql

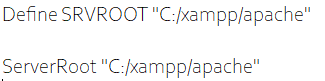
Borramos todo el contenido y pegamos el que hemos copiado del otro sitio.

## 4.3 Ficheros de configuración

* **conf**: directorio en el que se guardan los ficheros de configuración



* + httpd.conf: fichero de configuración principal. Dividido en tres secciones:
    - Sección Global: Controla el comportamiento general de Apache
      * ServerRoot: ruta al directorio donde se encuentran las carpetas bin, conf, htdocs.



* + - * Listen: dirección y puerto en el que escucha Apache. Por ejemplo Listen 127.0.0.1:80 hace que apache sólo responda a peticiones realizadas desde la propia máquina.
    - Sección Principal.
      * ServerName. nombre completo del servidor. Normalmente ó tiene el valor localhost o el nombre de un dominio ([www.mipagina.com](http://www.mipagina.com)).
      * DocumentRoot: directorio a partir del cual se buscará el índice principal del servidor. apache buscará un archivo de los que aparecen en la directiva DirectoryIndex.
      * DirectoryIndex: lista de recursos (por orden) que se buscarán cuando un cliente solicite un directorio.
    - Sección de los Host Virtuales (VirtualHosts): Los Host virtuales permiten que un mismo apache de servicio a varios dominios web. Su configuración se realiza en la directiva <VirtualHost>
* **htdocs**: directorio en el que se guardan los documentos del sitio web
* **include**: directorio de los módulos adicionales
* **logs**: directorio donde se guarda información sobre los accesos.

# Bibliografía

Mozilla. ***MDN Web Docs. Introducción al lado del servidor***. <https://developer.mozilla.org/es/docs/Learn/Server-side/First_steps/Introduction>

Prieto Donate, Francisco. ***Protocolo HTTP***. <https://www.academia.edu/36026373/5_Protocolo_HTTP_5_PROTOCOLO_HTTP>

Mozilla. ***MDN Web docs. HTTP***. <https://developer.mozilla.org/es/docs/Web/HTTP>

Stackscale**. *HTTP/3, el nuevo protocolo HTTP basado en UDP.***  <https://www.stackscale.com/es/blog/http3/>

Ionos, Digital Guide**. *¿Qué es el HTTP?.*** <https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/protocolo-http/>

Acebeiro, Maria. ***¿Cuál es la diferencia entre HTTP y HTTPS?***. <https://es.godaddy.com/blog/diferencia-entre-http-y-https/>