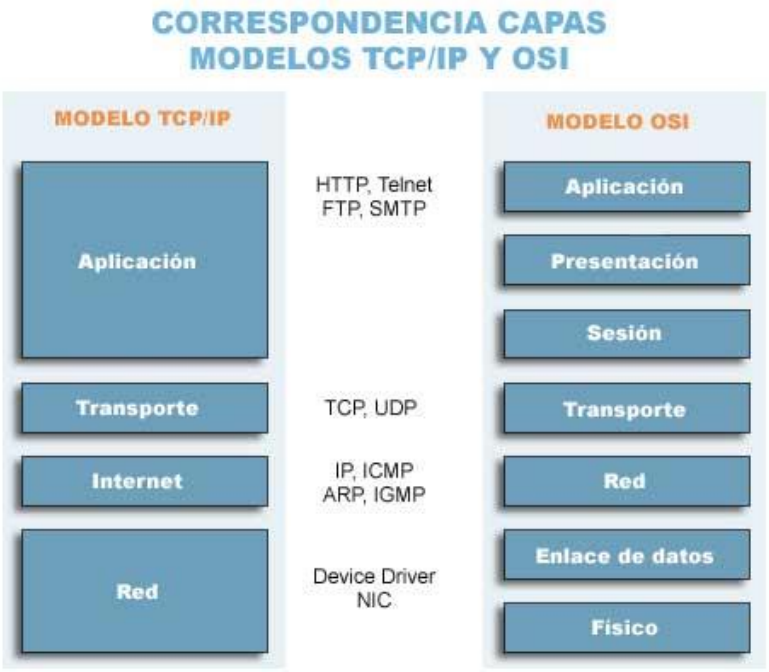
EL PROTOCOLO H.T.T.P.

Tabla de contenido

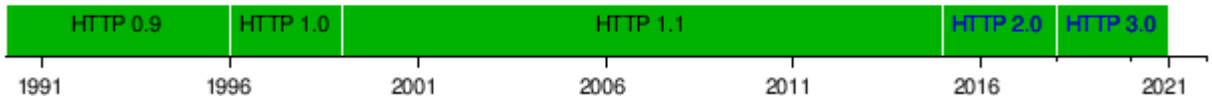
INTRODUCCIÓN



- El Protocolo de Transferencia de HiperTexto (Hypertext Transfer Protocol) es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP.

- Desde el punto de vista de las comunicaciones, está soportado sobre los servicios de conexión TCP/IP, y funciona de la misma forma que el resto de los servicios comunes de los entornos UNIX: un proceso servidor escucha en un puerto de comunicaciones TCP (por defecto, el 80), y espera las solicitudes de conexión de los clientes Web. Una vez que se establece la conexión, el protocolo TCP se encarga de mantener la comunicación y garantizar un intercambio de datos libre de errores.
- HTTP se basa en sencillas operaciones de solicitud/respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud **(request).** El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado **(response**). Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto Web (documento HTML, fichero multimedia o aplicación CGI) es conocido por su URL.
- HTTP es un protocolo **STATELESS (SIN ESTADO) no guarda información sobre las conexiones, es decir el proceso de solicitud-respuesta finaliza en cada ciclo.** (no se actualiza automáticamente)

CRONOLOGÍA DEL PROTOCOLO



FUNCIONAMIENTO DEL PROTOCOLO

Cada vez que un cliente realiza una petición a un servidor, se ejecutan los siguientes pasos:

1. Abrir una conexión TCP: La conexión TCP se utiliza para enviar una solicitud, o varias, y recibir una respuesta. El cliente puede abrir una nueva conexión, reutilizar una conexión existente o abrir varias conexiones TCP a los servidores.
2. Envíe un mensaje HTTP: los mensajes HTTP (antes de HTTP/2) son legibles por humanos. Con HTTP/2, estos mensajes simples se encapsulan en marcos, lo que los hace imposibles de leer directamente, pero el principio sigue siendo el mismo. Por ejemplo:

   GET / HTTP/1.1

   Host: developer.mozilla.org

   Accept-Language: fr

3. Leer la respuesta enviada por el server, algo como:

   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 come the 29769 bytes of the requested web page)

4. Cierre o reúso de la conexión para más solicitudes.

Si la canalización HTTP está activada, se pueden enviar varias solicitudes sin esperar a que se reciba completamente la primera respuesta. La canalización de HTTP ha resultado difícil de implementar en las redes existentes, donde las piezas antiguas de software coexisten con las versiones modernas. La canalización HTTP se reemplazó en HTTP/2 con solicitudes de multiplexación más sólidas dentro de un marco.

VERBOS O MÉTODOS DEL PROTOCOLO

HTTP define **8 métodos** (algunas veces referido como "verbos") que indica la acción que desea que se efectúe sobre el recurso identificado.:

GET Pide una representación del recurso especificado. Por seguridad no debería ser usado por aplicaciones que causen efectos ya que transmite información a través de la URI agregando parámetros a la URL. Ejemplo: GET /images/logo.png HTTP/1.1 obtiene un recurso llamado logo.png Ejemplo con parámetros: /index.php?page=main&lang=es

HEAD Pide una respuesta idéntica a la que correspondería a una petición GET, pero sin el cuerpo de la respuesta. Esto es útil para la recuperación de meta-información escrita en los encabezados de respuesta, sin tener que transportar todo el contenido.

POST Somete los datos a que sean procesados para el recurso identificado. Los datos se incluirán en el cuerpo de la petición. Esto puede resultar en la creación de un nuevo recurso o de las actualizaciones de los recursos existentes o ambas cosas.

PUT Sube, carga o realiza un upload de un recurso especificado (archivo), es el camino más eficiente para subir archivos a un servidor, esto es porque en POST utiliza un mensaje multiparte y el mensaje es decodificado por el servidor. En contraste, el método PUT te permite escribir un archivo en una conexión socket establecida con el servidor. La desventaja del método PUT es que los servidores de hosting compartido no lo tienen habilitado. Ejemplo: PUT /path/filename.html HTTP/1.1

DELETE Borra el recurso especificado.

TRACE Este método solicita al servidor que envíe de vuelta en un mensaje de respuesta, en la sección del cuerpo de entidad, toda la data que reciba del mensaje de solicitud. Se utiliza con fines de comprobación y diagnostico.

OPTIONS Devuelve los métodos HTTP que el servidor soporta para un URL especifico.Esto puede ser utilizado para comprobar la funcionalidad de un servidor web mediante petición en lugar de un recurso especifico.

CONNECT Este método se reserva para uso con proxys. Permitirá que un proxy pueda dinámicamente convertirse en un túnel. Por ejemplo para comunicaciones con SSL.

HEADERS O CABECERAS HTTP

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.

Existen cabeceras de petición y cabeceras de repuesta.

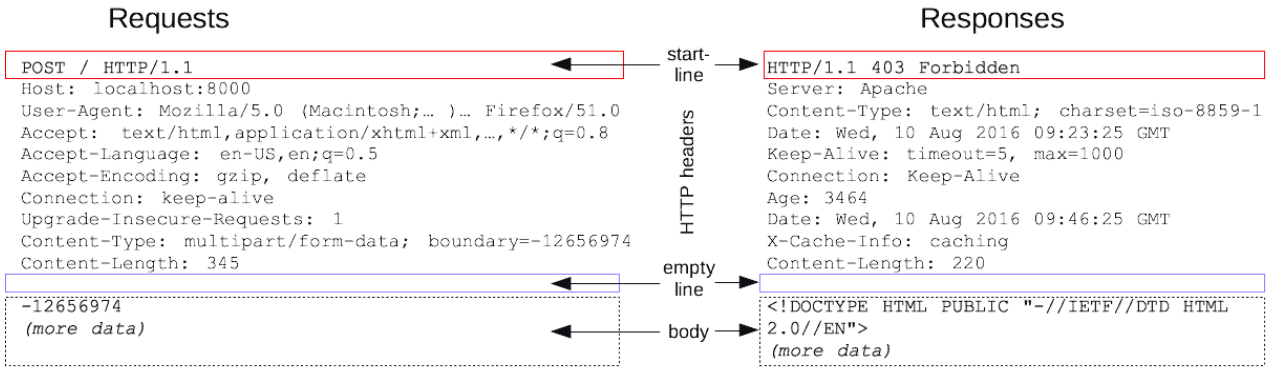
Un HTTP request se compone de:

Método: GET, POST, PUT, etc. Indica que tipo de request es.

Path: la URL que se solicita, donde se encuentra el resource.

Protocolo: contiene HTTP y su versión, actualmente 2.

EJEMPLO DE HEADERS



CODIGOS DE ERROR HTTP

Las 5 clases definidas son las siguientes:

1xx. Informacional. Se recibe la petición y se continua con el proceso. Los códigos en este rango indican respuestas provisionales. Los servidores web no deben enviar mensajes 1xx al cliente HTTP excepto bajo condiciones experimentales.

2xx. Éxito. Esta clase de códigos indican que la petición del cliente fue recibida, entendida, aceptada y procesada exitosamente.

3xx. Redireccionamiento. Para estos códigos el cliente debe realizar acciones adicionales para completar la petición. La acción requerida debe ser portada por el useragent sin la interacción del usuario si y solo si el método usado en la segunda petición es de tipo GET o HEAD. El useragent no debería redireccionar automáticamente más de 5 veces, sino se considera un bucle infinito.

4xx. Error en el Cliente. Estos códigos son arrojados cuando el cliente parece tener un error. Estos tipos de errores son los más comunes que se pueden encontrar. –

5xx. Errores de Servidor. El servidor falla cuando aparentemente se esta ante una petición válida. El Servidor responde con este tipo de errores cuando es incapaz de realizar la petición

PRÁCTICA: RECONOCER CABECERAS Y ERRORES; USO PHP.

MATERIAL ADICIONAL DE CONSULTA

- DOCUMENTOS: https://developer.mozilla.org/es/docs/Web/HTTP
- VIDEO 1: https://www.youtube.com/watch?v=FryJwCCEu7A
- VIDEO 2: https://www.youtube.com/watch?v=mr08bsged-o