# EL PROTOCOLO HTTP

## Introducción

El Protocolo de Transferencia de HiperTexto (*HyperText Transfer Protocol*) es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP. La especificación completa del protocolo HTTP 1/0 está recogida en el RFC 1945. Fue propuesto por Tim Berners-Lee, atendiendo a las necesidades de un sistema global de distribución de información como el World Wide Web. Desde el punto de vista de las comunicaciones, está soportado sobre los servicios de conexión TCP/IP, y funciona de la misma forma que el resto de los servicios comunes de los entornos UNIX: un proceso servidor escucha en un puerto de comunicaciones TCP (por defecto, el **80**), y espera las solicitudes de conexión de los clientes web. Una vez que se establece la conexión, el protocolo TCP se encarga de mantener la comunicación y garantizar un intercambio de datos libre de errores. HTTP se basa en sencillas operaciones solicitud/respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado. Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto Web (documento HTML, fichero multimedia o aplicación CGI) es conocido por su URL.

HTTP fue creado para que las computadoras se comuniquen mientras intercambian documentos, agregando conectividad e interfaces. Usando HTTP, una computadora que pida un archivo a otra sabrá, al recibirlo, si se trata de imagen, video o texto.

Con esta nueva función que agregó HTTP, Internet comenzó a demostrar que es inútil recuperar un archivo si no se sabe qué tipo de datos tiene. Con la cantidad de documentos diferentes que hay en la Web, sería imposible saber qué tipo de dato tiene uno de ellos por adelantado, ya que podría contener cualquier cosa. Pero la Web entiende los tipos de datos y pasa esa información.

Aunque no es necesario que conocer cómo funciona HTTP para, por ejemplo, construir un programa **CGI**, es bueno el conocimiento de todo esto para hacer tu trabajo con mayor fluidez y confianza. Como en cualquier rubro, un vistazo a los principios de base te permite tener una mayor comprensión general y un mejor y más completo modelo mental en el que basar tu trabajo.

Debajo de la interfaz de usuario, representada por los navegadores, están los protocolos. Estos viajan por los cables que conforman la red hacia los servidores o motores que procesan los pedidos de información (**requests**) y devuelven resultados. El protocolo de Web se conoce como HTTP. Es el mecanismo de base en el cual opera CGI, por lo que es quien determina lo que puedes y lo que no puedes hacer con CGI.

Según la especificación del protocolo, "*HTTP es un protocolo del nivel de aplicación con la agilidad y velocidad necesaria para sistemas de información distribuidos, colaborativos y de hipermedia. Es un protocolo orientado a objetos, genérico, que puede usarse para muchas*

*tareas extendiendo sus métodos. Una característica de HTTP es que permite que los sistemas se construyan independientemente de la información que se transfiere*".

Los recursos u objetos que actúan como entrada o salida de un comando HTTP están clasificados por su descripción MIME. De esta forma, el protocolo puede intercambiar cualquier tipo de dato, sin preocuparse de su contenido. La transferencia se realiza en modo binario, byte a byte, y la identificación MIME permitirá que el receptor trate adecuadamente los datos.

# Propiedades de HTTP

- ## Esquema de direccionamiento integral

  El protocolo usa el concepto de referencia dado por URI (Universal Resource Identifier) como una ubicación o como un nombre - URL y URN respectivamente -para indicar la fuente donde debe aplicarse un método. Cuando un hiperlink HTML se conforma, la URL es de la forma:

  ```
  http://host:número_puerto/path/archivo.html
  ```

  Para decir algo más general, la referencia URL es del tipo:

  ```
  servicio://host/archivo.extensión
  ```

  De esta manera el protocolo puede abarcar los servicios de Internet más básicos.

  HTTP también se usa para la comunicación entre agentes y gateways, permitiendo el acceso a otros protocolos de Internet existentes, como SMTP, NNTP, FTP, Gopher, etc. HTTP está diseñado para permitir la comunicación con esos gateways, vías servidores proxy, sin pérdida de la información transportada por estos otros protocolos.

- ## Arquitectura Cliente-Servidor

  HTTP se basa en el paradigma pedido/respuesta. La comunicación generalmente se lleva a cabo sobre una conexión **TCP/IP** en Internet. El puerto por defecto es el **80**, pero otros puertos también pueden usarse. Esto no imposibilita que el protocolo se implemente arriba de algún otro protocolo de Internet, siempre y cuando se garantice la confiabilidad.

  Un programa cliente establece la conexión con un programa del servidor y envía un pedido a este último mediante el método **request**, URL y versión de protocolo, seguido por un mensaje que contiene los modificadores del pedido, información sobre el cliente y contenido. El servidor responde con una línea de estado, incluyendo su versión de protocolo y un mensaje de éxito o error, seguido de un mensaje que contiene información del servidor, metainformación y contenido.

- ## Sin conexión

  Aunque dijimos que el cliente establece una conexión con el servidor, se dice que el protocolo es sin conexión porque una vez que el pedido está satisfecho, la conexión cae. Otros protocolos mantienen la conexión abierta. Por ejemplo, en una sesión FTP puedes

moverte de un directorio remoto a otro y el servidor irá siguiendo tus movimientos, para saber en todo momento quién eres y dónde estás.

Mientras que esto simplifica enormemente la construcción del servidor y libera al mismo de carga que disminuiría la performance, el seguimiento del usuario se hace imposible. Además, se debe recurrir a múltiples conexiones para documentos que consisten de más de una imagen en línea, ya que cada una se recupera individualmente en una conexión.

- Sin estado

  Luego que el servidor responde a un pedido del cliente, la conexión cae y es olvidada. Es decir, no hay una memoria de conexiones de un cliente. Las implementaciones de servidor HTTP puras tratan todos los pedidos como si fueran únicos y nuevos.

  Las aplicaciones CGI pueden salvar esto codificando el estado (o un identificador de estado) en campos ocultos, en la información del path o en URLs que se retornan al navegador. Los dos primeros métodos devuelven el estado (o su identificador) cuando el formulario es enviado de vuelta al servidor; el método que usa la URL sólo devuelve el estado (o su identificador) si el usuario hace click en el link y el link vuelve al servidor.

  Es aconsejable no codificar todo el estado, sino almacenarlo en un archivo e identificarlo con un identificador único, como un entero secuencial. Los programas contadores de visitantes pueden adaptarse muy bien a esto, por lo que son muy útiles. Entonces sólo debes enviar el identificador de estado en el formulario, lo que ahorra tráfico de red. Sin embargo, debes acordarte de hacer el mantenimiento de los archivos de estado.

- Representación abierta y extensible de tipos de datos

  HTTP usa Internet Media Types, comúnmente llamados tipos de contenido MIME (Multipart Internet Mail Extension), para brindar una negociación de tipos extensible y abierta. Con HTTP, donde el emisor y el receptor se pueden comunicar directamente, las aplicaciones tienen permitida una mayor libertad en el uso de tipos no registrados. Para aplicaciones de mail, por ejemplo, donde no hay una negociación de tipos entre el emisor y el receptor, es razonable poner límites estrictos en el conjunto de tipos permitidos.

  Cuando el cliente envía una transacción al servidor, se agregan encabezados (headers) para conformar las especificaciones estándares de e-mail (RFC822). La mayoría de los pedidos de clientes esperan una respuesta en "sólo texto" o en HTML. Cuando el servidor HTTP transmite información de vuelta al cliente, incluye un encabezado simil-MIME para informar al cliente qué tipo de dato viene después de dicho encabezado. La traducción depende de que el cliente procese la utilidad adecuada que corresponda a ese tipo de dato. Por ejemplo, un visor de imágenes si es una imagen, de video si es un video, etc.).

## Campos de encabezados

Una transacción HTTP consiste de 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.

El uso de campos de encabezados enviados en las transacciones HTTP le da gran flexibilidad al protocolo. Estos campos permiten que se envíe información descriptiva en la transacción, permitiendo así la autenticación, encriptación e identificación de usuario.

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-.

Si se reciben líneas de encabezado del cliente, el servidor las coloca en las **variables de ambiente de CGI** con el prefijo HTTP_ seguido del nombre del encabezado. Cualquier carácter guión (**-**) del nombre del encabezado se convierte a caracteres "_". El servidor puede excluir cualquier encabezado que ya esté procesado, como Authorization, Content-type y Content-length. El servidor puede elegir excluir alguno o todos los encabezados si incluirlos excede algún límite del ambiente de sistema. Ejemplos de esto son las variables HTTP_ACCEPT y HTTP_USER_AGENT.

- **`HTTP_ACCEPT`** Los tipos MIME que el cliente aceptará, dado los encabezados HTTP. Otros protocolos quizás necesiten obtener esta información de otro lugar. Los ítems en esta lista deben estar separados por una coma, como lo dice la especificación HTTP: tipo, tipo
- **`HTTP_USER_AGENT`** El navegador que utiliza el cliente para enviar el pedido. El formato general para esta variable es: software/versión librería/versión

El servidor envía al cliente:

- Un código de estado que indica si el pedido fue exitoso o no. Los códigos de error típicos indican que el archivo solicitado no se encontró, que el pedido no está en forma 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. Esta libertad es una de las mayores ventajas de HTTP.
- También envía información sobre el objeto que se retorna. La **lista de campos de encabezados** que hacen esto se muestra más abajo. La lista no es una lista completa de los campos de encabezado y que algunos de ellos sólo tienen sentido en una dirección.

# Métodos HTTP

HTTP ofrece un conjunto de métodos para indicar el propósito del pedido. Los tres más usados son HEAD, GET y POST.

- ## El método GET

  Se usa para pedir un documento específico. Cuando haces click en un link, se está usando el método GET. Este método debería usarse cuando el acceso URL no cambiará el estado de una base de datos. La semántica de GET cambia a un GET condicional si el mensaje de pedido incluye un campo de encabezado If-Modified-Since. Un GET condicional pide que la fuente identificada sea transferida sólo si ha

sido modificada después de la fecha dada por el encabezado. El GET condicional intenta reducir el uso de la red permitiendo que las entidades que están en caché sean refrescadas sin requerir la transferencia innecesaria de datos.

- El método HEAD

  Este método se usa sólo para preguntar información *sobre* un documento, no para el propio documento. HEAD es mucho más rápido que GET, ya que se transfiere mucha menos información. Es generalmente usada por clientes que usan caching para ver si el documento ha cambiado desde la última vez que lo accedieron. Si no cambió, la copia local puede usarse nuevamente; de otra forma, la versión actualizada debe recuperarse con GET. La metainformación contenida en los encabezados HTTP en respuesta a un pedido HEAD debería ser idéntica a la información enviada en respuesta a un pedido GET. Este método se puede usar para obtener metainformación sobre una fuente identificada por el URI sin transferir la información. Este método se usa mucho para testear la validez, accesibilidad y reciente modificación de links.

- El método POST

  Es usado para transferir datos del cliente al servidor; está diseñado para permitir que un método uniforme cubra funciones como: anotación de fuentes existentes; envío de mensajes para un *bulletin board*, grupo de noticias o lista de mailing; proveer un bloque de datos (generalmente un formulario) a un proceso de manejo de datos; extensión de una base de datos.

```
POST /cgi-bin/post-query HTTP/1.0
Accept: text/html
Accept: video/mpeg
Accept: image/gif
Accept: application/postscript
User-Agent: Lynx/2.2 libwww/2.14
From:xx@unp.edu.pe
Content-type: application/x-www-form-urlencoded
Content-length: 150
                                        * una línea en blanco*
org=CyberWeb%20SoftWare
&users=10000
&browsers=lynx
```

  Esto es una consulta POST dirigida a un programa residente en "**/cgi-bin/post-query**", que simplemente repite los valores que recibe. El cliente lista los tipos MIME que es capaz de aceptar y se identifica junto con la versión de la librería WWW que está usando. Finalmente indica el tipo MIME que ha usado para codificar la información que envía, el número de carácter incluido y una lista de variables y sus valores. **application/x-www-form-urlencoded** significa que los pares nombre-

valor de la variable se codificarán de la misma manera como se codificará la URL. Cualquier carácter especial, incluyendo puntuación, se codificará como %nn donde nn es el valor ASCII para el carácter en hexadecimal.

## Campos de encabezados HTTP

- ### Content-type

  Este campo indica el tipo de datos enviado o, en el caso del método **HEAD**, el tipo de datos que hubiera sido enviado si el método hubiera sido **GET**. Es usado por los navegadores para saber cómo manejar los datos. El cliente usa esta información para determinar cómo manejar un archivo de video o un gráfico. Un ejemplo:

  ```
  Content-Type: text/html
  ```

- ### Date

  El encabezado Date representa la fecha y hora en la que el mensaje se originó. Un ejemplo (fecha en inglés):

  ```
  Date: Wed, 31 May 2000 08:12:31 GMT
  ```

- ### Expires

  Da la fecha después de la cual la información en el documento deja de ser válida. Los clientes que hacen caching, incluyendo los proxys, no deben hacer caché de esa página más allá de la fecha indicada, a no ser que su estado haya sido actualizado por un chequeo posterior del servidor origen.

  ```
  Expires: Wed, 31 May 2000 08:12:31 GMT
  ```

- ### From

  Si está presente, debería contener una dirección de e-mail para el usuario. Puede usarse como medio para log-in y para identificar la fuente de un pedido inválido o no deseado. La interpretación de este campo es que el pedido está siendo realizado en nombre de la persona indicada, quien acepta la responsabilidad por el método ejecutado. En particular, los **robots** deberían incluir este encabezado para que la persona responsable de ejecutar el robot pueda ser contactada si ocurre un problema en el receptor.

  ```
  From: gm@unp.edu.pe
  ```

- ### If-Modified-Since

  Se usa con el método GET para hacerlo condicional: si la fuente demandada no ha sido modificada desde la fecha especificada por el encabezado, no se devolverá una copia de la misma desde el servidor. En su lugar, se retornará una respuesta 304 (no modificado).

  Un ejemplo:

```
If-Modified-Since: Wed, 31 May 2000 08:12:31 GMT
```

- Last-Modified

  Indica la fecha y hora en la que el emisor cree que se realizó la última modificación. Es útil para clientes que eliminan lo innecesario haciendo caching. La semántica de este campo está definida según cómo el receptor debe interpretarla: si el receptor tiene una copia que es anterior a la fecha indicada por este campo, dicha copia se considera obsoleta. Un ejemplo:

  ```
  Last-Modified: Wed, 31 May 2000 08:12:31 GMT
  ```

- Location

  Este encabezado de respuesta define la ubicación exacta de la fuente que fue identificada por el URI. Si el valor es una URL completa, el servidor retorna una redirección al cliente para recuperar directamente el objeto especificado.

  ```
  Location: http://www.unp.edu.pe/index.html
  ```

  Si quieres referenciar otro archivo en el mismo servidor, debes poner una URL parcial:

  ```
  Location: /desarrollo/notas/index.html
  ```

  El servidor actuará como si el cliente hubiera pedido

  ```
  http://servidoractual/desarrollo/notas/index.html
  ```

  Se encargará del control de acceso, determinando el tipo de archivo, etc. En este caso los clientes no hacen la redirección, sino que el servidor la hace en el momento.

- Referer

  Este encabezado de pedido permite que el cliente especifique la dirección (URL) de la fuente donde obtuvo la URL del pedido. Esto permite que el servidor genere listas de back-links a fuentes. También permite que los links obsoletos o mal tipeados se rastreen para mantenimiento.

  Un ejemplo:

  ```
  Referer: http://www.unp.edu.pe/index.html
  ```

  Si se da una URL parcial, debe interpretarse como relativa a la URL del pedido. La URL no debe incluir un fragmento (#label dentro de un documento).

- Server

  Es un encabezado de respuesta. Contiene información sobre software usado por el servidor de origen para manejar el pedido. El campo puede contener múltiples fichas (*tokens*) de productos comentarios que identifiquen el servidor y cualquier subproducto significante. Por convención, los tokens de producto se listan en orden según la significancia con que identifiquen la aplicación.

  Por ejemplo:

  ```
  Server: CERN/3.0 libwww/2.17
  ```

- User-Agent

  Este campo contiene información sobre el agente usuario que origina el pedido. Esto es para fines estadísticos, para rastrear violaciones de protocolo y reconocimiento automático de agentes usuarios. Por convención, los tokens de producto se listan en orden según la significancia con que identifiquen la aplicación.

  Por ejemplo:

  ```
  User-Agent: CERN-LineMode/2.15 libwww/2.17b3
  ```

- Respuesta HTTP (Response)

  Aquí hay un ejemplo de una respuesta HTTP de un servidor a un pedido de cliente:

  ```
  HTTP/1.0 200 OK
  Date: Wednesday, 02-Jun-00 23:04:12 GMT
  Server: NCSA/1.3
  MIME-version: 1.0
  Last-modified: Monday, 15-Nov-99 23:33:16 GMT
  Content-type: text/html
  Content-length: 2345
  *una línea en blanco*

  <HTML> ...
  ```

  El servidor acepta usar la versión 1.0 de HTTP para comunicarse y envía el estado 200 indicando que ha procesado satisfactoriamente el pedido del cliente.

  Luego envía la fecha y se identifica como un servidor HTTP NCSA. También indica que está utilizando la versión 1.0 de MIME para describir la información que está enviando e incluye el tipo MIME (MIME-type) de la información que está por enviarse en el encabezado Content.-type. Finalmente envía el número de caracteres que va a enviar, seguido de una línea en blanco y de la información propiamente dicha.

## Las principales características del protocolo HTTP son:

- Toda la comunicación entre los clientes y servidores se realiza a partir de caracteres de 8 bits. De esta forma, se puede transmitir cualquier tipo de documento: texto, binario, etc., respetando su formato original.
- Permite la transferencia de objetos multimedia. El contenido de cada objeto intercambiado está identificado por su clasificación MIME.
- Existen tres verbos básicos (hay más, pero por lo general no se utilizan) que un cliente puede utilizar para dialogar con el servidor: GET, para recoger un objeto, POST, para enviar información al servidor y HEAD, para solicitar las características de un objeto (por ejemplo, la fecha de modificación de un documento HTML).

- Cada operación HTTP implica una conexión con el servidor, que es liberada al término de la misma. Es decir, en una operación se puede recoger un único objeto.
- 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 la información de situación del final de la URL.

HTTP se diseñó específicamente para el World Wide Web: es un protocolo rápido y sencillo que permite la transferencia de múltiples tipos de información de forma eficiente y rápida. Se puede comparar, por ejemplo, con FTP, que es también un protocolo de transferencia de ficheros, pero tiene un conjunto muy amplio de comandos, y no se integra demasiado bien en las transferencias multimedia.

## Etapas de una transacción HTTP

Para profundizar más en el funcionamiento de HTTP, veremos primero un caso particular de una transacción HTTP; en los siguientes apartados se analizarán las diferentes partes de este proceso. Cada vez que un cliente realiza una petición a un servidor, se ejecutan los siguientes pasos:

- Un usuario accede a una URL, seleccionando un enlace de un documento HTML o ingresando directamente en el campo "Location" del cliente Web. El cliente Web descodifica la URL, separando sus diferentes partes. Así identifica el protocolo de acceso, la dirección DNS o IP del servidor, el posible puerto opcional (el valor por defecto es 80) y el objeto requerido del servidor.
- Se abre una conexión TCP/IP con el servidor, llamando al puerto TCP correspondiente.
  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 (casi siempre HTTP/1.0) y un conjunto variable de información, que incluye datos sobre las capacidades del browser, datos opcionales para el servidor,... 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. Se cierra la conexión TCP.
- Este proceso se repite en cada acceso al servidor HTTP. Por ejemplo, si se recoge un documento HTML en cuyo interior están insertadas cuatro imágenes, el proceso anterior se repite cinco veces, una para el documento HTML y cuatro para las imágenes.

En la actualidad se ha mejorado este procedimiento, permitiendo que una misma conexión se mantenga activa durante un cierto periodo de tiempo, de forma que sea utilizada en sucesivas transacciones. Este mecanismo, denominado **HTTP Keep Alive**, es empleado por la mayoría de los clientes y servidores modernos. Esta mejora es imprescindible en una Internet saturada, en la que el establecimiento de cada nueva conexión es un proceso lento y costoso.

Veamos el proceso completo con un ejemplo:

- Desde un cliente se solicita la URL

  `http://www.unican.es/invest/default.html`

- Se abre una conexión TCP/IP con el puerto 80 del sistema

  `www.unican.es`

- El cliente realiza la solicitud, enviando algo similar a esto:

| | |
|---|---|
| `GET /invest/default.html HTTP/1.0` | Operación solicitada+objeto+versión de http |
| `Accept: text/plain` | Lista de tipos MIME que acepta o entiende el cliente |
| `Accept: text/html` | |
| `Accept: audio/*` | |
| `Accept: video/mpeg` | |
| `.. .. ..` | |
| `Accept: */*` | Indica que acepta otros posibles tipos MIME |
| `User-Agent: Mozilla/3.0 (WinNT; I)` | Información sobre el tipo de cliente |
| | Línea en blanco, indica el final de la petición |

- El servidor responde con la siguiente información:

| | |
|---|---|
| `HTTP/1.0 200 OK` | Status de la operación; en este caso, correcto |
| `Date: Monday, 7-Oct-96 18:00:00` | Fecha de la operación |
| `Server: NCSA 1.4` | Tipo y versión del servidor |
| `MIME-version: 1.0` | Versión de MIME que maneja |
| `Content-type: text/html` | Definición MIME del tipo de datos a devolver |
| `Content-length: 254` | Longitud de los datos que siguen |
| `Last-modified: 6-Oct-96 12:30:00` | Fecha de modificación de los datos |
| | Línea en blanco |

```
<HTML> Comienzo de los datos
<HEAD><TITLE>Recursos de investigación en UNICAN</TITLE></HEAD>
<BODY>
.. .. ..
.. .. ..
</HTML>
```

- Se cierra la conexión.