HTTP (Hypertext Transfer Protocol)

Es un protocolo sin estado de la capa de aplicación para sistemas de información de hipertexto distribuidos y colaborativos.

HTTP se basa en el envío de mensajes sobre el protocolo de transporte TCP:

- El cliente envía un mensaje de petición a un servidor, solicitando realizar una acción sobre un recurso determinado (habitualmente, obtener el recurso).
- El servidor envía un mensaje de respuesta a la petición del cliente (habitualmente, incluyendo el recurso solicitado).

Las versiones más utilizadas actualmente son:

- HTTP/1.1: versión utilizada mayoritariamente desde finales de los 90.
- HTTP/2: versión desplegada en los últimos años. Mejora la eficiencia mediante codificación binaria de mensajes, compresión de cabeceras, multiplexación de múltiples peticiones/respuestas sobre una única conexión TCP, peticiones iniciadas por el servidor, etc.

(Ambas versiones del protocolo son compatibles en términos de semántica y estructura de los mensajes, cambiando principalmente la codificación de los mensajes y el transporte de los mismos mediante conexiones TCP.)

Identificadores de recursos

Los recursos se identifican en HTTP mediante identificadores uniformes de recurso (URI, Uniform Resource Identifier).

Un URI es una secuencia de caracteres compacta que identifica a un recurso abstracto o físico, utilizado en múltiples protocolos y aplicaciones.
Un URI que además proporciona la información necesaria para localizar y acceder al recurso se denomina localizador uniforme de recurso (URL, Uniform Resource Locator).

- Esquema: hace referencia al nombre de un esquema, que define cómo se asignan los identificadores en su ´ámbito. Los esquemas habituales en la Web serán http y https.
- Autoridad: elemento de una autoridad jerárquica de asignación de nombres, típicamente basado en un nombre de dominio de DNS o una dirección de red (IP, IPv6) y, opcionalmente, un número de puerto.
- Ruta: elemento que identifica un recurso en el ´ámbito del esquema y autoridad proporcionados, típicamente organizado jerárquicamente en fragmentos separados por “/”.
- Consulta: datos no jerárquicos que permiten, en combinación en la ruta, identificar el recurso. Es habitual representarlo como uno o más pares nombre/valor.
- Identificador de fragmento: identifica un recurso secundario en el contexto del recurso primario como, por ejemplo, un fragmento concreto de una página Web.

Los dos elementos imprescindibles que deben contener todos los identificadores son scheme y path. En la estructura del URI, los componentes se enumeran uno tras otro por este orden y están separados por caracteres estándar.

scheme :// authority path ? query # fragment

Las dos barras después de los primeros dos puntos solo son necesarias si hay contenido en la parte de authority. Asimismo, authority puede contener información del usuario, que se separa del dominio mediante el signo de @, e incluir una especificación de puerto al final, que se separa a su vez del dominio mediante dos puntos.

Ejemplos:

- https://aulaglobal.uc3m.es/course/view.php?id=91019
- https://www.google.com/search?q=madrid&tbm=isch
- http://example.com/manual#cap3
- http://example.com/manual?lang=es#cap3

Un URI puede contener solo letras de la tabla US-ASCII, dígitos y unos pocos símbolos gráficos (“-”, “.”, “ ” y “~”), además de los caracteres reservados utilizados entre otros para delimitar elementos (“:”, “/”, “?”, “#”, “[”, “]”, “@”, etc.) I Otros caracteres, as´ı como los caracteres reservados cuando se utilicen para representar datos y no como delimitadores, deben ser codificados con codificación de URL.

Se codifica cada carácter no permitido en un URI como una secuencia de octetos, y cada octeto se representa mediante el símbolo “%” seguido del valor del octeto codificado con dos caracteres hexadecimales.

Por ejemplo:

- “path=docs/index.html” se codifica como
  “path=docs%2Findex.html” (el carácter “/” se codifica mediante el octeto 2F en ASCII, UTF-8 y la mayoría de sistemas de codificación de caracteres).

- “q=evaluación” se codifica como “q=evaluaci%C3%B3n” (el carácter “´o” se codifica en UTF-8 mediante la secuencia de octetos C3 y B3).

Métodos de HTTP

Los mensajes de petición especifican un método, que define la acción a realizar sobre el recurso.

Los principales métodos utilizados por las aplicaciones Web son:
- GET: obtener el recurso.
- POST: realizar un procesado (de naturaleza específica para el recurso) basado en los datos que se envían con la petición.
- Otros métodos definidos por el estándar son: HEAD, PUT, DELETE, CONNECT, OPTIONS y TRACE.

Las peticiones GET:

- Se utilizan para obtener el contenido de recursos (páginas
- HTML, imágenes, etc.)
- Son generadas por los navegadores Web, entre otros, cuando se introduce un URL en la barra de direcciones, se pincha en un hipervínculo, se deben pedir recursos adicionales ligados a una página HTML recibida, se envían algunos formularios, etc. I Están sujetas al uso de caches para optimizar el uso de recursos.
- Se consideran seguras, esto es, no deben tener efectos no deseados en el servidor, estado de la aplicación, etc.

Las peticiones POST:

- Se utilizan para realizar acciones (autenticar a un usuario, añadir un producto al carro de la compra, confirmar un pedido en una tienda, subir un nuevo mensaje a una red social, etc.). I Son generadas por los navegadores Web cuando se envían algunos formularios.
- No están sujetas al uso de caches.
- Pueden no ser seguras. Entre otros problemas potenciales, repetir la petición podría tener efectos no deseados (por ejemplo, realizar un mismo pedido dos veces).

Una petición HTTP incluye:

- URL del recurso (sin esquema ni autoridad).
- Método.
- Cabeceras de la petición: datos adicionales acerca de cómo debe ser procesada la petición.
- Cuerpo de la petición (solo con algunos métodos): datos a ser procesados por el servidor.

El cuerpo de la petición:

- No puede ser incluido en peticiones GET.
- Incluye, en un petición POST, los datos a utilizar en el servidor para procesarla.
- Se suele acompañar de las cabeceras Content-Type y Content-Length de la petición.

Una respuesta HTTP incluye:

- Un estado: código numérico indicando el resultado del procesado de la petición.
- Cabeceras de la respuesta: datos adicionales acerca de cómo debe ser procesada la respuesta.
- Cuerpo de la respuesta: representación de la respuesta a la petición, típicamente una página HTML, una imagen, etc.

Cookie

HTTP es un protocolo sin estado, esto es, cada petición es independiente de las demás.

Las cookies permiten mantener estado: consisten en pequeñas cantidades de datos asociados a un nombre que el servidor
genera y envía al cliente en mensajes de respuesta para que este las incluya en sucesivas peticiones.

Estructura de las cookies

Una cookie se representa como una pequeña cadena de texto que contiene:

- Un nombre: un servidor puede establecer varias cookies con distintos nombres.
- Un valor: los datos de la cookie en sí mismos. I Atributos:
- Domain y Path: definen en qué peticiones, por autoridad y ruta, el cliente enviar a la cookie al servidor.
- Expires y Max-Age: definen cuándo la cookie debe dejar de ser utilizada por el cliente. Si no se especifica ninguno, se elimina al cierre del navegador.
- Secure: la cookie solo puede ser enviada por canales seguros (HTTPS típicamente).
- HttpOnly: solo se debe enviar o permitir acceso a la cookie a través de HTTP o HTTPS. Por ejemplo, no debe ser accesible a código JavaScript.

Algunos usos típicos de las cookies son los siguientes:

- Gestión de sesiones: el usuario se autentica al principio para crear una sesión (el servidor envía una cookie con un token de sesión). El servidor identifica peticiones subsiguientes como parte de la misma sesión porque incluyen este mismo token de sesión.
- Almacenamiento de preferencias: se pueden almacenar en cookies las preferencias del usuario para el sitio Web.
- Rastreo de usuarios: los sitios Web pueden utilizar cookies para rastrear el comportamiento de los usuarios (cuando terceros hacen este rastreo, por ejemplo, con fines comerciales, se puede considerar que su uso es abusivo).

HTTPS

HTTP sobre TLS, también llamado HTTPS (Hypertext
Transfer Protocol Secure), define cómo se transporta HTTP sobre un canal seguro TLS (Transport Layer Security).

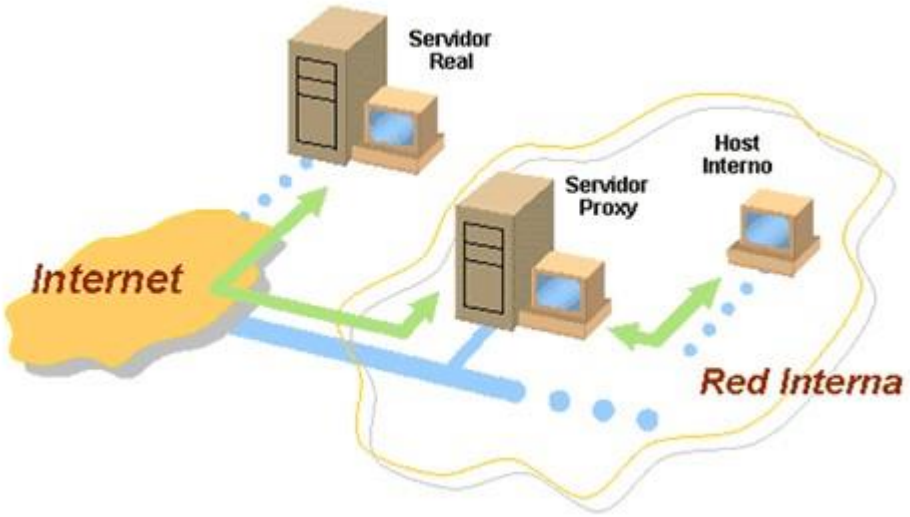
El uso de HTTP sobre TLS proporciona las siguientes propiedades de seguridad:

- Autenticación: se autentica siempre al servidor y, opcionalmente, al cliente.
- Confidencialidad: los datos enviados por el canal seguro una vez este se ha establecido son solo visibles por los dos extremos del mismo.
- Integridad: cualquier modificación de los datos enviados por el canal seguro una vez se ha establecido este será detectada.

Servidores proxy:

Los proxies son máquinas que tienen como misión distribuir el tráfico en la red, de tal manera que las conexiones que se solicitan desde un ordenador local a Internet, pueden dirigirse hacia un servidor o hacia otro según la carga solicitada.

Los proxies instalados en una red también tienen una función "caché". Cuando se solicita una conexión a la red (URL), la página que se ha "bajado" hasta el ordenador a través del navegador, se guarda en la memoria de ese servidor proxy durante un periodo de tiempo y así, cuando se vuelve a solicitar esa dirección desde otro ordenador de la red, el servidor proxy le ofrece la página que tiene guardada en la memoria, consiguiendo una mayor velocidad de respuesta y un ahorro en el tráfico de la red.



Un servidor proxy es un equipo que actúa de intermediario entre un explorador web (como Internet Explorer) e Internet. Los servidores proxy ayudan a mejorar el rendimiento en Internet ya que almacenan una copia de las páginas web más utilizadas. Cuando un explorador solicita una página web almacenada en la colección (su caché) del servidor proxy, el servidor proxy la proporciona, lo que resulta más rápido que consultar la Web. Los servidores proxy también ayudan a mejorar la seguridad, ya que filtran algunos contenidos web y software malintencionado.

Los servidores proxy se utilizan a menudo en redes de organizaciones y compañías. Normalmente, las personas que se conectan a Internet desde casa no usan un servidor proxy.

Proxies directos

Recibe la petición iniciada por un cliente web y se la traslada al servidor web. La solicitud del cliente es hacia el servidor web no hacia el proxy, este solo hace de intermediario o representante del cliente. Usados habitualmente para optimizar y controlar el acceso a redes externas.

Proxies inversos

Igualmente en este caso reciben la petición de un cliente web y la reenvían hacia uno o varios servidores web. En este caso, la solicitud del cliente es hacia el proxy. Usados habitualmente para ofrecer acceso a servidores web que están detrás de un cortafuegos y no son accesibles directamente, para balancear la carga entre varios servidores web, para aumentar los accesos ofreciendo almacenamiento cache etc.