HTTP(Hypertext Transfer Protocol)

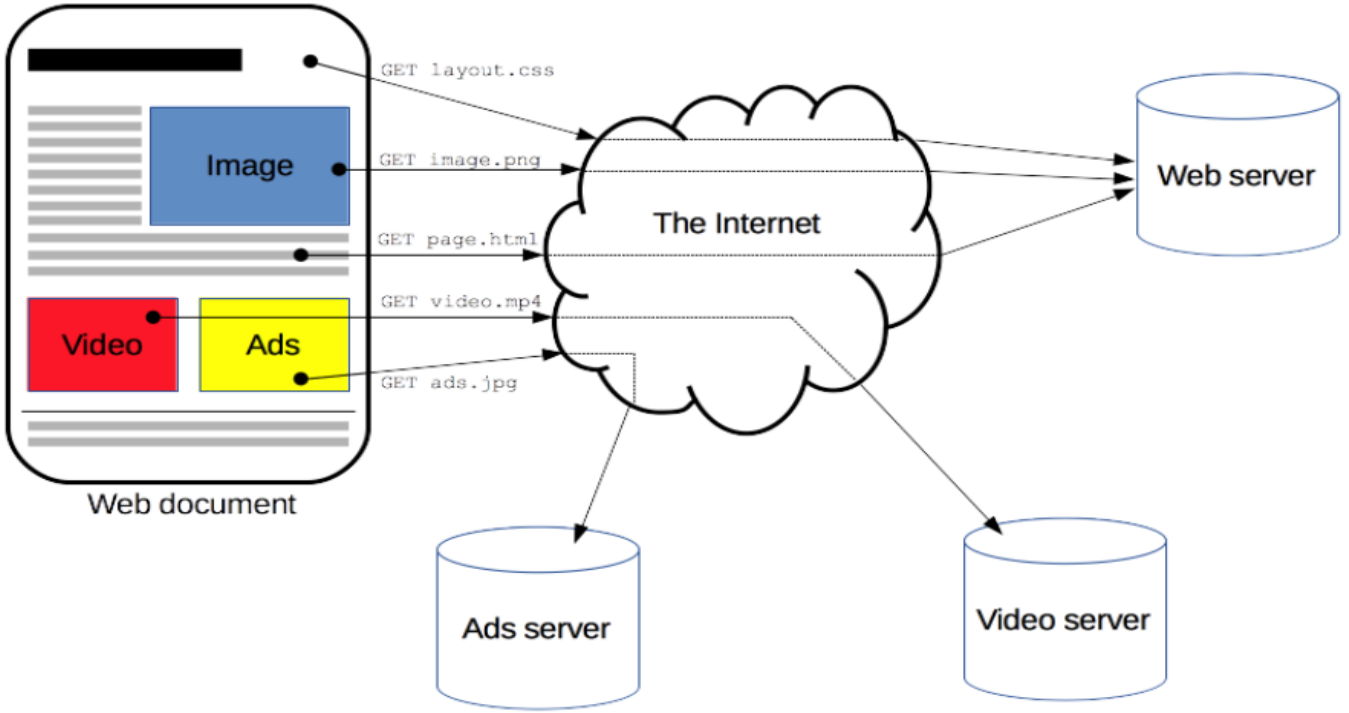
Agenda

1. Definición de HTTP
2. ¿Qué es un recurso?
3. Diferencias entre URL, URI y URN
4. Funcionamiento de HTTP
5. Versiones
6. ¿Qué se puede controlar con HTTP?
7. Flujo de HTTP
8. Mensajes HTTP
9. Verbos HTTP
10. Códigos de respuesta
11. Cabeceras
12. Parámetros
13. Lecturas recomendas

HTTP

Protocolo de Transferencia de Hipertexto, que consiste en un protocolo de comunicación entre sistemas de información que permite la transferencia de dados entre redes de computadores.

nos permite realizar una petición de datos y recursos, como pueden ser documentos HTML.



¿Qué es un recurso?

El objetivo de una solicitud HTTP se denomina "recurso", (es decir: datos), y dicho recurso, no posee un tipo definido por defecto; puede ser un documento, o una foto, o cualquier otra posibilidad. Cada recurso es identificado por un Identificador Uniforme de Recursos (URI) y es utilizado a través de HTTP, para la identificación del tipo de recurso.

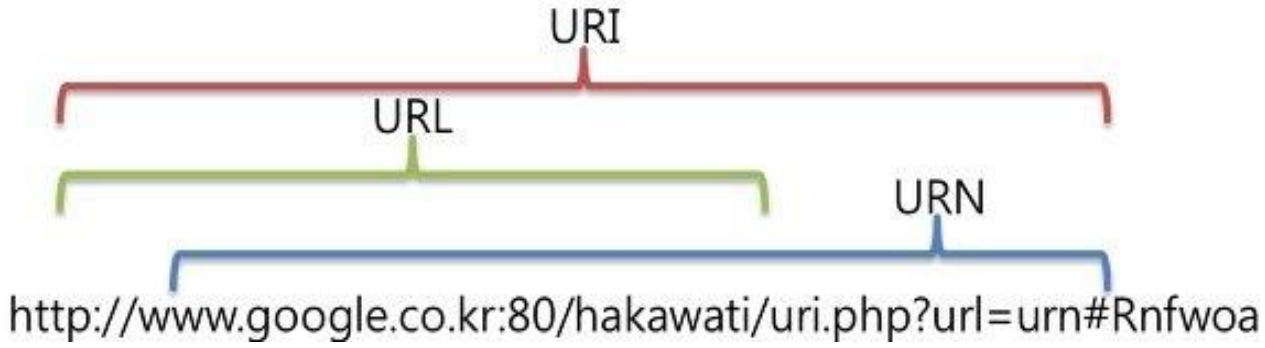
Diferencias entre URL, URI y URN

- URL significa Uniform Resource Locator (Localizador Uniforme de Recursos) que se utiliza para identificar un recurso, y es un subconjunto de URI. URI (Uniform Resource Identifier) ofrece una manera más simple y extensible de identificar un recurso.
- La URL y la URI se pueden diferenciar con el hecho de que la URI puede representar la URL y el URN del recurso al mismo tiempo, pero la URL sólo puede especificar la dirección del recurso. URI es un término más general en lugar de URL y URN que son más restrictivos en cierto sentido.

	URL	URI
ES	La URL proporciona un método para encontrar la identidad de un elemento.	URI se utiliza para definir la identidad de un elemento.
Sintaxis	https://www.pc-solucion.es/foto.jpg	public: //foto.jpg
Relación	Tipo de URI	Superconjunto de URL
Especificación del protocolo	Previsto	No se proporciona información de protocolo.

Diferencias entre URL, URI y URN

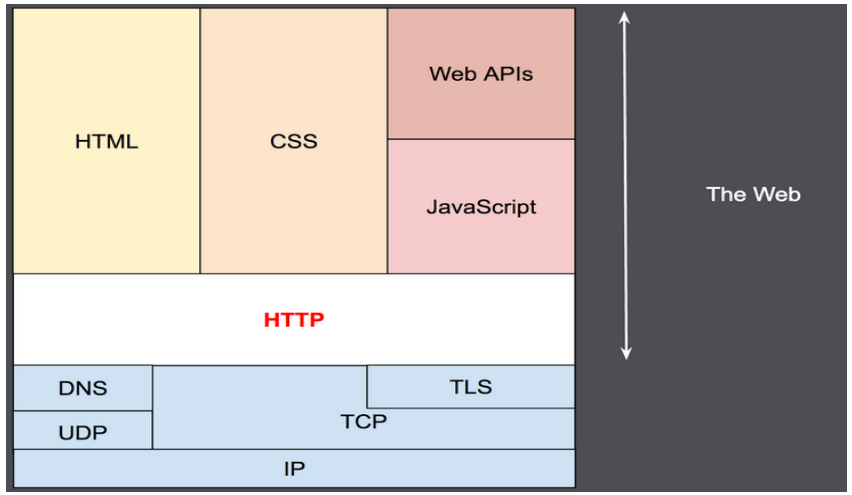
- Una URL (Uniform Resource Locator) se utiliza principalmente para apuntar a una página web, a un componente de una página web o a un programa en una página web con la ayuda del método de acceso (protocolos como http, ftp, mailto) para acceder a la ubicación del recurso. Por el contrario, URI (Uniform Resource Identifier) se utiliza para definir la identidad de un ítem aquí la palabra identificador significa distinguir un recurso de otro independientemente del método utilizado (URL o URN).
- Una URL es una URI, pero una URI nunca puede ser una URL.
- URL especifica qué tipo de protocolo se va a utilizar mientras que URI no incluye la especificación del protocolo.

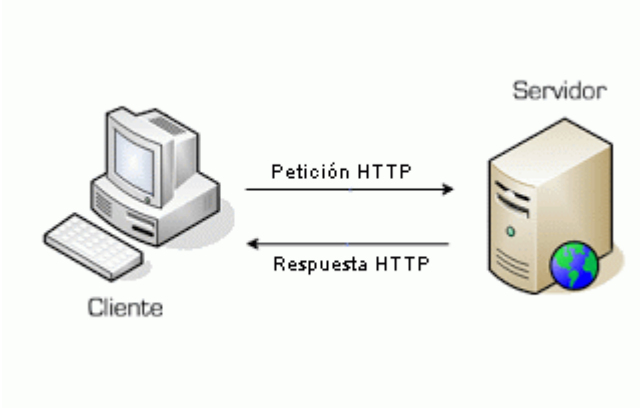


Funcionamiento de HTTP

- Clientes y servidores se comunican intercambiando mensajes individuales (En contraposición a las comunicaciones que utilizan flujos continuos de datos). Los mensajes que envía el cliente, normalmente un navegador Web (No siempre), se llaman peticiones y los mensajes enviados por el servidor se llaman respuestas. En este flujo la comunicación siempre las inicia el cliente.

- Http fue diseñado a principios de la década de 1990, HTTP es un protocolo ampliable, que ha ido evolucionando con el tiempo. Es lo que se conoce como un protocolo de la capa de aplicación y se transmite sobre el protocolo TCP, o el protocolo encriptado TLS, aunque teóricamente podría usarse cualquier otro protocolo fiable

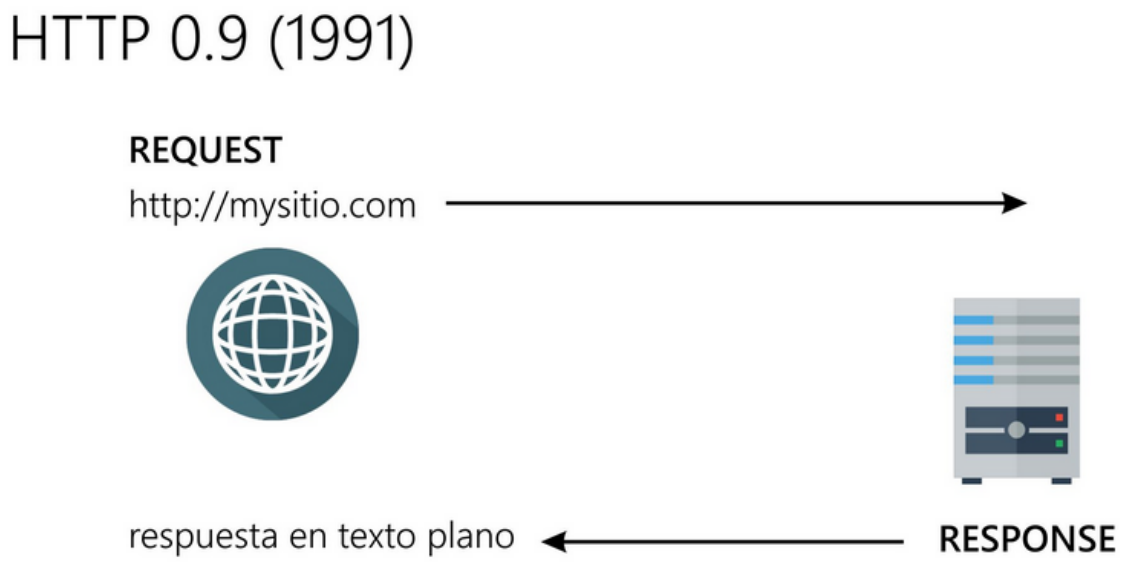




Versiones

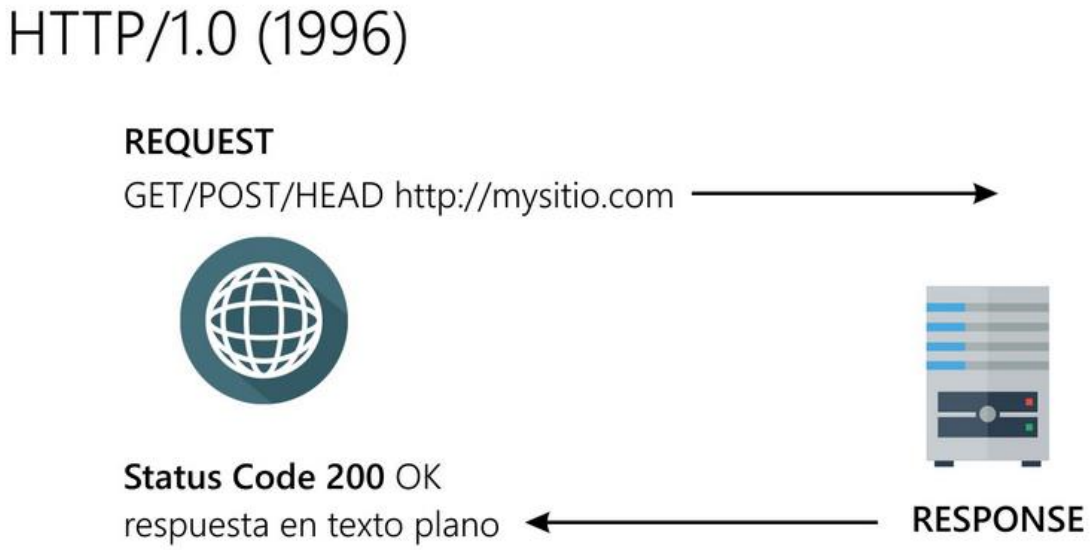
Este protocolo fue lanzado en 1991 y desde ahí ha ido evolucionando.

- Al inicio de la historia el protocolo HTTP solo permitía realizar peticiones sin especificar el verbo, es decir solo se podían hacer peticiones GET.



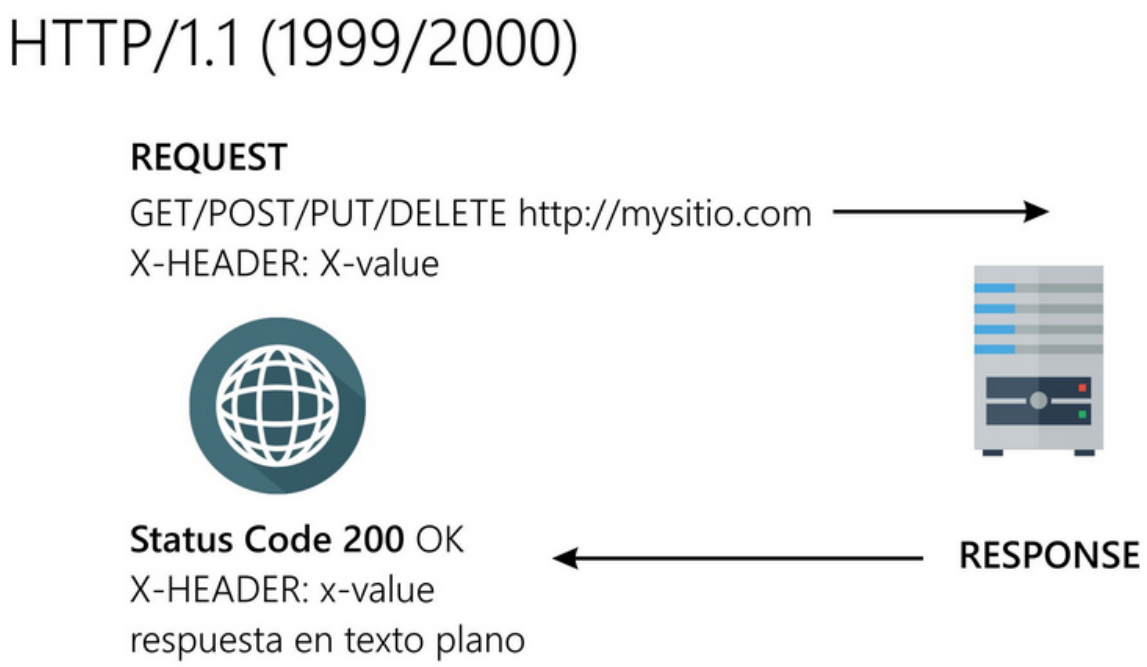
Versiones

- Luego se mejoraron las cosas y se agregó el soporte a algunos verbos como GET, POST y HEAD, se implementó los códigos de estado HTTP entre otras muchas mejoras.



Versiones

- Ya en la versión 1.1 teníamos los verbos GET,POST,PUT,DELETE,etc y la web se empezaba a orientar a recursos (REST), teníamos las cabeceras en las peticiones, etc.
- Sobre esta especificación se empezó a desarrollar HTTP1.2 pero luego termino convirtiéndose en una extensión de la versión 1.1, lo que tienen en común todas estas versiones es que tanto las respuestas como las peticiones se realizan a través de texto plano.



Versiones

- **Http/2(2015)**

HTTP 2.0 no modifica la semántica de aplicación de Http. Todos los conceptos básicos, tales como los métodos HTTP, códigos de estado, URI, y campos de cabecera, se mantienen sin cambios; sin embargo, HTTP 2.0 introduce innumerables mejoras como el uso de una única conexión, la compresión de cabeceras o el servicio 'server push'.

- **Http/3(2018)**

HTTP/3 es el sucesor propuesto de HTTP/2, que ya está en uso en la web, utilizando UDP en lugar de TCP para el protocolo de transporte subyacente. Al igual que el HTTP/2, no es obsoleto en las versiones principales anteriores del protocolo. El soporte para HTTP/3 fue agregado a Cloudflare y Google Chrome en septiembre de 2019,89 y puede ser habilitado en las versiones estables de Chrome y Firefox.

¿Qué se puede controlar con HTTP?

- La característica del protocolo HTTP de ser ampliable, ha permitido que durante su desarrollo se hayan implementado más funciones de control y funcionalidad sobre la web:
  - Cache

  El cómo se almacenan los documentos en la caché, puede ser especificado por HTTP.

  El servidor puede indicar a los proxies y clientes, que quiere almacenar y durante cuanto tiempo.

  - Flexibilidad del requisito de origen

  Para prevenir invasiones de la privacidad de los usuarios, los navegadores WEB, solamente permiten a página del mismo origen, compartir la información o datos. Esto es una complicación para el servidor, así que mediante cabeceras HTTP, se puede flexibilizar o relajar esta división entre el cliente y el servidor.

  - Autenticación

  Hay páginas WEB, que pueden estar protegidas, de manera que solo los usuarios autorizados puedan acceder. Http provee de servicios básicos de autenticación, por ejemplo mediante uso de cabeceras como: WWW-Authenticate, o estableciendo una sesión especifica mediante el uso de HTTP cookies.

¿Qué se puede controlar con HTTP?

- Proxies y tunneling

Servidores y/o clientes pueden estar en intranets y esconder así su verdadera dirección IP a otros. Las peticiones HTTP utilizan utilizan proxies para acceder a ellos.

- Sesiones

El uso de HTTP cookies permite relacionar peticiones con el estado del servidor. Esto define las sesiones, a pesar de que por definición el protocolo HTTP es un protocolo sin estado. Esto es muy útil no sólo para aplicaciones de comercio electrónica, sino también para cualquier sitio que permite configuración al usuario

Flujo de HTTP

- Cuando el cliente quiere comunicarse con el servidor, tanto si es directamente con él, o a través de un proxy intermedio, realiza los siguientes pasos:

1. Abre una conexión TCP:

La conexión TCP se usará para hacer una petición, o varias, y recibir la respuesta. El cliente puede abrir una conexión nueva, reusar una existencia, o abrir varias a la vez hacia al servidor.

2. Hacer una petición HTTP

Los mensajes HTTP(previos a HTTP/2) son legibles en texto plano. A partir de la versión del protoclo HTTP/2, los mensajes se encapsulan en franjas, haciendo que no sean directamente interpretables, aunque el principio es el mismo.

```
GET / HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
```

Flujo de HTTP

3. Leer la respuesta enviada por el servidor:

```
HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
ETag: "51142bc1-7449-479b075b2891b"
Accept-Ranges: bytes
Content-Length: 29769
Content-Type: text/html

<!DOCTYPE html... (here comes the 29769 bytes of the requested web page)
```

4. Cierre o reusó de la conexión para futuras peticiones:

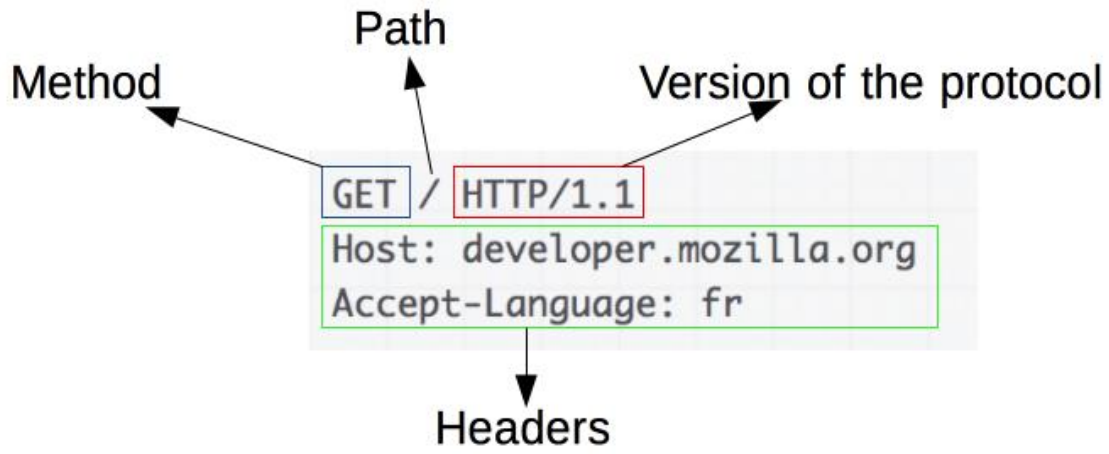
Si está activo el HTTP pipelining, varias peticiones pueden enviarse sin tener que esperar que la primera haya sido satisfecha. Este procedimiento es difícil de implementar en las redes de computadoras actuales, donde se mezclan software antiguo y moderno. Así que el HTTP pipelining ha sido sustituido en HTTP/2 por el multiplexado (En telecomunicación, la multiplexación es la técnica de combinar dos o más señales, y transmitirlas por un solo medio de transmisión) de varias peticiones.

Mensajes HTTP

En las versiones del protocoló HTTP/1.1 y anteriores los mensajes eran de formato texto y eran totalmente comprensibles directamente por una persona. En HTTP/2 los mensajes están estructurados en un nuevo formato binario y las tramas permiten la compresión de las cabeceras y su multiplexación.

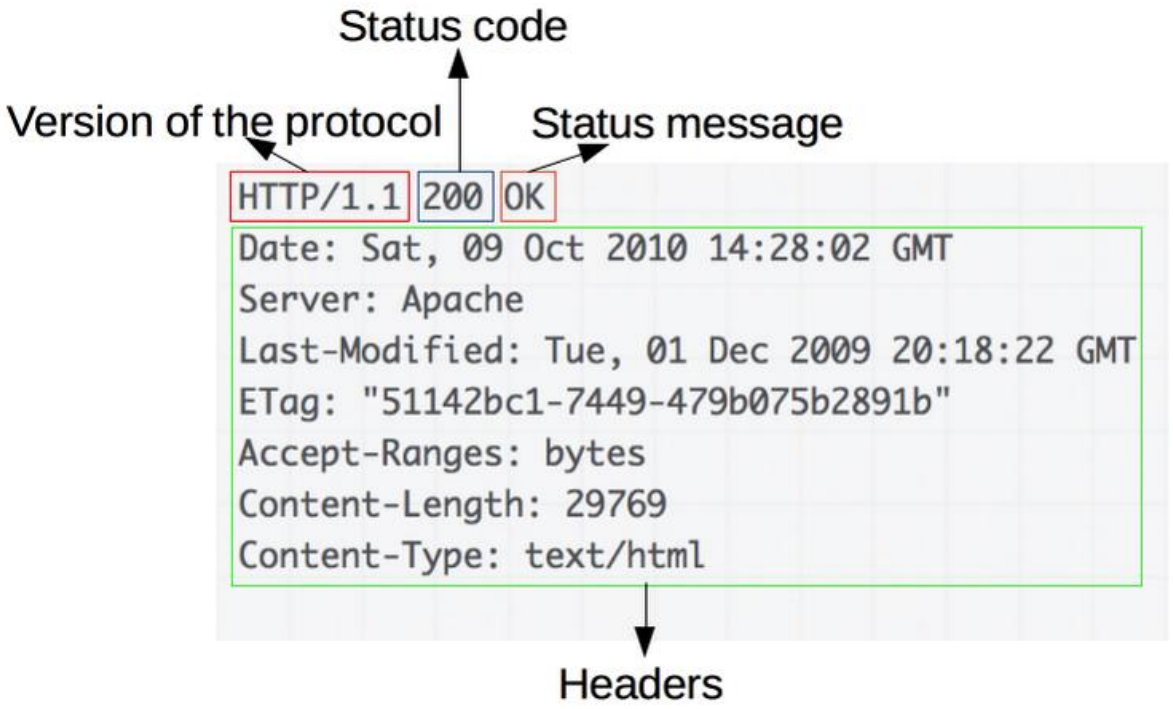
Existen dos tipos de mensajes HTTP: **peticiones(request)** y **respuestas(reponse)**, cada uno sigue su propio formato.

Peticiones



Mensajes HTTP

Respuestas



Verbos HTTP(Métodos HTTP)

Http define una serie predefinida de métodos de petición (algunas veces conocidos como como "**verbos**" http) que pueden utilizarse. El protocolo tiene flexibilidad para ir añadiendo nuevos métodos y así añadir nuevas funcionalidades. EL número de métodos de petición se ha ido aumentando según se avanza en el desarrollo del protocolo.

Cada método (o **verbo**) indica la acción que desea que se efectúe sobre el recurso identificado. Lo que este recurso representa depende de la aplicación del servidor. Por ejemplo, el recurso puede corresponderse con un archivo que reside en el servidor.

GET

El método GET solicita una representación del recurso especificado. Las solicitudes que usan GET solo deben recuperar datos y no deben tener ningún otro efecto. Muy parecidos a realizar un SELECT a la base de datos.

Verbos HTTP(Métodos HTTP)

HEAD

Pide una respuesta idéntica a la que correspondería a una petición GET, pero en la respuesta no se devuelve el cuerpo. Eso es útil para poder recuperar los metadatos de los encabezados de respuesta, sin tener que transportar el contenido..

POST

Envía datos para que sean procesados por el recurso identificado en la URL de la petición. Los datos se incluirán en el cuerpo de la petición. A nivel semántico está orientado a crear un nuevo recurso o modificarlo, la naturaleza o tipo de recurso se especifica por medio de la cabecera Content-Type. Es equivalente a realizar un INSERT en la base de datos.

- Para datos formularios codificados como una URL (aunque viajan en el cuerpo de la petición, no en la URL): application/x-www-form-urlencoded
- Para bloques a subir, ej. ficheros: multipart/form-data
- Además de los anteriores, no hay un estándar obligatorio y también podría ser otros como text/plain, application/json, application/octet-stream, etv.

Verbos HTTP(Métodos HTTP)

PUT

Envía datos al servidor, pero a diferencia del método POST la URI de la línea de petición no hace referencia al recurso que los procesará, sino que identifica a los propios datos. Otra diferencia con POST es semántica mientras que POST está orientado a la creación de nuevos contenidos, PUT está más orientado a la actualización de los mismos (aunque también podría crearlos).

El método PUT se debe utilizar cuando sea necesario hacer un reemplazo total de un recurso. Es decir, si tenemos una tabla Usuarios y queremos actualizar todos los datos de un usuario lo deberíamos implementar.

PUT va a esperar que los nuevos datos sean completamente iguales a los que se recibieron mediante un formulario. Es parecido a realizar un UPDATE a la base de datos.

Verbos HTTP(Métodos HTTP)

- **PATCH**

El método HTTP PATCH se utiliza para realizar una actualización parcial de un recurso. A diferencia del método PUT, el método PATCH no suprime las propiedades de recurso local que no están incluidas en la petición. Es parecido a realizar un UPDATE a la base de datos.

El método PATCH debe ser utilizado cuando se quiere modificar un solo campo. En este caso, si solo queremos actualizar el Email de uno de nuestros usuarios, PATCH sería el recomendado.

Verbos HTTP(Métodos HTTP)

DELETE

Borra el recurso especificado. Este método se utiliza para eliminar un registro existente, es similar a DELETE a la base de datos.

- Hasta aquí los métodos más utilizados, sin embargo, existen algunos métodos más que son interesantes conocer, pues no los encontraremos al momento de depurar o analizar el tráfico de red.
  - **CONNECT:** Se utiliza para establecer una comunicación bidireccional con el servidor. En la práctica no es necesario ejecutarlo, si no el mismo API de HTTP se encarga de ejecutarlo para establecer la comunicación previo a lanzar alguna solicitud al servidor.
  - **OPTIONS:** Este método es utilizado para describir las opciones de comunicación para el recurso de destino. Es muy utilizado con CORS (Cross-Origin Resource Sharing) para validar si el servidor acepta peticiones de diferentes origines. También devuelve los métodos HTTP que el servidor soporta para un URL específico.

Códigos de Respuesta

- El código de respuesta HTTP es un número que indica que ha pasado con la petición. El resto del contenido de la respuesta dependerá del valor de este código. El sistema es flexible y de hecho la lista de códigos ha ido aumentando para así adaptarse a los cambios e identificar nuevas situaciones.

- **Familias de códigos**
  - 1XX: Respuestas informativas. Indican que la petición ha sido recibida y se está procesando.
  - 2XX: Respuestas correctas. Indican que la petición ha sido procesada correctamente.
  - 3XX: Respuestas de redirección: Respuestas de redirección. Indica que el cliente necesita realizar más acciones para finalizar la petición. La redirección es el proceso utilizado para comunicar que un recurso ha sido trasladado a una nueva ubicación.
  - 4XX: Errores causados por el cliente: Indica que ha habido un error en el procesado de la petición a causa de que el cliente ha hecho algo mal.
  - 5XX: Errores causados por el servidor: Indica que ha habido un error en el procesado de la petición a causa de un fallo en el servidor.

Códigos HTTP 1XX

- **100**: «Continue(Continuar)». Esto significa que el servidor en cuestión ha recibido las cabeceras de solicitud de tu navegador, y ahora está listo para que el cuerpo de la solicitud sea enviado también. Esto hace que el proceso de solicitud sea más eficiente ya que evita que el navegador envíe una solicitud de cuerpo, aunque los encabezados hayan sido rechazados.

- **101**: «Switching Protocols(Cambiando protocolos)». Tu navegador ha pedido al servidor que cambie los protocolos, y el servidor ha cumplido.

- **103**: «Checkpoint(Primeros avisos)». Esto devuelve algunos encabezados de respuesta antes de que el resto de la respuesta del servidor esté lista.

Códigos HTTP 2XX

- **200**: «Ok (Todo está bien)». Este es el código que se entrega cuando una página web o recurso actúa exactamente como se espera.

- **201**: «Created (Creado)». El servidor ha cumplido con la petición del navegador y, como resultado, ha creado un nuevo recurso.

- **202**: «Accepted (Aceptado)». El servidor ha aceptado la solicitud, pero aún la está procesando. La solicitud puede, en última instancia, dar lugar o no a una respuesta completa.

- **203**: «Non-Authoritative Information (Información no autorizada)». Este código de estado puede aparecer cuando se utiliza un apoderado. Significa que el servidor proxy recibió un código de estado de 200 «Todo está bien» del servidor de origen, pero ha modificado la respuesta antes de pasarla a su navegador.

Códigos HTTP 2XX

- **204**: «No Content (Sin contenido)». Este código significa que el servidor ha procesado con éxito la solicitud, pero no va a devolver ningún contenido.

- **205**: «Reset Content (Restablecer el contenido)». Como un código 204, esto significa que el servidor ha procesado la solicitud, pero no va a devolver ningún contenido. Sin embargo, también requiere que tu navegador restablezca la vista del documento.

- **206**: «Partial Content (Contenido parcial)». Puedes ver este código de estado si tu cliente HTTP (también conocido como tu navegador) usa «cabeceras de rango». Esto permite a tu navegador reanudar las descargas en pausa, así como dividir una descarga en múltiples flujos. Se envía un código 206 cuando un encabezado de rango hace que el servidor envíe sólo una parte del recurso solicitado.

Códigos HTTP 3XX

- **300**: «Multiple Choices (Opciones Múltiples)». A veces, puede haber múltiples recursos posibles con los que el servidor puede responder para cumplir con la solicitud de su navegador. Un código de estado 300 significa que tu navegador ahora tiene que elegir entre ellos. Esto puede ocurrir cuando hay múltiples extensiones de tipo de archivo disponibles, o si el servidor está experimentando desambiguación del sentido de las palabras.

- **301**: «Moved Permanently (El recurso solicitado ha sido trasladado permanentemente)». Este código se entrega cuando una página web o un recurso ha sido reemplazado permanentemente por un recurso diferente. Se utiliza para la redirección permanente del URL.

- **302**: «Found (antes "Moved Temporarily") (El recurso solicitado se ha movido, pero fue encontrado)». Este código se utiliza para indicar que el recurso solicitado se encontró, pero no en el lugar donde se esperaba. Se utiliza para la redirección temporal de la URL.

Códigos HTTP 4XX

- **400**: «Bad Request (Mala petición)». El servidor no puede devolver una respuesta debido a un error del cliente (Por ejemplo, envío de datos que no cumplen con validaciones).
- **401**: «Unauthorized (No autorizado)» o «Se requiere autorización». Esto Similar al 403 Forbidden, pero específicamente para su uso cuando la autentificación es posible, pero ha fallado o aún no ha sido provista.

- **402**: «Payment Required (Pago requerido)». Originalmente, este código fue creado para ser usado como parte de un sistema de dinero digital. Sin embargo, ese plan nunca se llevó a cabo. En cambio, es utilizado por diversas plataformas para indicar que una solicitud no se puede cumplir, por lo general debido a la falta de los fondos necesarios. Los casos más comunes incluyen:
  - Has alcanzado el límite de solicitudes diarias al API de los desarrolladores de Google.
  - No ha pagado tus honorarios de Shopify y su tienda ha sido desactivada temporalmente.
  - Tu pago a través de Stripe ha fallado, o Stripe está tratando de evitar un pago fraudulento.

Códigos HTTP 4XX

- **403**: «Forbidden (El acceso a ese recurso está prohibido)». Este código se devuelve cuando un usuario intenta acceder a algo a que no tiene permiso para ver. Por ejemplo, intentar acceder a un contenido protegido por contraseña sin registrarse podría producir un error 403.

- **404**: «Not Found (No se encontró el recurso solicitado)». Este es el mensaje de error más común de todos ellos. Este código significa que el recurso solicitado no existe, y el servidor no sabe si alguna vez existió.

- **405**: «Method Not Awolled (Método no permitido)». Esto se genera cuando el servidor de alojamiento (servidor de origen) soporta el método recibido, pero el recurso de destino no lo hace.

- **406**: «Not Acceptable (Respuesta no aceptable)». El recurso solicitado es capaz de generar sólo contenido que no es aceptable según los encabezamientos de aceptación enviados en la solicitud.

Códigos HTTP 4XX

- **408**: «Request Timeout (El servidor se agotó esperando el resto de la petición del navegador)». Este código se genera cuando un servidor se apaga mientras espera la solicitud completa del navegador. En otras palabras, el servidor no recibió la solicitud completa que fue enviada por el navegador. Una posible causa podría ser la saturación de la red, lo que provoca la pérdida de paquetes de datos entre el navegador y el servidor.

Códigos HTTP 5XX

- **500:** «Internal Server Error (Hubo un error en el servidor y la solicitud no pudo ser completada)». Este es un código genérico que simplemente significa «error interno del servidor». Algo salió mal en el servidor y el recurso solicitado no fue entregado. Este código es típicamente generado por plugins de terceros, PHP defectuoso, o incluso la ruptura de la conexión a la base de datos.

- **501:** «Not Implemented (No implementado)». Este error indica que el servidor no es compatible con la funcionalidad necesaria para cumplir con la solicitud. Esto es casi siempre un problema en el propio servidor web, y por lo general debe ser resuelto por el host.

- **502:** «Bad Gateway (Mala entrada)». Este código de error significa típicamente que un servidor ha recibido una respuesta inválida de otro, como cuando se utiliza un servidor proxy. Otras veces una consulta o petición tardará demasiado, y así es cancelada o asesinada por el servidor y la conexión a la base de datos se rompe.

Códigos HTTP 5XX

- **503:** «Service Unavailable (El servidor no está disponible para manejar esta solicitud en este momento)». La solicitud no puede ser completada en este momento. Este código puede ser devuelto por un servidor sobrecargado que no puede manejar solicitudes adicionales.

- **504:** «Gateway Timeout (El servidor, actuando como una puerta de enlace, se ha agotado esperando a que otro servidor responda)». Este es el código devuelto cuando hay dos servidores involucrados en el procesamiento de una solicitud, y el primer servidor se apaga esperando que el segundo servidor responda.

- **505:** «HTTP Version Not Supported (Versión HTTP no soportada)». El servidor no soporta la versión HTTP que el cliente usó para hacer la solicitud.

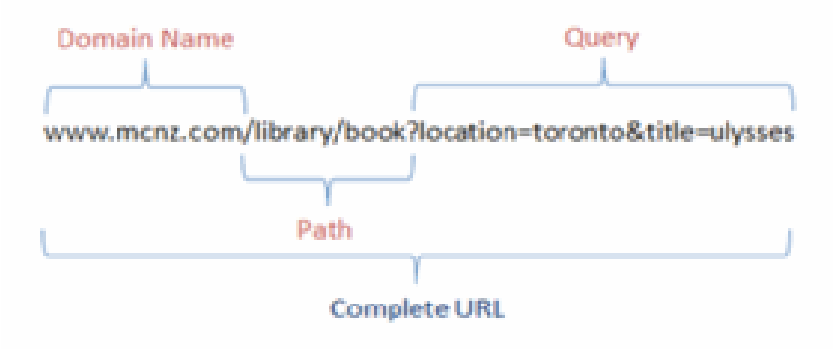
Cabeceras(Headers)

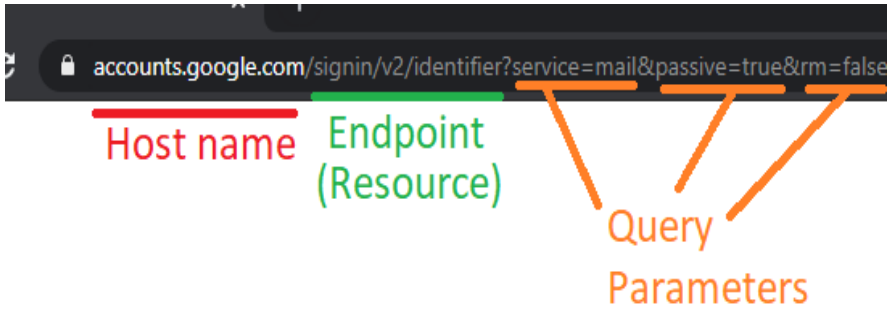
Las Cabeceras HTTP son los parámetros que se envían en una petición o respuesta HTTP al cliente o al servidor para proporcionar información esencial sobre la transacción en curso. Estas cabeceras proporcionan información mediante la sintaxis 'Cabecera: Valor' y son enviadas automáticamente por el navegador o el servidor WEB.

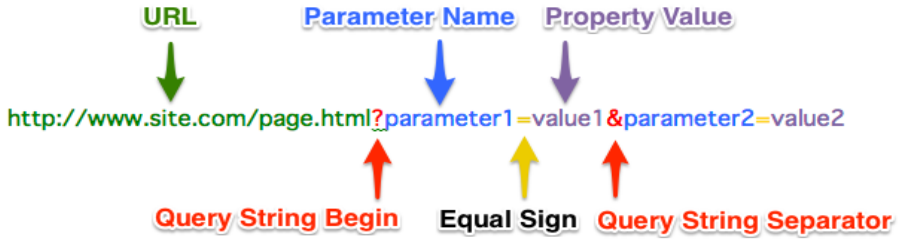
Campo de cabecera	Significado	Ejemplo
Accept	Qué tipos de contenido puede procesar el cliente; si el campo está vacío, esos son todos los tipos de contenido	Accept: text/html, application/xml
Accept-Charset	Qué caracteres puede mostrar el cliente	Accept-Charset: utf-8
Accept-Encoding	Qué formatos comprimidos soporta el cliente	Accept-Encoding: gzip
Accept-Language	Preferencia de idioma	Accept-Language: de-DE
Authorization	Datos de autenticación (por ejemplo, para un inicio de sesión)	Basic WjbU7D25zTAlV2tZ7==
Cache-Control	Opciones de mecanismo de caching	Cache-Control: no-cache
Cookie	Cookies almacenadas para este servidor	Cookie: $Version=1; Content=23
Content-Length	Longitud del cuerpo de la solicitud	Content-Length: 212
Content-Type	Tipo MIME del cuerpo; pertinente para las solicitudes POST y PUT	Content-Type: application/x_222-form-urlencoded
Date	Fecha y hora de la solicitud	Date: Mon, 9 March 2020 09:02:22 GMT
Expect	Formula una expectativa al servidor, generalmente la recepción de una solicitud grande	Expect: 100-continue (El servidor debe enviar el código 100 cuando esté preparado para recibir una solicitud)
Host	Nombre de dominio del servidor	Host: ejemplo.es
If-Match	Permite la ejecución condicional de una acción, dependiendo de la coincidencia de un código transmitido	If-Match: „ft678iujhnjio90'pöl“

Parámetros en URL

- Se pueden agregar dos tipos de parámetros de PATH y de QUERY, la principal diferencia es que los de PATH son obligatorios ya forman parte del URL en si, mientras que los de QUERY son opcionales.
  - Parámetros de Query

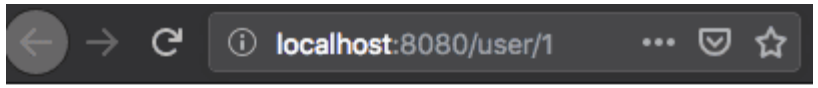






Parámetros en URL

- Parámetros de Path



My Vue App

Home User

User with id 1

Lecturas recomendas

https://developer.mozilla.org/es/docs/Web/HTTP/Overview

https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/protocolo-http/

https://developer.mozilla.org/es/docs/Web/HTTP/Headers

https://developer.mozilla.org/es/docs/Web/HTTP/Status