educación IT
CODE YOUR FUTURE

Bootcamp Java Developer

Fase 3 - Java Architect

Módulo 29

educación IT
CODE YOUR FUTURE

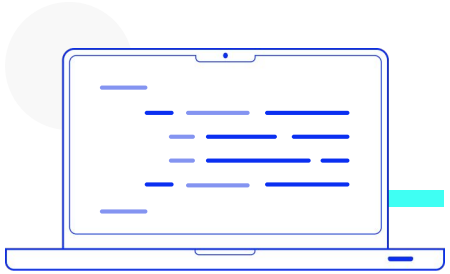
Protocolo HTTP

El protocolo HTTP

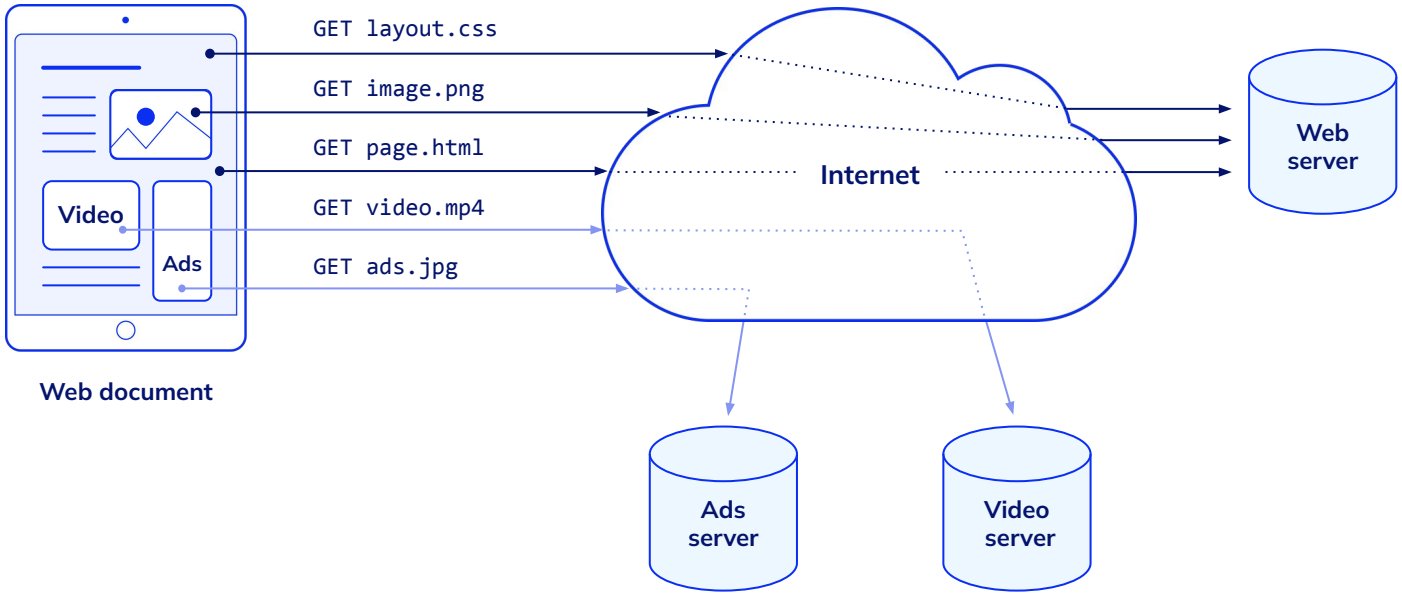
Hypertext Transfer Protocol es el nombre del protocolo que permite realizar una **petición de datos o recursos**, como documentos HTML, archivos PDF, imágenes, y otros.

Es la base de cualquier intercambio de datos en la web y tiene una estructura cliente-servidor. Esto quiere decir que una petición de datos es iniciada por el elemento que recibirá los datos (el cliente), que normalmente es un navegador Web.

Nota: una **página web completa resulta de la unión de distintos sub-documentos** recibidos, como por ejemplo: un documento que especifique el estilo de la página web (CSS), el texto, imágenes, vídeos, *scripts*.



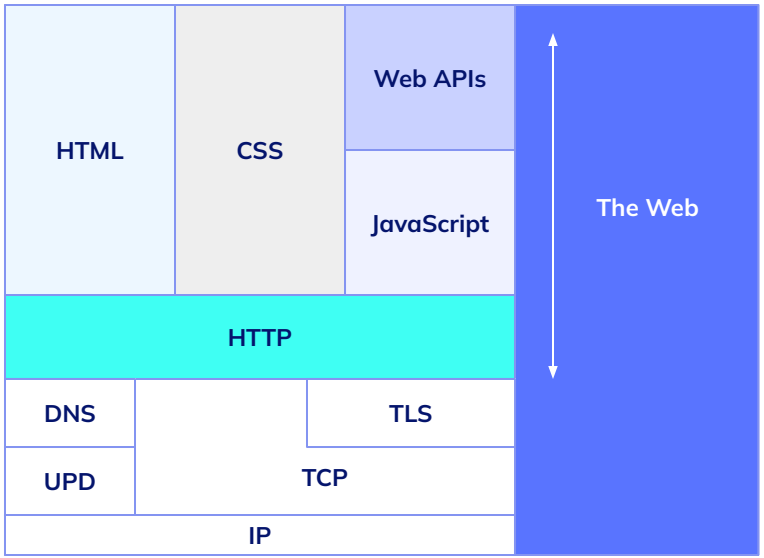
Protocolo HTTP



Peticiones y respuestas

Clientes y servidores se comunican intercambiando mensajes individuales (en contraposición a las comunicaciones que usan flujos continuos de datos).

Los mensajes que envía el cliente, normalmente un navegador web, se llaman peticiones (*Request*), y los mensajes enviados por el servidor, se llaman respuestas (*Response*).



Diseñado a principios de los 90, 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**.

Gracias a que es un protocolo capaz de ampliarse, se usa no solo para transmitir documentos de hipertexto (HTML), sino que además, se usa para transmitir imágenes o vídeos, o enviar datos o contenido a los servidores, como en el caso de los formularios de datos. **HTTP puede incluso ser utilizado para transmitir partes de documentos, y actualizar páginas Web en el acto**.

Características clave del protocolo HTTP

- Sencillo
- Extensible
- Con sesiones pero sin estados

Sencillo

Está pensado y desarrollado para ser **leído y fácilmente interpretado por las personas**. Esta característica hace que sea más fácil la depuración de errores, y reduce la curva de aprendizaje para las personas que empiezan a trabajar con él.

Extensible

Las cabeceras de HTTP, presentadas en la versión HTTP/1.0, han hecho que este protocolo sea **fácil de ampliar** y se pueda experimentar con él. Pueden desarrollarse **funcionalidades nuevas**, sin más que un cliente y su servidor, comprendan la misma semántica de las cabeceras de HTTP.

Con sesiones, pero sin estados

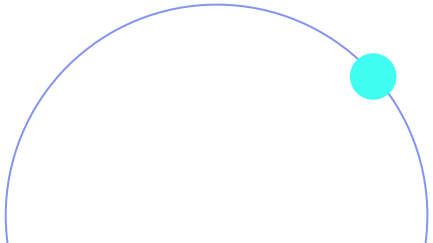
HTTP es un protocolo sin estado, es decir: **no guarda ningún dato entre dos peticiones en la misma sesión**. Esto plantea una problemática, en caso de que los usuarios requieran interactuar con determinadas páginas web de forma ordenada y coherente, por ejemplo, para el uso de "cestas de la compra" en páginas que utilizan en comercio electrónico. Pero, **mientras HTTP ciertamente es un protocolo sin estado, el uso de HTTP *cookies*, sí permite guardar datos con respecto a la sesión de comunicación. Usando la capacidad de ampliación del protocolo HTTP, las *cookies* permiten crear un contexto común para cada sesión de comunicación**.

HTTP y conexiones

Una conexión se gestiona al nivel de la capa de transporte, y por tanto queda fuera del alcance del protocolo HTTP. Aún con este factor, HTTP no necesita que el protocolo que lo sustenta mantenga una conexión continua entre los participantes en la comunicación, solamente necesita que sea un protocolo fiable o que no pierda mensajes (como mínimo, en todo caso, un protocolo que sea capaz de detectar que se ha pedido un mensaje y reporte un error).

De los dos protocolos más comunes en Internet, TCP es fiable, mientras que UDP, no lo es. Por lo tanto, HTTP se apoya en el uso del protocolo TCP, que está orientado a conexión, aunque una conexión continua no es necesaria siempre.

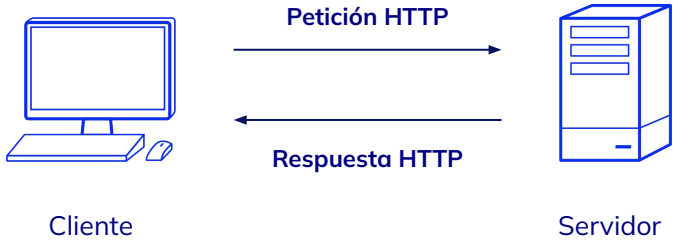
> Fuente: Developer.mozilla.org.



Mensajes HTTP

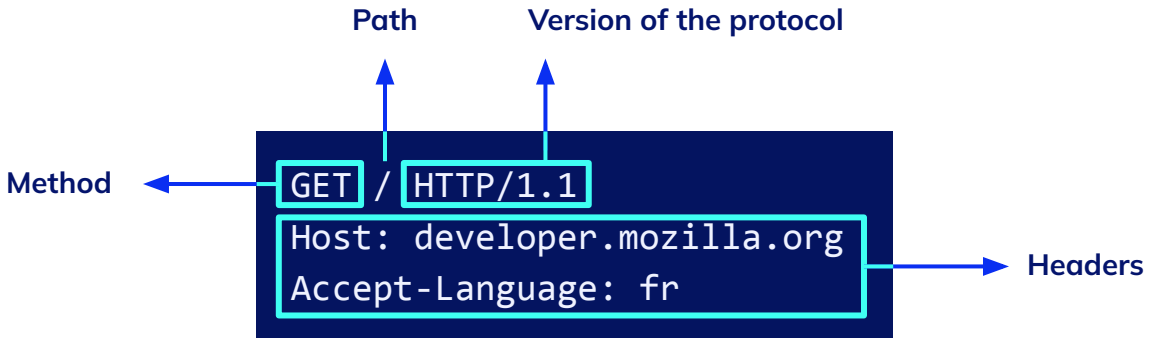
Están estructurados en un formato binario y las tramas permiten la compresión de las cabeceras y su multiplexación. Así pues, incluso si solamente parte del mensaje original en HTTP se envía en este formato, la semítica de cada mensaje es la misma y el cliente puede formar el mensaje original en HTTP/1.1. Luego, es posible interpretar los mensajes HTTP/2 en el formato de HTTP/1.1.

Como se vio anteriormente, existen **dos tipos de mensajes HTTP: peticiones** y **respuestas**, cada uno sigue su propio formato. Profundizaremos en cada uno de ellos, en las próximas pantallas.



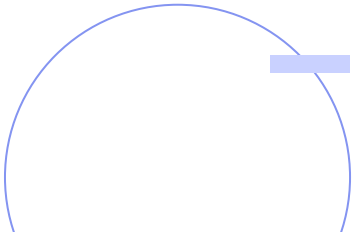
Peticiones (Request)

Ejemplo:



Una petición de HTTP, está formada por los siguientes campos:

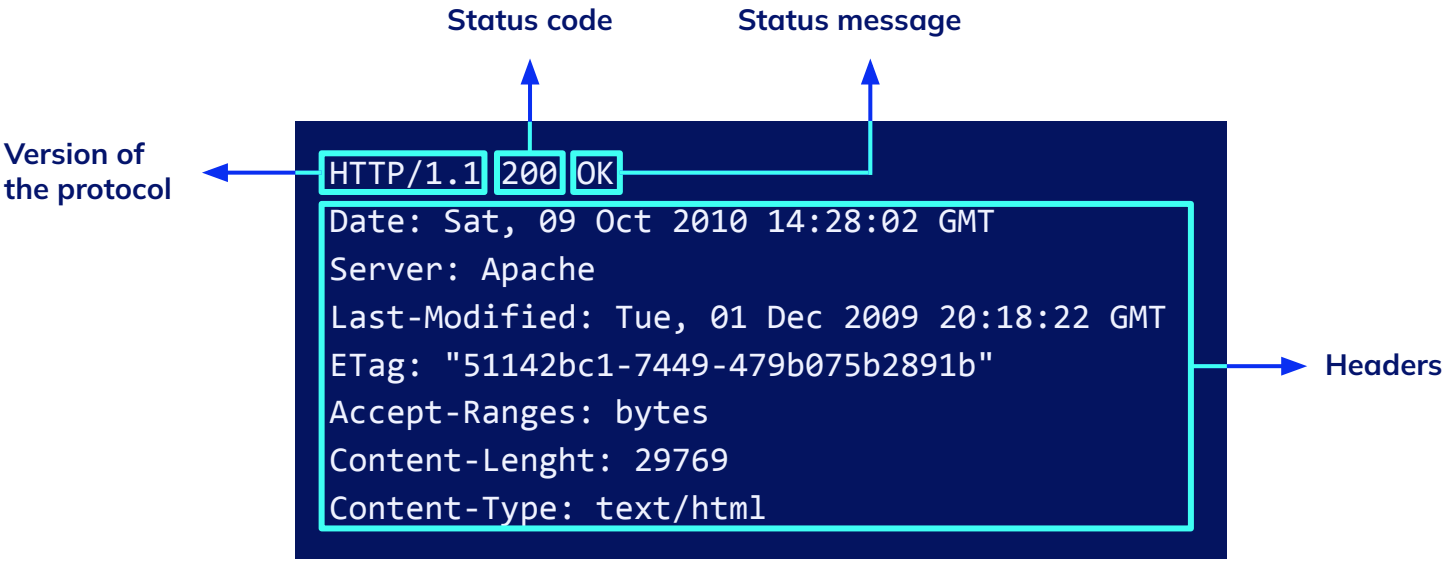
- **Un método HTTP:** normalmente puede ser un verbo, como: `GET`, `POST` o un nombre como: `OPTIONS` o `HEAD`, que defina la operación que el cliente quiera realizar. El objetivo de un cliente, suele ser una petición de recursos, usando `GET`, o presentar un valor de un formulario HTML, usando `POST`, aunque en otras ocasiones puede hacer otros tipos de peticiones.



- **La dirección del recurso pedido:** la URL del recurso, sin los elementos obvios por el contexto, como pueden ser: sin el protocolo (`http://`), el dominio (aquí `developer.mozilla.org`), o el puerto TCP (aquí el `80`).
- **La versión del protocolo HTTP**.
- **Cabeceras HTTP opcionales:** pueden aportar información adicional a los servidores.
- **Un cuerpo de mensaje, en algún método:** como `POST`, en el que envía la información para el servidor.

Respuestas (Response)

Ejemplo:



Las respuestas HTTP están formadas por los siguientes campos:

- **La versión del protocolo HTTP** que están usando.
- **Un código de estado**, indicando si la petición ha sido exitosa, o no.
- **Un mensaje de estado**, una breve descripción del código de estado.
- **Cabeceras HTTP**, como las de las peticiones.
- Opcionalmente, **el recurso** que se ha pedido**.**

¡Sigamos trabajando!