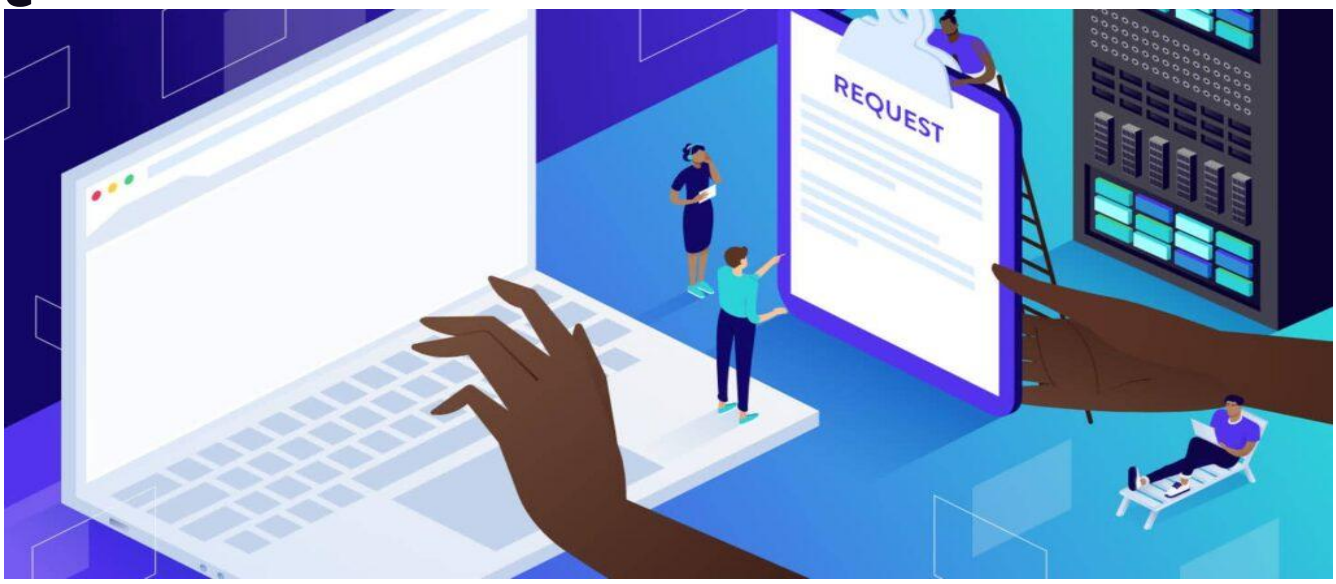
Carrera: Telecomunicaciones **Materia: Arquitectura y Conectividad**
Docente: Jorge Morales
Alumno: Emma Gutiérrez

¿Qué es una Petición HTTP?



Cuando se visita un sitio web, tu navegador realiza docenas o cientos de peticiones a su servidor en segundo plano. El servidor responde a esas peticiones entregando todos los datos y archivos que el sitio necesita cargar. Sin embargo, el proceso real es más complejo que eso.
Entender cómo funcionan las peticiones HTTP es esencial si quieres aumentar el rendimiento de tu sitio web. Algunas de estas medidas de optimización implican minimizar y comprimir las peticiones. Básicamente, estás optimizando tu servidor para que responda mejor a las peticiones HTTP.

Cómo funcionan las peticiones HTTP. Y cual es la estructura de las peticiones HTTP y cómo solucionar los problemas si es necesario.

1. ¿Qué es el HTTP?
2. ¿Qué es una Petición HTTP (y Cómo Funciona)?
3. Introducción a las Estructuras de las Peticiones y Respuestas HTTP
4. Cómo Supervisar y Solucionar los Problemas de las Peticiones HTTP

¿Qué es el HTTP?

HTTP es un protocolo. De hecho, el acrónimo significa Protocolo de Transferencia de Hipertexto. Este protocolo rige la estructura y el lenguaje de las peticiones y respuestas que tienen lugar entre clientes y servidores. Los clientes suelen ser los navegadores web, pero pueden presentarse de muchas formas, como los robots de los motores de búsqueda.
Cuando visitas sitios web a través de un navegador, toda la conexión tiene lugar a través de HTTP. El protocolo te permite recibir datos, incluyendo texto, imágenes, vídeos, hojas de estilo, scripts, etc.
HTTP ha sido una de las columnas vertebrales de la web desde principios de los 90. En las últimas décadas, ha evolucionado para ser más eficiente. En la segunda mitad de la década de 2010 se desarrolló HTTP/2, que permite a los clientes cargar los recursos de forma simultánea en lugar de asíncrona. Esto da lugar a un aumento masivo del rendimiento.
En 2022, el 46% de la web utilizará HTTP/2. Ahora, ya se habla de adoptar HTTP/3, que también se conoce como HTTP-sobre-QUIC. HTTP/3 funciona con el protocolo UDP, lo que le da una ventaja sobre las conexiones TCP tradicionales (que es lo que utilizan HTTP y HTTP/2).

Carrera: Telecomunicaciones **Materia: Arquitectura y Conectividad**
Docente: Jorge Morales
Alumno: Emma Gutiérrez

¿Qué es una Petición HTTP y Cómo Funciona?

Piensa en una petición HTTP como si tu navegador se conectara al servidor y le pidiera un recurso específico o le enviara datos. Hay varios tipos de métodos de petición HTTP, que modifican completamente el tipo de respuesta que obtienes del servidor. Los más comunes son:

GET. Es el método de petición HTTP más utilizado con diferencia. Una petición GET solicita al servidor una información o recurso concreto. Cuando te conectas a un sitio web, tu navegador suele enviar varias peticiones GET para recibir los datos que necesita para cargar la página.

HEAD. Con una petición HEAD, sólo recibes la información de la cabecera de la página que quieres cargar. Puedes utilizar este tipo de petición HTTP para conocer el tamaño de un documento antes de descargarlo mediante GET.

POST. Tu navegador utiliza el método de petición HTTP POST cuando necesita enviar datos al servidor. Por ejemplo, si rellenas un formulario de contacto en un sitio web y lo envías, estás utilizando una petición POST para que el servidor reciba esa información.

PUT. Las peticiones PUT tienen una funcionalidad similar a la del método POST. Sin embargo, en lugar de enviar datos, utilizas las peticiones PUT para actualizar información que ya existe en el servidor final.

Hay otros tipos de peticiones HTTP que puedes utilizar, como los métodos DELETE, PATCH y OPTIONS. Sin embargo, son relativamente poco frecuentes en el uso cotidiano.

Presentar una solicitud HTTP implica enviar un mensaje al servidor receptor en un formato específico. El servidor devuelve una respuesta y el cliente realiza la acción correspondiente. Por ejemplo, puede cargar recursos o redirigirte a otra página.

Cuando obtienes un error HTTP, suele ser porque el servidor no puede satisfacer tu petición. El código de error que obtienes debería explicar el motivo. Algunas de las causas más comunes de los errores HTTP son la imposibilidad de conectarse al servidor y encontrar los recursos que se han solicitado.

Prueba nuestra herramienta HTTP header Checker para revisar el estado de cualquier página.

Introducción a las Estructuras de las Peticiones y Respuestas HTTP

Las peticiones y respuestas HTTP comparten estructuras similares. Si quieres ser capaz de analizar las peticiones y respuestas HTTP para entender los posibles errores del sitio, es importante que entiendas esas estructuras.

En general, las peticiones HTTP se dividen en tres secciones. Echemos un vistazo a cada una de ellas.

Carrera: Telecomunicaciones **Materia: Arquitectura y Conectividad**
Docente: Jorge Morales
Alumno: Emma Gutiérrez

Línea de Solicitud HTTP

Toda petición HTTP comienza con una línea que indica el tipo de método que estás utilizando y la versión del protocolo HTTP. Por ejemplo, el inicio de una petición HTTP GET podría ser así

```
GET /XXX HTTP/1.1
```

En este caso, el parámetro «XXX» después del método GET indica el archivo que quieres recibir.
El inicio de una respuesta HTTP reitera la versión del protocolo que ambas partes están utilizando. También incluye un código HTTP que corresponde al estado de la respuesta.
La visita a un sitio web y se carga con éxito, verás un mensaje de éxito HTTP 2XX:
HTTP/1.1 200 OK
Esta parte de la respuesta HTTP mostrará códigos de error si el recurso no se carga por cualquier motivo. Si el servidor no puede encontrar la página, verás una cabecera de respuesta como ésta:
HTTP/1.1 400 OK
Se entiende los métodos de solicitud y los códigos de estado HTTP,

Las cinco clases incluyen:

- **100s: Códigos informativos que indican que la solicitud iniciada por el navegador continúa.**
- **200s: Los códigos con éxito regresaron cuando la solicitud del navegador fue recibida, entendida y procesada por el servidor.**
- **300s: Códigos de redireccionamiento devueltos cuando un nuevo recurso ha sido sustituido por el recurso solicitado.**
- **400s: Códigos de error del cliente que indican que hubo un problema con la solicitud.**
- **500s: Códigos de error del servidor que indican que la solicitud fue aceptada, pero que un error en el servidor impidió que se cumpliera**.

La línea de salida te indica exactamente qué tipo de transacción se está realizando entre el cliente y el servidor. En general, ésta es la parte de la solicitud más sencilla de entender.

Cabeceras de Solicitud
Las cabeceras de solicitud vienen justo después de las líneas de solicitud y proporcionan información adicional sobre la transacción. La cabecera especifica información sobre el host, el software del servidor web que utiliza el cliente final, cuál es el agente de usuario del cliente, etc.
Este es el aspecto de una cabecera de solicitud HTTP:
Host: website.com

User-Agent: Chrome/5.0 (Windows 10)

Accept-Language: en-US

Accept-Encoding: gzip, deflate

Connection: keep-alive</code.

Carrera: Telecomunicaciones **Materia: Arquitectura y Conectividad**
Docente: Jorge Morales
Alumno: Emma Gutiérrez

Estos son sólo algunos ejemplos de los parámetros de cabecera HTTP que puedes utilizar. Esto es lo que significa cada línea de esa cabecera:

Host: Es la IP o la URL del servidor al que haces la petición.
User-agent: Este parámetro contiene información sobre el cliente y su Sistema Operativo (SO). Normalmente, indica el navegador que está utilizando y su versión.
Accept-language: Esta línea indica al servidor qué idioma prefiere el cliente, en caso de que haya varias versiones del archivo que estás solicitando.
Accept-encoding: Esta línea indica el tipo de codificación o compresión que el cliente puede procesar.
Connection: Este parámetro indica al servidor si debe mantener viva la conexión o establecer un tiempo de espera para la misma. Si la conexión se agota antes de completar la solicitud, recibirás un error.
Si se junta la línea de solicitud y las cabeceras para hacernos una idea de la estructura general que tendrás que utilizar:

```
GET /XXX HTTP/1.1

Host: website.com

User-Agent: Chrome/5.0 (Windows 10)

Accept-Language: en-US

Accept-Encoding: gzip, deflate

Connection: keep-alive</code.
```

En el ejemplo anterior, estás enviando una solicitud **GET** al host website.com para un recurso específico. Ahora, veamos cómo podría ser la cabecera en la respuesta:

```
HTTP/1.1 200 OK

Date: Mon, 27 Jul 2022 12:28:53 GMT

Server: Apache/2.2.14 (Win32)

Last-Modified: Wed, 22 Jul 2022 19:15:56 GMT

Content-Length: 88

Content-Type: text/html

Connection: Closed</code.
```

Carrera: Telecomunicaciones **Materia: Arquitectura y Conectividad**
Docente: Jorge Morales
Alumno: Emma Gutiérrez

La cabecera de la respuesta comienza a partir de la segunda línea e incluye la fecha de la conexión e información sobre qué servidor web y sistema operativo utiliza el host. Si estás solicitando un archivo, la cabecera también mostrará información sobre su última fecha de modificación, la longitud del archivo y el tipo de contenido que tienes entre manos. La última línea te indica que la conexión se ha cerrado, ya que la solicitud se ha completado.
La información y los parámetros de las cabeceras pueden variar en función del tipo de solicitud que estés realizando. Sin embargo, la estructura general sigue siendo la misma.

Cuerpo del Mensaje HTTP
El cuerpo del mensaje es la parte más sencilla de una petición HTTP. Contiene los datos que estás enviando o recibiendo, dependiendo del método de solicitud que estés utilizando.
Si solicitas un archivo HTML utilizando el método GET, podrías recibir una respuesta estructurada así

```
HTTP/1.1 200 OK

Date: Mon, 27 Jul 2022 12:28:53 GMT

Server: Apache/2.2.14 (Win32)

Last-Modified: Wed, 22 Jul 2022 19:15:56 GMT

Content-Light: 88

Content-Type: text/html

Connection: Closed

<!DOCTYPE HTML PUBLIC “-//IETF//DTD HTML 2.0//EN”></code.
```

El resto del archivo va aquí
El cuerpo de una petición o respuesta HTTP se separa de la cabecera mediante una sola línea vacía. No hemos incluido un archivo HTML completo en el ejemplo anterior para evitar confusiones.

Cómo Supervisar y Solucionar los Problemas de las Peticiones HTTP

Hay varias formas de supervisar las peticiones HTTP en tu sitio web, como por ejemplo utilizando herramientas de gestión del rendimiento de las aplicaciones (APM). Éstas te permiten controlar las «transacciones» en tus sitios web, como las tareas PHP, los errores HTTP, las peticiones a la base de datos, etc.

Entender qué son las peticiones HTTP y cómo funcionan puede ayudar a solucionar los problemas de un sitio web. Cuando se encuentra con errores HTTP, significa que el servidor no ha podido cumplir la petición que el cliente ha realizado. Si se sabes cuál era esa petición y se entiende el código de error de la respuesta HTTP, se tiene información más que suficiente para solucionar el problema.

Para entender una petición HTTP, se tendrás que saber qué métodos se puede utilizar. Además, se tendrá que saber cómo se estructuran las peticiones y respuestas HTTP, y entender los diferentes códigos de estado HTTP.