UNIVERSIDAD DE LA SIERRA SUR

GENERALIDADES DEL PROTOCOLO HTTP

GAMALILE SILVA LOPEZ
GAMALIEL.SILVA.LOP@GMAIL.COM
FECHA: 14/03/2022

1. Introduccion

En la actualiza la tecnología está creciendo de una manera exponencial al grado que todo se está automatizando, en el caso de los protocolos que se utilizan también se están volviendo cada vez más eficientes, en este caso tenemos el protocolo HTTP el cual es de gran ayuda para poder compartir diversos archivos por medio del internet además de eso nos garantiza un transporte de datos eficiente y trabaja con un modelo de comunicación cliente-servidor en donde el usuario hace una petición, para que después pueda ser respondida. También el protocolo HTTP tiene diferentes tipos de respuestas a las solicitudes del usuario y esto dependerá de la petición que se hagan, pero también existen los mensajes de respuesta, por e igual estos dependen del error que se presenten, los errores más comunes podrían ser el 404 el cual ocurre cuando no se encuentra la petición del usuario.

2. Generalidades del protocolo HTTP

El protocolo HTTP nos permite realizar peticiones de datos y recursos. Este protocolo cuenta con una estructura cliente-servidor en pocas palabras la petición de datos es iniciada por elemento que recibirá los datos que normalmente es una navegador web. Así, una página web completa resulta de la unión de distintos sub-documentos recibidos, como, por ejemplo: un documento que especifique el estilo de maquetación de la página web (CSS), el texto, las imágenes, vídeos, scripts, etc.

Los mensajes que envía el cliente, normalmente un navegador Web, se llaman peticiones, y los mensajes enviados por el servidor se llaman respuestas.

También HTTP pertenece a la capa de aplicación del modelo OSI y se transmite sobre el protocolo TCP, o el protocolo encriptado TLS (en-US), aunque teóricamente podría usarse cualquier otro protocolo fiable. Gracias a que es un protocolo capaz de ampliarse, se usa no solo para transmitir documentos de hipertexto (HTML), si no 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.

3. Métodos de petición

3.1. GET

El método GET se emplea para leer una representación de un *resource*. En caso de respuesta positiva (200 OK), GET devuelve la representación en un formato concreto: HTML, XML, JSON o imágenes, JavaScript, CSS, etc. En caso de respuesta negativa devuelve 404 *(not found)* o 400 *(bad request)*.

3.2. Post

Post se utiliza por las limitaciones de GET. En caso de respuesta positiva devuelve 201 *(created)*. Los POST requests se envían normalmente con formularios.

3.3. PUT

PUT se utiliza para actualizar contenidos, pero también pueden crearlos. Tampoco muestra ninguna información en la URL. En caso de éxito devuelve 201 (*created*, en caso de que la acción haya creado un elemento) o 204 (*no response*, si el servidor no devuelve ningún contenido). A diferencia de POST es idempotente, si se crea o edita un *resource* con PUT y se hace el mismo *request* otra vez, el *resource* todavía está ahí y mantiene el mismo estado que en la primera llamada.

3.4. DELETE

Simplemente elimina un *resource* identificado en la URI. Si se elimina correctamente devuelve 200 junto con un *body* response, o 204 sin *body*. DELETE, al igual que PUT y GET, también es idempotente.

3.5. HEAD

Tiene la misma función que get, pero el servidor no devuelve el contenido en el HTTP *response*. Cuando se envía un *HEAD request*, significa que sólo se está interesado en el código de respuesta y los headers HTTP, no en el propio documento. Con este método el navegador puede comprobar si un documento se ha modificado, por razones de *caching*. Puede comprobar también directamente si el archivo existe.

4. Códigos de respuestas

Los códigos de estado de respuesta HTTP indican si se ha completado satisfactoriamente una solicitud HTTP específica. Las respuestas se agrupan en cinco clases:

1. Respuestas informativas (100–199),
2. Respuestas satisfactorias (200–299),
3. Redirecciones (300–399),
4. Errores de los clientes (400–499),
5. Errores de los servidores (500–599).

4.1. Respuestas informativas

4.1.1. 100 Continue

Esta respuesta provisional indica que todo hasta ahora está bien y que el cliente debe continuar con la solicitud o ignorarla si ya está terminada.

4.1.2. 101 Switching Protocol

Este código se envía en respuesta a un encabezado de solicitud Upgrade (en-US) por el cliente e indica que el servidor acepta el cambio de protocolo propuesto por el agente de usuario.

4.1.3. 102 Processing (WebDAV (en-US))

Este código indica que el servidor ha recibido la solicitud y aún se encuentra procesandola, por lo que no hay respuesta disponible.

4.1.4. 103 Early Hints (en-US)

Este código de estado está pensado principalmente para ser usado con el encabezado Link, permitiendo que el agente de usuario empiece a pre-cargar recursos mientras el servidor prepara una respuesta.

4.2. Respuestas satisfactorias

4.2.1. 200 OK

La solicitud ha tenido éxito.

4.2.2. 201 Created

La solicitud ha tenido éxito y se ha creado un nuevo recurso como resultado de ello. Ésta es típicamente la respuesta enviada después de una petición PUT.

4.2.3. 202 Accepted

La solicitud se ha recibido, pero aún no se ha actuado. Es una petición "sin compromiso", lo que significa que no hay manera en HTTP que permite enviar una respuesta asíncrona que indique el resultado del procesamiento de la solicitud. Está pensado para los casos en que otro proceso o servidor maneja la solicitud, o para el procesamiento por lotes.

4.2.4. 203 Non-Authoritative Information

La petición se ha completado con éxito, pero su contenido no se ha obtenido de la fuente originalmente solicitada, sino que se recoge de una copia local o de un tercero. Excepto esta condición, se debe preferir una respuesta de 200 OK en lugar de esta respuesta.

4.2.5. 204 No Content (en-US)

La petición se ha completado con éxito pero su respuesta no tiene ningún contenido, aunque los encabezados pueden ser útiles. El agente de usuario puede actualizar sus encabezados en caché para este recurso con los nuevos valores.

4.2.6. 205 Reset Content (en-US)

La petición se ha completado con éxito, pero su respuesta no tiene contenidos y además, el agente de usuario tiene que inicializar la página desde la que se realizó la petición, este código es útil por ejemplo para páginas con formularios cuyo contenido debe borrarse después de que el usuario lo envíe.

4.2.7. 206 Partial Content

La petición servirá parcialmente el contenido solicitado. Esta característica es utilizada por herramientas de descarga como wget para continuar la transferencia de descargas anteriormente interrumpidas, o para dividir una descarga y procesar las partes simultáneamente.

4.2.8. 207 Multi-Status (WebDAV (en-US))

Una respuesta Multi-Estado transmite información sobre varios recursos en situaciones en las que varios códigos de estado podrían ser apropiados. El cuerpo de la petición es un mensaje XML.

4.2.9. 208 Multi-Status (WebDAV (en-US))

El listado de elementos DAV ya se notificó previamente, por lo que no se van a volver a listar.

4.2.10. 226 IM Used (HTTP Delta encoding)

El servidor ha cumplido una petición GET para el recurso y la respuesta es una representación del resultado de una o más manipulaciones de instancia aplicadas a la instancia actual.

4.3. Redirecciones

4.3.1. 300 Multiple Choice (en-US)

Esta solicitud tiene más de una posible respuesta. User-Agent o el usuario debe escoger uno de ellos. No hay forma estandarizada de seleccionar una de las respuestas.

4.3.2. 301 Moved Permanently (en-US)

Este código de respuesta significa que la URI del recurso solicitado ha sido cambiado. Probablemente una nueva URI sea devuelta en la respuesta.

4.3.3. 302 Found

Este código de respuesta significa que el recurso de la URI solicitada ha sido cambiado temporalmente. Nuevos cambios en la URI serán agregados en el futuro. Por lo tanto, la misma URI debe ser usada por el cliente en futuras solicitudes.

4.3.4. 303 See Other (en-US)

El servidor envía esta respuesta para dirigir al cliente a un nuevo recurso solicitado a otra dirección usando una petición GET.

4.3.5. 304 Not Modified

Esta es usada para propósitos de çaché". Le indica al cliente que la respuesta no ha sido modificada. Entonces, el cliente puede continuar usando la misma versión almacenada en su caché.

4.4. Errores de cliente

4.4.1. 400 Bad Request

Esta respuesta significa que el servidor no pudo interpretar la solicitud dada una sintaxis inválida.

4.4.2. 401 Unauthorized

Es necesario autenticar para obtener la respuesta solicitada. Esta es similar a 403, pero en este caso, la autenticación es posible.

4.4.3. 402 Payment Required

Este código de respuesta está reservado para futuros usos. El objetivo inicial de crear este código fue para ser utilizado en sistemas digitales de pagos. Sin embargo, no está siendo usado actualmente.

4.4.4. 403 Forbidden

El cliente no posee los permisos necesarios para cierto contenido, por lo que el servidor está rechazando otorgar una respuesta apropiada.

4.4.5. 404 Not Found

El servidor no pudo encontrar el contenido solicitado. Este código de respuesta es uno de los más famosos dada su alta ocurrencia en la web.

5. Cabeceras

Las cabeceras (*headers*) HTTP permiten al cliente y al servidor enviar información adicional junto a una petición o respuesta. Una cabecera de petición esta compuesta por su nombre (no sensible a las mayusculas) seguido de dos puntos ':', y a continuación su valor (sin saltos de línea). Los espacios en blanco a la izquierda del valor son ignorados Se pueden agregar cabeceras propietarias personalizadas usando el prefijo 'X-', pero esta convención se encuentra desfasada desde Julio de 2012, debido a los inconvenientes causados cuando se estandarizaron campos no estandar en el RFC 6648; otras están listadas en un registro IANA, cuyo contenido original fue definido en el RFC 4229, IANA tambien mantiene un registro de propuestas para nuevas cabeceras HTTP

Las Cabeceras pueden ser agrupadas de acuerdo a sus contextos:

- Cabecera general: Cabeceras que se aplican tanto a las peticiones como a las respuestas, pero sin relación con los datos que finalmente se transmiten en el cuerpo.
- Cabecera de consulta: Cabeceras que contienen más información sobre el contenido que va a obtenerse o sobre el cliente.
- Cabecera de respuesta: Cabeceras que contienen más información sobre el contenido, como su origen o el servidor (nombre, versión, etc.).
- Cabecera de entidad: Cabeceras que contienen más información sobre el cuerpo de la entidad, como el tamaño del contenido o su tipo MIME.

Las cabeceras también pueden clasificarse de acuerdo a cómo se comportan frente a ellas los proxies:

- Cabeceras de extremo a extremo Estas cabeceras deben ser enviadas al recipiente final del mensaje; esto es, el servidor (para una petición) o el cliente (para una respuesta). Los proxies intermediarios deben transmitir las cabeceras de extremo-a-extremo sin modificar, y las cachés deben guardarlas tal y como son recibidas.
- Cabeceras de paso Estas cabeceras sólo son significativas para conexiones de un paso, y no deben ser transmitidas por proxies o almacenarse en caché. Éstas cabeceras son: Connection (en-US), Keep-Alive, Proxy-Authenticate (en-US), Proxy-Authorization (en-US), TE (en-US),

Trailer (en-US), Transfer-Encoding and Upgrade (en-US). La cabecera general Connection (en-US) sólo puede usarse para este tipo de cabeceras.

6. Ejemplo de un diálogo HTTP

Para obtener un recurso con el URL http://www.midominio.cu/index.html

1. Se abre una conexión al host www.midominio.cu, puerto 80 que es el puerto por defecto para HTTP.

2. Se envía un mensaje en el estilo siguiente:

GET /index.html HTTP/1.1
Host: www.midominio.cu
User-Agent: nombre-cliente
[Línea en blanco]

La respuesta del servidor está formada por encabezados seguidos del recurso solicitado, en el caso de una página web:

HTTP/1.1 200 OK
Date: Fri, 31 Dec 2003 23:59:59 GMT
Content-Type: text/html
Content-Length: 1221
¡html¿
¡body¿
¡h1¿Página principal de tuHost¡/h1¿
(Contenido)
.....
.....’
¡/body¿
¡/html¿

7. Concluciones

Los diferentes protocolos nos sirven para poder tener una comunicación entre los diferentes servicios o sistemas, pero en este caso se habló sobre HTTP gran protocolo porque hace que las diferentes páginas web se pueden comunicar con el usuario e intercambiar información, documentos, imágenes hasta videos de una forma seguro. Este protocolo ofrece diferentes funciones y dependiendo de las funciones o errores que se provoquen, tiene mensajes los cuales ayudan al usuario a comprender lo que está pasando. De toda esta información se puede tener en cuenta la importancia del protocolo HTTP el cual lo ocupamos cotidianamente.

8. BIBLIOGRAFÍA

Referencias

Códigos de estado de respuesta HTTP - HTTP — MDN. (2022). Recuperado 13 marzo 2022, de https://developer.mozilla.org/es/docs/Web/HTTP/Status

Generalidades del protocolo HTTP - HTTP — MDN. (2022). Recuperado 13 marzo 2022, de https://developer.mozilla.org/es/docs/Web/HTTP/Overview

HTTP headers - HTTP — MDN. (2022). Recuperado 13 marzo 2022, de https://developer.mozilla.org/es/docs/Web/HTTP/Headers

protocolo http. (2022). Recuperado 13 marzo, de https://neo.lcc.uma.es/evirtual/cdd/tutorial/aplicacion/http.html

¿Protocolo HTTP. (2022). Recuperado 13 marzo, de https://victorponz.github.io/Ciberseguridad-PePS/tema1/http/2020/11/05/Protocolo-HTTP.html