Tecnicatura Superior en Telecomunicaciones.
Profesor Ing. Jorge Morales.
Grupo 6
Alumna: Emma Gutiérrez

Actividad Nro. 2

6) ¿Qué es el Protocolo HTTP?,
¿Cuáles son sus características? Ejemplifica.

Este protocolo fue creado en 1999 por el World Wide Web Consortium.

¿Qué es el protocolo http?
El http (del inglés HyperText Transfer Protocol o Protocolo de Transferencia de Hiper Textos) es el protocolo de transmisión de información de la, es decir, el código que se establece para que el solicitante y el que contiene la información solicitada puedan “hablar” un mismo idioma a la hora de transmitir información por la red.

Con el http **se establecen criterios de y informática** (forma y significado) para el establecimiento de la entre los diferentes elementos que constituyen la arquitectura web:, clientes, proxies. Fue creado en 1999 por el World Wide Web Consortium en colaboración con la Internet Engineering Task Force.

Se trata de un protocolo “sin estado”, vale decir, que no lleva registro de visitas ante riores sino que siempre empieza de nuevo. La información relativa a visitas previas se almacena en estos sistemas en las llamadas “cookies”, almacenadas en el sistema .

El http ha pasado por numerosas versiones hasta alcanzar la actual a principios del siglo XXI, llamada HTTP/2. **Sus primeros intentos se produjeron en 1991** y arrojaron versiones parciales en 1996, 1999, 2000 y, finalmente, la vigente en 2015.

¿Para qué sirve el protocolo http?

El http, como se ha dicho, es un lenguaje que **media entre las peticiones del cliente y las respuestas del servidor en la Internet,** para permitir una comunicación fluida y en un mismo “lenguaje”. Este protocolo establece las pautas a seguir, los de petición (llamados “verbos”) y

cuenta con cierta flexibilidad para incorporar nuevas peticiones y funcionalidades, en especial a medida que se avanza en sus versiones.

Considerando que la Internet es poco más que una compleja red de intercambio de información entre computadores a distancia, este tipo de herramientas digitales son clave en establecer las bases para ordenar y facilitar la transmisión de la información.

El funcionamiento del http se basa en **un esquema de petición-respuesta entre el servidor web y el "agente usuario"** (del inglés user agent) o cliente que realiza la solicitud de transmisión de datos. Un cliente puede ser un explorador determinado, cuando intentamos abrir una respuesta estructurada, o los rastreadores web (webcrawlers o arañas web) que las inspeccionan.

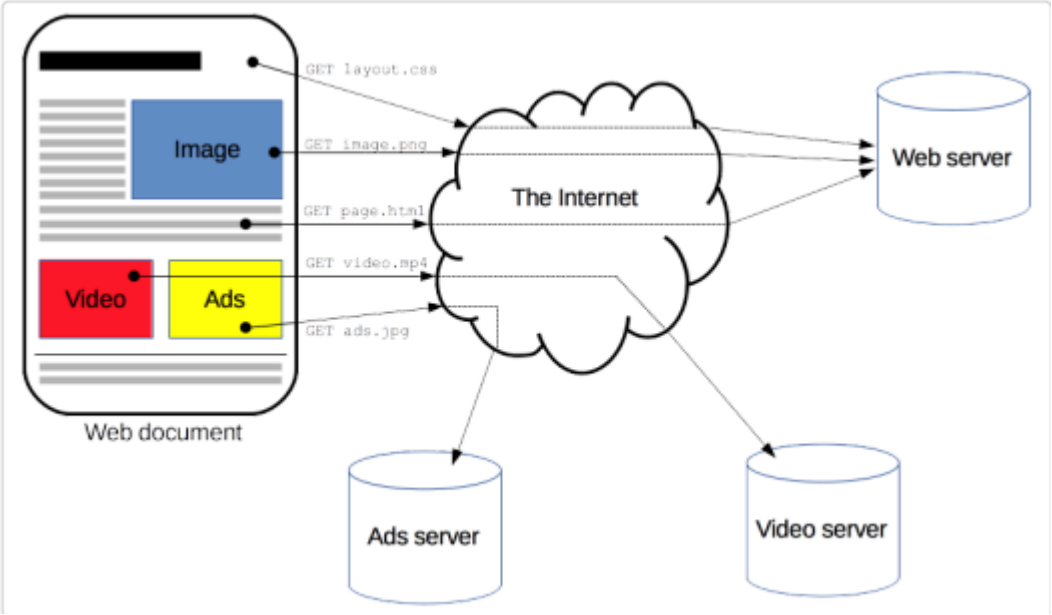
A ellos el servidor brinda una respuesta estructurada de modo puntual y dotada de una serie de metadatos, que establecen las pautas para el inicio, desarrollo y cierre de la transmisión de la información.
Estos son los "métodos de petición", es decir, los comandos que disparan la ejecución de recursos determinados, cuyos archivos residen en el servidor.

Por ejemplo: Al abrir una página web específica, el intercambio informativo entre nuestro explorador web y el servidor donde reside la información establecerá de qué manera debe transmitirse la información, en qué lugar están las imágenes y en qué orden se me mostrarán, etc. Este intercambio de comandos de solicitud y códigos de respuesta da como resultado la representación en mi computador de la misma información contenida originalmente en el servidor, que puede estar a miles de kilómetros de distancia.

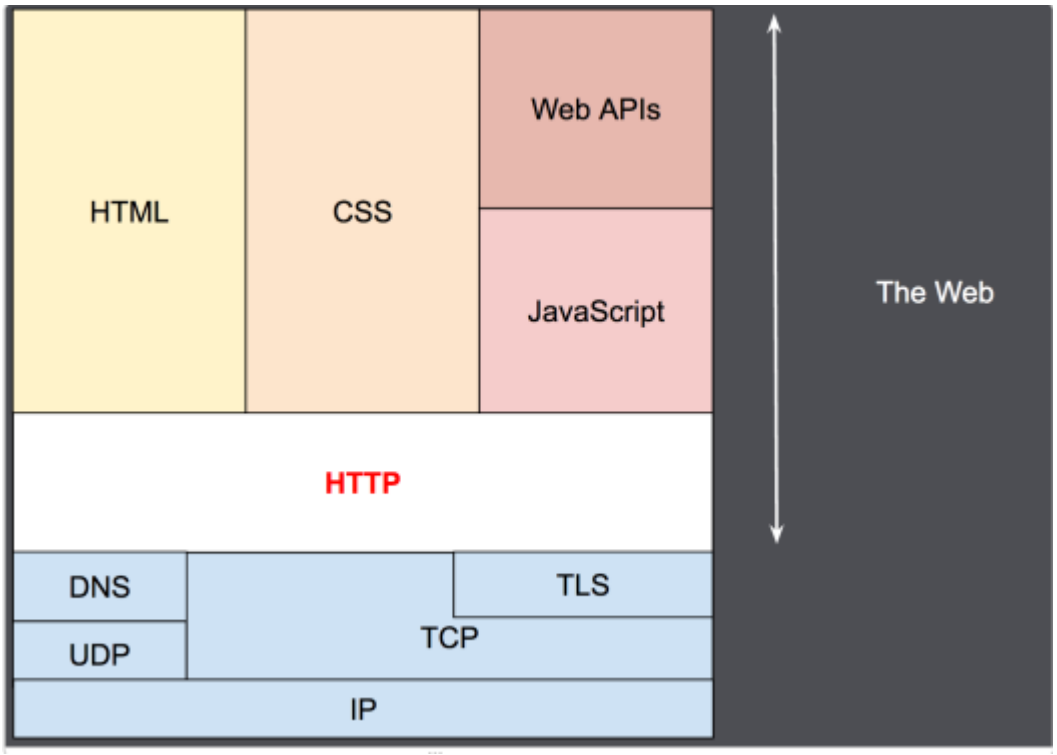
Por **https** se entiende **HyperText Transfer Procotol Secure o Protocolo Seguro de Transferencia de Hipertexto**, que **no es más que la versión segura del http,** es decir, una variante del mismo protocolo que se basa en la creación de un canal cifrado para la transmisión de la información, lo cual lo hace más apropiado para ciertos datos de tipo sensible (como claves y personales).

A diferencia del http, el https **está protegido contra la intervención de terceros** que puedan espiar el intercambio de información o hacerse con datos del mismo, mediante el uso de "capas de red" que permiten sólo al servidor y al cliente cifrar y descifrar la información enviada mediante el intercambio previo de certificados de red, una suerte de convalidación inicial de confianza para establecer la transferencia de información.

Generalidades del protocolo HTTP HTTP, de sus siglas en inglés: "Hypertext Transfer Protocol", es el nombre de un protocolo el cual nos permite realizar una petición de datos y recursos, como pueden ser documentos HTML.
Es la base de cualquier intercambio de datos en la Web, y un protocolo de estructura cliente-servidor, esto quiere decir que una petición de datos es iniciado por el elemento que recibirá los datos (el cliente), normalmente un 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...

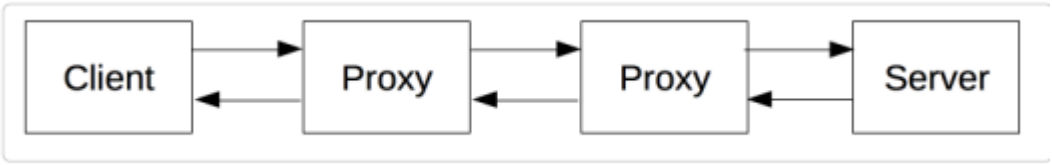


Clientes y servidores se comunican intercambiando mensajes individuales (en contraposición a las comunicaciones que utilizan flujos continuos de datos). Los mensajes que envía el cliente, normalmente un navegador Web, se llaman peticiones, y los mensajes enviados por el servidor se llaman respuestas.



Diseñado a principios de la década de 1990, 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 (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. Arquitectura de los sistemas basados en HTTP

HTTP es un protocolo basado en el principio de cliente-servidor: las peticiones son enviadas por una entidad: el agente del usuario (o un proxy a petición de uno). La mayoría de las veces el agente del usuario (cliente) es un navegador Web, pero podría ser cualquier otro programa, como por ejemplo un programa-robot, que explore la Web, para adquirir datos de su estructura y contenido para uso de un buscador de Internet. Cada petición individual se envía a un servidor, el cual la gestiona y responde. Entre cada petición y respuesta, hay varios intermediarios, normalmente denominados proxies (en-US), los cuales realizan distintas funciones, como: gateways o caches.



El servidor que gestiona su petición: hay otros tipos de dispositivos: como routers, modems ...
Es gracias a la arquitectura en capas de la Web, que estos intermediarios, son transparentes al navegador y al servidor, ya que HTTP se apoya en los protocolos de red y transporte. HTTP es un protocolo de aplicación, y por tanto se apoya sobre los anteriores. Aunque para diagnosticar problemas en redes de comunicación, las capas inferiores son irrelevantes para la definición del protocolo HTTP .

Cliente: el agente del usuario

El agente del usuario, es cualquier herramienta que actué en representación del usuario. Esta función es realizada en la mayor parte de los casos por un navegador Web. Hay excepciones, como el caso de programas específicamente usados por desarrolladores para desarrollar y depurar sus aplicaciones.
El navegador es **siempre** el que inicia una comunicación (petición), y el servidor nunca la comienza (hay algunos mecanismos que permiten esto, pero no son muy habituales).

Para poder mostrar una página Web, el navegador envía una petición de documento HTML al servidor. Entonces procesa este documento, y envía más peticiones para solicitar scripts, hojas de estilo (CSS), y otros datos que necesite (normalmente vídeos y/o imágenes). El navegador, une todos estos documentos y datos, y compone el resultado final: la página Web. Los scripts, los ejecuta también el navegador, y también pueden generar más peticiones de datos en el tiempo, y el navegador, gestionará y actualizará la página Web en consecuencia.

Una página Web, es un documento de hipertexto (HTTP), luego habrá partes del texto en la página que puedan ser enlaces (links) que pueden ser activados (normalmente al hacer click sobre ellos) para hacer una petición de una nueva página Web, permitiendo así dirigir su agente de usuario y navegar por la Web. El navegador, traduce esas direcciones en peticiones de HTTP, e interpretara y procesará las respuestas HTTP, para presentar al usuario la página Web que desea.

El servidor Web

Al otro lado del canal de comunicación, está el servidor, el cual "sirve" los datos que ha pedido el cliente. Un servidor conceptualmente es una unica entidad, aunque puede estar formado por varios elementos, que se reparten la carga de peticiones, (load balancing), u otros programas, que gestionan otros computadores (como cache, bases de datos, servidores de correo electrónico,...), y que generan parte o todo el documento que ha sido pedido.
Un servidor no tiene que ser necesariamente un único equipo físico, aunque si que varios servidores pueden estar funcionando en un único computador. En el estándar HTTP/1.1 y Host, pueden incluso compartir la misma dirección de IP.

Proxies

Entre el cliente y el servidor, además existen distintos dispositivos que gestionan los mensajes HTTP. Dada la arquitectura en capas de la Web, la mayoría de estos dispositivos solamente gestionan estos mensajes en los niveles de protocolo inferiores: capa de transporte, capa de red o capa física, siendo así transparentes para la capa de comunicaciones de aplicación del HTTP, además esto aumenta el rendimiento de la comunicación. Aquellos dispositivos, que sí operan procesando la capa de aplicación son conocidos como proxies. Estos pueden ser transparentes, o no (modificando las peticiones que pasan por ellos), y realizan varias funciones:
caching (la caché puede ser pública o privada, como la caché de un navegador)
filtrado (como un anti-virus, control parental, ...)
balanceo de carga de peticiones (para permitir a varios servidores responder a la carga total de peticiones que reciben)
autentificación (para el control al acceso de recursos y datos)
registro de eventos (para tener un histórico de los eventos que se producen)

Características clave del protocolo HTTP

HTTP es sencillo

Incluso con el incremento de complejidad, que se produjo en el desarrollo de la versión del protocolo HTTP/2, en la que se encapsularon los mensajes, HTTP está pensado y desarrollado para ser leído y fácilmente interpretado por las personas, haciendo de esta manera más fácil la depuración de errores, y reduciendo la curva de aprendizaje para las personan que empieza a trabajar con él.

HTTP es extensible

Presentadas en la versión HTTP/1.0, las cabeceras de HTTP, han hecho que este protocolo sea fácil de ampliar y de experimentar con él. Funcionalidades nuevas pueden desarrollarse, sin más que un cliente y su servidor, comprendan la misma semántica sobre las cabeceras de

HTTP. HTTP es un protocolo 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 crea problemáticas, 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, si 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 trasporte, 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.

En la versión del protocolo HTTP/1.0, habría una conexión TCP por cada petición/respuesta intercambiada, presentando esto dos grandes inconvenientes: abrir y crear una conexión requiere varias rondas de mensajes y por lo tanto resultaba lento. Esto sería más eficiente si se mandaran varios mensajes.

Para atenuar estos inconvenientes, la versión del protocolo HTTP/1.1 presentó el 'pipelining' y las conexiones persistentes: el protocolo TCP que lo transmitía en la capa inferior se podía controlar parcialmente, mediante la cabecera 'Connection'. La versión del protocolo HTTP/2 fue más allá y usa multiplexación de mensajes sobre un única conexión, siendo así una comunicación más eficiente.

Todavía hoy se sigue investigando y desarrollando para conseguir un protocolo de transporte más conveniente para el HTTP. Por ejemplo, Google está experimentado con QUIC , que se apoya en el protocolo UDP y presenta mejoras en la fiabilidad y eficiencia de la comunicación.

Que se puede controlar con HTTP?

La característica del protocolo HTTP de ser ampliable, ha permitido que durante su desarrollo se hayan implementado más funciones de control y funcionalidad sobre la Web: caché o métodos de identificación o autentificación fueron temas que se abordaron pronto en su historia. Al contrario la relajación de la restricción de origen solo se ha abordado en los años de la década de 2010.
Se presenta a continuación una lista con los elementos que se pueden controlar con el protocolo HTTP:

- **Cache** El cómo se almacenan los documentos en la caché, puede ser especificado por HTTP. El servidor puede indicar a los proxies y clientes, que quiere almacenar y durante cuánto tiempo. Aunque el cliente, también puede indicar a los proxies de caché intermedios que ignoren el documento almacenado.
- **Flexibilidad del requisito de origen** Para prevenir invasiones de la privacidad de los usuarios, los navegadores Web, solamente permiten a páginas del mismo origen, compartir la información ó datos. Esto es una complicación para el servidor, asi que mediante cabeceras HTTP, se puede flexibilizar o relajar esta división entre cliente y servidor
- **Autentificación** Hay páginas Web, que pueden estar protegidas, de manera que solo los usuarios autorizados puedan acceder. HTTP provee de servicios básicos de autentificación, por ejemplo mediante el uso de cabeceras como: WWW-Authenticate , o estableciendo una sesión especifica mediante el uso de HTTP cookies.
- **Proxies y tunneling (en-US)** Servidores y/o clientes pueden estar en intranets y esconder así su verdadera dirección IP a otros. Las peticiones HTTP utilizan los proxies para acceder a ellos. Pero no todos los proxies son HTTP proxies. El protocolo SOCKS, por ejemplo, opera a un nivel más bajo. Otros protocolos, como el FTP, pueden ser servidos mediante estos proxies.
- **Sesiones** El uso de HTTP cookies permite relacionar peticiones con el estado del servidor. Esto define las sesiones, a pesar de que por definición el protocolo HTTP es un protocolo sin estado. Esto es muy útil no sólo para aplicaciones de comercio electrónico, sino también para cualquier sitio que permita configuración al usuario.

Flujo de HTTP

Cuando el cliente quiere comunicarse con el servidor, tanto si es directamente con él, o a través de un proxy intermedio, realiza los siguientes pasos:
1. Abre una conexión TCP: la conexión TCP se usará para hacer una petición, o varias, y recibir la respuesta. El cliente pude abrir una conexión nueva, reusar una existente, o abrir varias a la vez hacia el servidor.
2. Hacer una petición HTTP: Los mensajes HTTP (previos a HTTP/2) son legibles en texto plano. A partir de la versión del protocolo HTTP/2, los mensajes se encapsulan en franjas, haciendo que no sean directamente interpretables, aunque el principio de operación es el mismo.

```
GET / HTTP/1.1 Host: developer.mozilla.org Accept-Language: fr
```

3. Leer la respuesta enviada por el servidor:

```
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 comes the 29769 bytes of the requested web
page)
```

4. Cierre o reuso de la conexión para futuras peticiones.
Si está activado el HTTP pipelining, varias peticiones pueden enviarse sin tener que esperar que la primera respuesta haya sido satisfecha. Este procedimiento es difícil de implementar en las redes de computadores actuales, donde se mezclan software antiguos y modernos. Así que el HTTP pipelining ha sido substituido en HTTP/2 por el multiplexado de varias peticiones en una sola trama

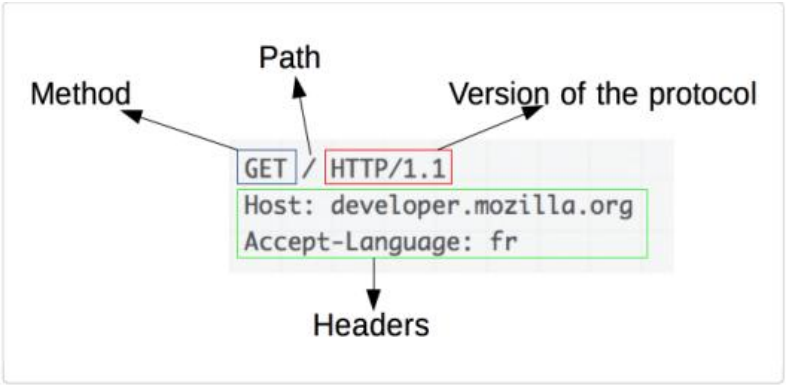
Mensajes HTTP

En las versiones del protocolo HTTP/1.1 y anteriores los mensajes eran de formato texto y eran totalmente comprensibles directamente por una persona. En HTTP/2, los mensajes están estructurados en un nuevo 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 sematica 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. GET / HTTP/1.1

Existen dos tipos de mensajes HTTP: peticiones y respuestas, cada uno sigue su propio formato.

Peticiones

Un ejemplo de petición HTTP:

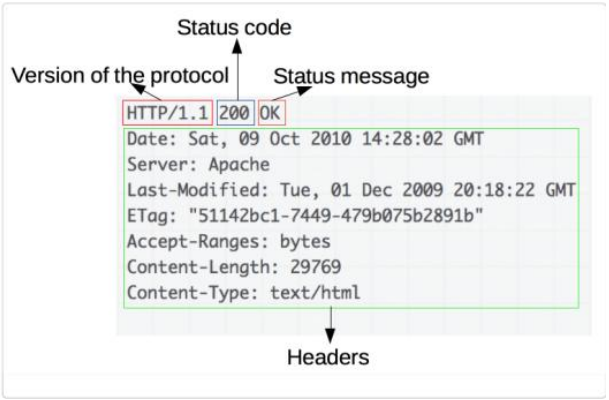


Una petición de HTTP, está formado por los siguientes campos:

- Un método HTTP, normalmente pueden ser un verbo, como: GET , POST o un nombre como: OPTIONS (en-US) o HEAD (en-US), 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 (en-US), 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, que pueden aportar información adicional a los servidores.
- un cuerpo de mensaje, en algún método, como puede ser POST, en el cual envía la información para el servidor

Respuestas:

Un ejemplo de respuesta:



Las respuestas 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, y debido a que.
- 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.

Conclusión

El protocólo HTTP es un protocolo ampliable y fácil de usar. Su estructura cliente-servidor, junto con la capacidad para usar cabeceras, permite a este protocolo evolucionar con las nuevas y futuras aplicaciones en Internet