

# HTTP

## CLIENTE:

- Navegador
- CURL
- Librerías nativas de diversos lenguajes de desarrollo.

# HTTP Hipertext transport protocol

Version 1.1 RFC 2616

## SERVIDOR: Web server

Socket en el cliente  
abierto en port alto

Conexión TCP

Producida después de un handshaking  
de 3 vías (solicitud, aceptación  
y confirmación de conexión).

Pedido del Recurso  
principal

Mensaje de requerimiento  
+ información

Respuesta del recurso  
Principal como por ej.  
Una página HTML con  
sus elementos asociados

Mensaje de respuesta  
+info que puede reemplazar o sumar  
a la presentada en el mensaje anterior

Se pide y se recibe el  
primer elemento asociado  
al recurso principal por ej.  
Una imagen.

Mensaje de requerimiento

Socket tcp abierto en modalidad  
pasiva en la dirección IP y el port 80  
del servidor. La conexión se establece  
sobre pares IP-PORT. Los nombres y  
dominios expresados en el requerimiento  
deben ser traducidos previamente por  
servicios de resolución de nombres.  
El navegador realiza una consulta DNS  
antes de establecer esta conexión.

Socket fantasma abierto para  
sostener la conexión

Mensaje de respuesta

...

## CLIENTE

## SERVIDOR

Conexión TCP

HTTP → stateless protocol

## Estructura del mensaje HTTP

Válido tanto para mensajes de requerimiento como para mensajes de respuesta.

Aclaración (servidores RTSP): Los servidores de streaming de alta calidad tienen las siguientes ventajas sobre los basados en http:  
El cliente puede avanzar o retroceder a cualquier punto fino del video ya que este es transmitido en forma continua y no por trozos.  
Permite al server conocer exactamente lo que la gente consume.  
Usa eficientemente el ancho de banda ya que solo transmite lo que el cliente pide.  
El video nunca es almacenado del lado del cliente. La única desventaja es que por lo general estos protocolos son bloqueados por los firewalls corporativos. Por este motivo es poco común ver servidores RTSP en redes privadas corporativas. Por lo general se usan servicios de conexión para teleconferencia o de contenido en la nube (ej: netflix) y a los cuales se accede desde requerimientos salientes de las redes privadas.

<Linea inicial> (TEXTO)

Líneas de encabezados: (texto puro)

Header1:valor1

Header2:valor2

Header3:valor3

Definidos en RFC 822  
Todos terminan con CRLF  
Ascii 13 y 10

<Linea en blanco (CRLF)>

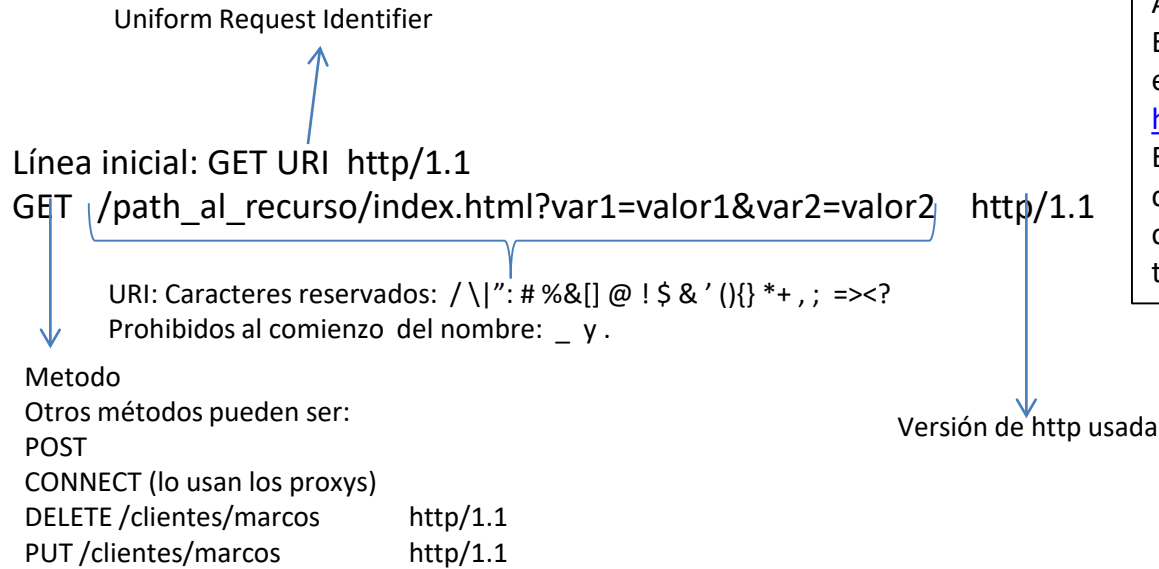
Puede llevar texto, files o flujo de datos  
Los files deben tener un header asociado que indique el tipo y otro que indique el tamaño.

El flujo (streaming http) no lleva header asociado (ej: video online). Se llama http live streaming (HLS).  
El servidor divide el flujo de video que debe enviar en pequeños fragmentos que viajarán como archivos independientes y que irán siendo almacenados temporalmente en el navegador cliente.

Los datos transferidos pueden ser de tipo texto para el control en el navegador o de tipo binario para las muestras de video.  
Los servidores de streaming de alta performance no usan protocolo http. (Usan por ej. RTSP real time streaming protocol y funcionan sobre UDP)

<BODY>

# Mensaje de Requerimiento



## Aclaración:

El campo URL del navegador es el que contiene el conjunto:

<http://nombreHost.dominioHost:puerto> + URI

Esta data la necesita el navegador primero para consultar la ip de destino al servidor de nombres configurado y luego para establecer la conexión tcp con dicho destino.

Los metodos pueden ser utilizados como verbos para la manipulación de recursos remotos en servicios web:

GET → Solo lectura

PUT → Utilizado para crear recursos en un servidor

DELETE → Utilizado para borrar recursos en un servidor

POST → Utilizado normalmente para actualizar recursos en un servidor.

/index.html  
/clientes  
/clientes/marta

Son los posibles recursos (resources) referenciados en la URI

## Headers: Contienen información relativa a la codificación del mensaje (metadatos)

Hay Tecnologías para implementar servicios en la WEB que usan estos headers para rutearlos

```
Remote address: 72.50.50.8:80
Request URL: http://www.xxx/documento.php
Request method: GET
Header size: 390 bytes      → tamaño total de encabezados
Request Protocol: http /1.1
User-Agent: Mozilla/5.0
Content-type: text/html (para datos enviados en el req.); charset=utf-8
Content-length: xxxxx      → logitud de los datos transmitidos hacia el servidor (textos o files)
Date:xxxxx
Accept image: /gif, application ms-word, application ms-powerpoint
Accept encoding: gzip, deflate
Accept language: es ar
Connection: Keep alive      → Pide al servidor que no corte la conexión mientras sea posible
Host: nombrehost.mombredominio
Accept-encoding: gzip
Accept-language: es-ES, en-US
Un site para consultar que parámetros son enviados por nuestro navegador puede ser: http://request.urih.com/
```

**Body:** Puede contener datos en cualquier formato.

- Datos ingresados en campos de un formulario.
- Files enviados en un proceso de upload

# Mensaje de Respuesta http

Línea inicial:

- Es una línea que indica el estado (status line)
- Informa acerca del estado del requerimiento

HTTP/1.1



versión

404



Código de estado  
Entero de 3 dígitos

Not Found



Descripción de estado

Códigos:

1xx	Mensaje de información provisorio para procedimientos experimentales
2xx	Recurso existente en el servidor (Ej: 200 OK, 201 Created)
3xx	Redirección a otra URL o manejo de cache (Por ejemplo cuando el servidor indica al cliente que recupere el recurso de su propio cache). Ej: 301 Recurso movido permanentemente a otra URL Ej: 303 o 304 Recurso no modificado desde que fue cacheado por el navegador.
4xx	Error en el requerimiento del cliente Ej: 400 Bad request, 401 Unauthorized Ej: 404 Recurso no encontrado 405 Method not allowed, 409 Conflict
5xx	Error en el procesamiento del requerimiento del lado del server (500 Server Error)

### Headers:

Los campos del Header indican entre otras cosas el tipo y el tamaño de los archivos y datos devueltos.

**Server: Apache/1.2**

**Last-Modified: Fri, 3 Dec 2017**

**Cache-control: no-cache → El servidor no admite cache (el valor public es para admitir cache)**

//ETag: "33a64df551425fcc55e4d42a148795d9f25f89d4" para el caso

//de querer forzar este tipo de manejo de cache.

**Content-type: text/html**

**Content-Encoding: gzip**

**Content-Length: xxx (en http 1.1 no es obligatorio. Si no está se trata de streaming)**

**Expires: 22 Jul 2018**

Todos los navegadores tienen la opción de visualizar los headers que vienen

Con la respuesta http (ej: chrome → Descargar la extensión HTTP headers (Configuración –

Más herramientas → extensiones). Un nuevo icono se agregará al lado del menú de configuración

de Chrome. Al entrar en cualquier página y clicar este icono se podrán visualizar los headers http de la respuesta)

### Body:

- En el Body de la respuesta va el recurso pedido en el requerimiento (Página html o archivo).

- Puede ir texto explicativo de algún estado de error.

## Ejemplo completo: Caso de envío de formulario

**POST /path/file.html HTTP/1.1**

**<headers>**

User-Agent: Mozilla/3.0, Content-type: text/html

Content-length:xxxxx,Date:xxxxx, host: www.miempresa.com:8080

**[CRLF]**

**<body>**

**Variables de formulario**

Headers

**HTTP/1.1 200 OK**

**Date: xxxxxxxx**

**Content-type: text/html**

**<CRLF>**

Body

**<html>**

**<body>**

**<h1>Texto</h1>**

**</body>**

**</html>**

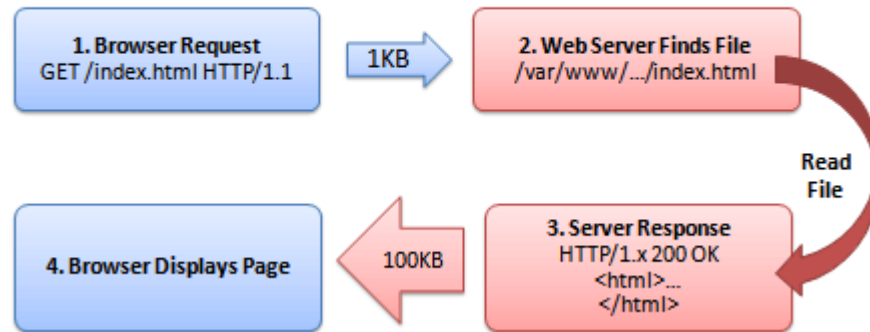


## Caches:

### Caso1: Sin cache

Requerimiento normal:  
el navegador no almacena ningún  
cache. La pagina es cargada totalmente  
en cada requerimiento.

## HTTP Request and Response



### Caso2: Cache utilizado fechas

Requerimiento que envía un http header como el siguiente:

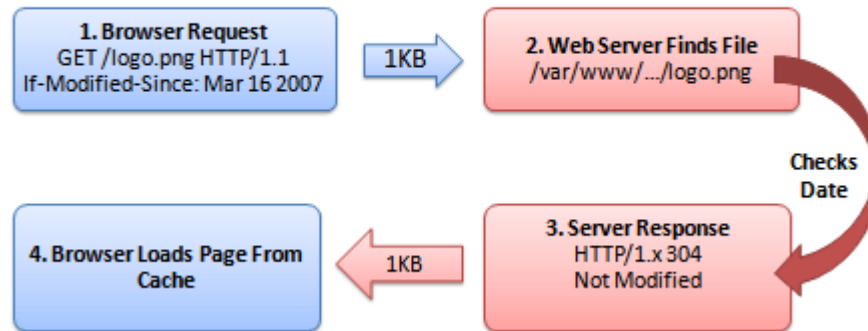
if-Modified-Since : xxxx (corresponde a ultima fecha de acceso al recurso)

El servidor busca el recurso y compara la fecha enviada en el header con la de ultima modificación. Si su fecha de ultima modificación es anterior entonces solo envia una respuesta con header «Modified = Not Modified» y el navegador carga el recurso de su cache.

Para que un navegador presente un header (if-Modified-Since) en el req. Este parámetro tiene que estar grabado en el cache. Es decir la asociación entre el recurso (url) y la fecha de último req.

Este valor de fecha será el enviado en el header "if-Modify since:".

## HTTP Cache: Last-Modified

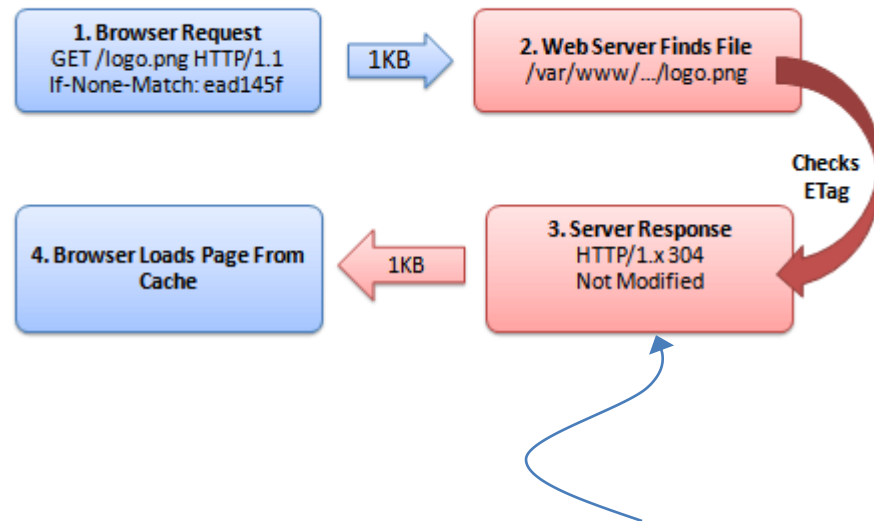


## Caso3: Cache utilizando Etag's o Hash's

Igual que el caso anterior pero utilizando un identificador (Etag) asociado a cada file del servidor. Este es una etiqueta o «finger print» que se modifica en el servidor cada vez que se modifica el recurso.

Lo que se compara entonces en el server es este header con el identificador del recurso. Si no fue modificado entonces solo devuelve una respuesta con un header «not Modified» Esto es mejor ya que no depende de errores en la configuración de fechas del servidor y cliente. Recordemos que en la mayoría de los casos los desarrolladores no tienen control sobre el sistema operativo del servidor donde corre el web server. Nuevamente aclaro que para que el navegador envíe un header «If-None-Match» habría sido necesario que la asociación entre recurso (url) y esta variable, esté grabada. Esto se habría realizado en el último acceso al recurso mencionado. El Etag queda grabado en el navegador ya que es también recibido por este en la cabecera de la respuesta y es enviado por con el requerimiento http en el siguiente acceso al URL.

### HTTP Cache: If-None-Match



En el caso de envío de nueva pagina. El server Debe enviar un header Con el nuevo eTag

## Mejoras del HTTP 1.1 respecto al HTTP1.0

1. Múltiples transacciones sobre conexiones persistentes.
2. Soporte de cache en el envío de respuestas. Se evalúan encabezados de requerimiento para comparar con los atributos de cada recurso y de esa manera decidir si el mismo se envía completo por haber sido modificado.
3. Permite respuestas de flujo de datos sin encabezados que indiquen su longitud previamente.
4. Virtual hosting (múltiples dominios sobre una misma ip).
5. Línea inicial en el requerimiento:  
GET /path/file.html HTTP/1.1
6. Nuevo campo de header obligatorio:  
Host: hostDeDestino.DominioDeDestino (el host virtual de destino)
7. El cliente pide requests sobre conexión persistente hasta el último recurso:  
GET /path/foto.gif HTTP/1.1
8. Connection: keep alive. (El requerimiento pide mantener la conexión mientras sea posible)
9. Connection:close. (El requerimiento lleva este header para pedir desconexión). Esto ocurre por ejemplo si el usuario cierra el navegador sin salir de la aplicación. El mismo navegador envía un nuevo requerimiento a todos los servidores con conexiones abiertas pidiendo a través de este header el cierre de la conexión al servidor remoto.
10. El servidor tiene capacidad para enviar un header de desconexión luego de un tiempo de inactividad.  
Aclaración → Un time Out (tiempo es especificado en el server para desconectar si no recibe nuevo requerimiento por parte del cliente.  
Existen requerimientos asincrónicos frente a un evento en el cliente (ajax) que también se producen sobre una conexión persistente.
11. El server puede terminar con la conexión (por ejemplo luego de un time out) con una respuesta conteniendo el siguiente campo header:  
Connection: close  
Significa que no aceptará más requerimientos.

# Directorios Virtuales y Host virtuales

## Directorios virtuales

URL → <http://www.servHostingXX.com.ar/~miempresa.midominio/mirecurso.html>

URI

- Un directorio virtual presenta los datos de un directorio real del disco del server por ejemplo → /home/miempresa
- No es necesario definir entradas de DNS en ningún servidor de nombres porque la IP es única y el nombre del servidor también lo es (por ej: telefonica.com.ar) en este caso particular.

## Hosts virtuales

URL → <http://miempresa.midominio/mirecurso.html>

URI

- Los servidores WEB que corren http 1.1 pueden atender múltiples dominios en una única IP.
- En el caso de usar hosts virtuales habrá que declarar el nombre de host en un servidor de DNS público. Por lo general quien hostea un recurso es un ISP que tiene su propia zona de nombres de dominio y solo tiene que agregarlo en su base de datos.
- El servidor WEB debe configurarse especialmente para atender host virtuales.
- Para simular servidores de DNS en una PC hay que editar el archivo de texto: `\winnt\system32\drivers\etc\hosts`