Universidad del Chubut

TP1 - Desarrollo de un Servidor Web

Informe Protocolo HTTP

Alumnos:

Macarena Belén Garcia Arcija
Mauro Leiva
Lucas Isaac Soto

Profesor: Ing. Fabio Gabriel Salerno.
Cátedra: Redes y Seguridad Informática.
Carrera: Tecnicatura Universitaria en Desarrollo de Software.
Año: 2023.

Índice

Protocolo HTTP:

En 1989, comenzó el desarrollo de la primera versión del protocolo **HTTP** (Protocolo de transferencia de hipertexto) con el propósito de permitir la transferencia de archivos entre un navegador(cliente) y un servidor web. La versión **0.9** sólo se encargaba de transferir los datos a través de Internet, solo era capaz de manejar archivos **HTML** y manejaba el método **GET**. Las peticiones de esta versión estaban conformadas únicamente por una línea con el siguiente formato:

GET /documento.html

En cuanto a la respuesta, el servidor solo devolvía **HTML** solicitado, ya que no manejaba cabeceras ni códigos de estado. Si se producía un error, se devolvía el **HTML** solicitado con una descripción del problema en su interior.

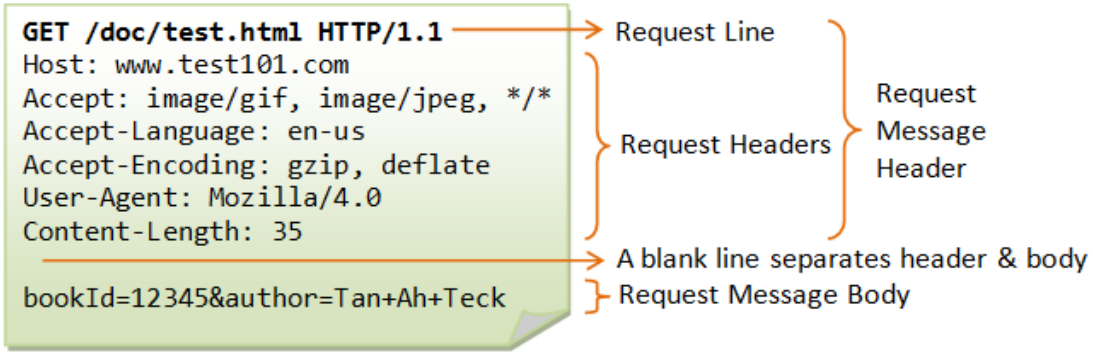
Luego surgió la versión **1.0** de **HTTP**, que permitió transmitir otros tipos de documentos(tipos **MIME**) además de los **HTML**. Esto se logró gracias al uso de cabeceras tanto en la petición como en la respuesta, específicamente la cabecera **Content-Type** es la que permite dicha cualidad. También se implementaron los códigos de estado para informar al cliente del éxito o fracaso de su petición. Además al principio de cada petición se especifica el protocolo utilizado y su versión.

Actualmente se encuentran las versiones **1.1**, la más utilizada, y **2.0**. Esta última es un protocolo binario, lo que permite utilizar en él técnicas de optimización, también deja al servidor almacenar datos en la caché del cliente, previendo su utilización antes de ser requeridos(mecanismo server push), comprime las cabeceras de los mensajes, y al ser un protocolo multiplexado, pueden hacerse peticiones paralelas en una misma conexión sin mantener el orden de los mensajes, ni otras restricciones.

Nota: Para este trabajo utilizaremos la versión **1.0** del protocolo **HTTP**, por lo cual este informe tratará sobre dicha versión.

Petición(request)

El cliente establece una conexión con el servidor y envía una petición a este para acceder a un archivo. Esta petición tiene el siguiente formato:



Request Line: en esta línea se especifica el método que implementa el servidor al devolver la respuesta, le sigue el recurso solicitado por el cliente, y por último se indica el protocolo y la versión a utilizar. Estos tres elementos deben estar separados por un espacio entre ellos. La request line siempre debe estar presente en una petición.
Request Headers: las cabeceras son un grupo de líneas opcionales que aportan información extra sobre el cliente y la solicitud. Cada línea está conformada por un header distinto y su correspondiente valor.
Blanck Line: es una línea en blanco que separa los headers del cuerpo mensaje , es de carácter obligatorio. Se encuentra conformada por un retorno de carro y avance de línea.
Request Message Body: es un conjunto de líneas opcionales que permiten enviar datos por un comando **POST** durante la transmisión de datos al servidor utilizando un formulario, entre otras cosas.

Request Line:

Los tipos de métodos que pueden ser implementados por la versión **1.0** de **HTTP** son **GET**, **HEAD** y **POST**. En la actualidad existen una amplia variedad de métodos HTTP, como DELETE y PUT, pero por implementación de la versión 1.0 de HTTP no son mencionados.

GET: Se usa para obtener información del servidor, puede ser algún archivo HTML, una imagen, un archivo de texto, un XML, etc. Este método solo debe usarse para obtener información del servidor. No debe cambiar el estado del servidor, es decir, no debe hacer ninguna modificación a cualquier archivo que en éste se encuentre.
HEAD: Se usa para obtener la cabecera de respuesta que devuelve el servidor al hacer una petición sobre éste. Similar a GET, ambos no cambian el estado del servidor, aunque HEAD solo devuelve los metadatos. Se puede usar para saber si cierto recurso está en el servidor
POST: Se encarga de crear un nuevo recurso y, por consiguiente, modificar el estado del servidor.

Request Headers:

Este grupo de headers está conformado por **general headers**, **headers** propios del **request**, y/o **entity headers**. Para la versión **1.0** de **HTTP**, la **RFC** recomienda utilizar los headers:

Tipo de Header	Header	Función
GENERAL	**Date**	Fecha y hora en que se originó el mensaje. Ej: **Date: Tue, 15 Nov 1994 08:12:31 GMT**
	Pragma	Se usa para especificar la implementación de directivas que podrían ser aplicadas a cualquier destinatario a lo largo de los **request/response**. Ej: **pragma-directive = "no-cache" | extension-pragma extension-pragma = token ["=" word]**
REQUE	**Authorization**	Identificación del navegador en el servidor.
	From	Permite especificar la dirección de correo electrónico del cliente. Ej: **From: webmaster@w3.org**
	If-Modified-Since	Permite especificar que debe enviarse el recurso si ha sido modificado desde una fecha en particular. Ej: **If-Modified-Since: Sat, 29 Oct 1994 19:43:31 GMT**
	Referer	Dirección **URL** donde se originó la petición. Ej: **Referer:**

S T		http://www.w3.org/hypertext/DataSources/Overview.html
	User-Agent	Cadena con información sobre el cliente. Ej: **User-Agent: CERN-LineMode/2.15 libwww/2.17b3**
E N T I T Y	**Allow**	Lista los métodos soportados por el recurso. No puede utilizarse en un requisito que use el método **POST**. No indica que métodos son implementados por el servidor. Ej: **Allowed: GET, HEAD**
	Content-Encoding	Indica que codificación se ha aplicado al recurso y que mecanismo de decodificación debe aplicarse para obtener el media-type referenciado en el Content-Type. Ej: **Content-Encoding: x-gzip**
	Content-Length	Indica el tamaño del cuerpo de la solicitud. Ej: **Content-Length: 3495**
	Content-Type	Indica el tipo MIME de la solicitud. Ej: **Content-Type: text/html**
	Expires	Indica a partir de que fecha y hora la entidad debe considerarse obsoleta. Esto no implica que el recurso original haya sido modificado o deje de existir luego de dicha fecha. Ej: **Expires: Thu, 01 Dec 1994 16:00:00 GMT**
	Last-Modified	Indica la última fecha y hora en la que el remitente cree que el recurso se modificó. Ej: **Last-Modified: Tue, 15 Nov 1994 12:45:26 GMT**

Tipos MIME

Al utilizarse la cabecera **Content-Type**, se deben utilizar los tipos **MIME(media-type).** En esta versión de **HTTP** manejaremos los siguientes tipos **MIME**:

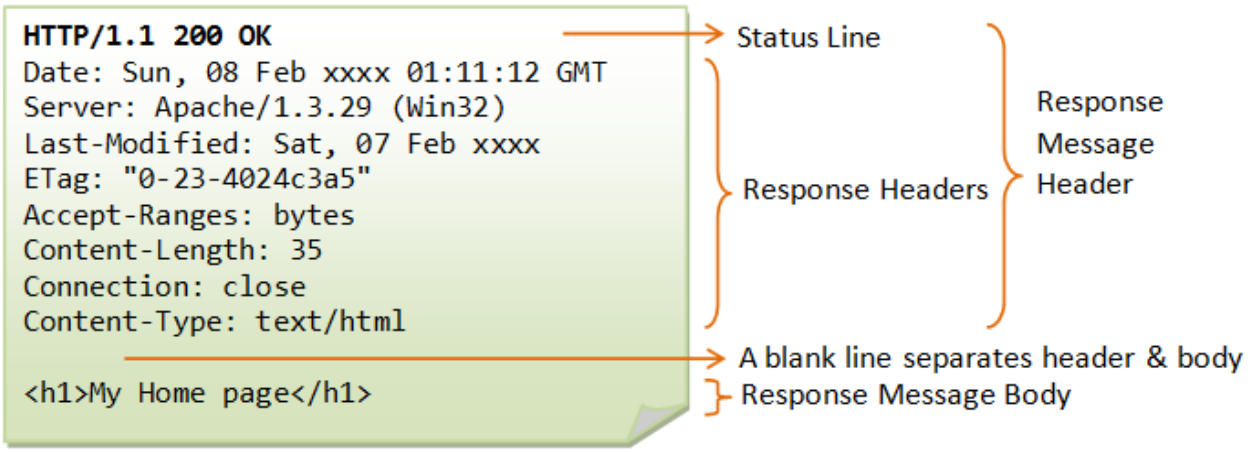
Extensión objeto	Tipo MIME
.htm .html	text/html
.txt	text/plain

.gif	image/gif
.jpg .jpeg	image/jpeg
.pdf	application/pdf
*	aplication/octet-stream

El tipo **MIME** "**aplication/octet-stream**" es utilizado cuando la extensión del recurso solicitado no corresponde con alguno de los otros tipos **MIME**.

Respuesta(response)

Luego de que el cliente envíe la petición(request), el servidor la recibe, la procesa y genera su respuesta(**response**). Esta respuesta debe cumplir con el siguiente formato:



Status Line: esta línea sirve para informarle al cliente si se tuvo éxito al resolver su petición o no. Se encuentra compuesta por el protocolo y la versión utilizada, seguido del código de estado y el mensaje del código. Estos elementos se encuentran separados por un espacio respectivamente. Es de carácter obligatorio.
Response Headers: son las cabezas de respuesta. Estas pueden ser **general headers**, **response headers**, y/o **entity headers**. Son opcionales.
Blank Line: es una línea en blanco que separa los headers del cuerpo mensaje , es de carácter obligatorio. Se encuentra conformada por un retorno de carro y avance de línea.
Response Body: en esta sección se envía el recurso requerido por el cliente. Es opcional.

Status Line - Códigos de estado

La versión **1.0** de **HTTP** es capaz de manejar códigos de estado para indicar el éxito o fracaso de la resolución de una petición **HTTP** por parte del servidor. La familia de códigos
1xx: se trata de códigos de mensaje. **HTTP/1.0** no considera este tipo de códigos como válidos, por lo que no se utilizan.

2xx: los códigos de este grupo son utilizados para indicar éxito en la operación.
3xx: los códigos de estado **3xx** son utilizados en caso de redireccionamiento
4xx: los códigos de estado **4xx** son utilizados para indicar errores por parte del cliente
5xx: los códigos de estado **5xx** son utilizados cuando se produce un error del lado del servidor.

Grupo de Códigos	Código	Función
2XX	**200 OK**	La solicitud se realizo con exito
	201 Created	Respuesta a un método POST. Se cumplio con la peticion y se creo un nuevo recurso
	202 Accepted	La petición fue aceptada para ser procesada, pero el procesamiento no fue completado
	204 No Content	El servidor cumplio con la peticion pero no hay nueva información para devolver
3XX	**301 Moved Permanently**	los datos fueron movidos a una nueva dirección permanentemente
	302 Moved Temporarily	los datos fueron movidos a otra URL temporariamente
	304 Not Modified	Se envía como respuesta si el cliente llevó a cabo un comando GET condicional(con la solicitud relativa a si el documento ha sido modificado desde la última vez) y el documento no ha sido modificado.
4XX	**400 Bad request**	La sintaxis de la petición está formulada de manera errónea o es imposible de responder
	401 Unauthorized	Los parámetros del mensaje aportan las especificaciones de formularios de autorización que se admiten. El cliente debe reformular la solicitud con los datos de autorización correctos
	403 Forbidden	El acceso al recurso simplemente se deniega
	404 Not Found	El servidor no halló nada en la dirección especificada y no hay dirección de redirección.
	500 Internal Server	El servidor encontró una condición

5XX	**Error**	inesperada que le impide seguir con la petición
	501 Not Implemented	El servidor no admite el servicio solicitado
	502 Bad Gateway	El servidor que actúa como una puerta de enlace o proxy ha recibido una respuesta no válida del servidor al que intenta acceder
	503 Service Unavailable	El servidor no puede responder en ese momento debido a que se encuentra congestionado

Response Headers

Este grupo de headers está conformado por **general headers**, **response headers**, y/o **entity headers**. Para la versión **1.0** de **HTTP**, la **RFC 1945** recomienda utilizar los headers:

Tipo de Header	Header	Función
GENERAL	**Date**	Fecha y hora en que se originó el mensaje. Ej: **Date: Tue, 15 Nov 1994 08:12:31 GMT**
	Pragma	Se usa para especificar la implementación de directivas que podrían ser aplicadas a cualquier destinatario a lo largo de los **request/response**. Ej: **pragma-directive = "no-cache" | extension-pragma extension-pragma = token ["=" word]**
RESPONSE	**Location**	Redireccionamiento a una nueva dirección URL asociada con el recurso. Ej: **Location:** http://www.w3.org/hypertext/WWW/NewLocation.html
	Server	Características del servidor que envió la respuesta. Ej: **Server: CERN/3.0 libwww/2.17**
	WWW-Authenticate	Solo se incluye cuando se responde un **401 Unauthorized**.
	Allow	Lista los métodos soportados por el recurso. No puede utilizarse en un requerimiento que

ENTITY		use el método **POST**. No indica qué métodos son implementados por el servidor. Ej: **Allowed: GET, HEAD**
	Content-Encoding	Indica que codificación se ha aplicado al recurso y que mecanismo de decodificación debe aplicarse para obtener el media-type referenciado en el Content-Type. Ej: **Content-Encoding: x-gzip**
	Content-Length	Indica el tamaño del cuerpo de la solicitud. Ej: **Content-Length: 3495**
	Content-Type	Indica el tipo **MIME** de la solicitud. Ej: **Content-Type: text/html**
	Expires	Indica a partir de qué fecha y hora la entidad debe considerarse obsoleta. Esto no implica que el recurso original haya sido modificado o deje de existir luego de dicha fecha. Ej: **Expires: Thu, 01 Dec 1994 16:00:00 GMT**
	Last-Modified	Indica la última fecha y hora en la que el remitente cree que el recurso se modificó. Ej: **Last-Modified: Tue, 15 Nov 1994 12:45:26 GMT**

Bibliografía:

In Introduction to HTTP Basics:
https://www3.ntu.edu.sg/home/ehchua/programming/webprogramming/http_basics.html
Evolución del protocolo HTTP:
https://developer.mozilla.org/es/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP
RFC 1945: Hypertext Transfer Protocol – HTTP/1.0:
https://www.rfc-editor.org/rfc/rfc1945