Tema 3-Protocolos del nivel de aplicación

La Web es una aplicación o servicio de Internet que combina cuatro ideas principales:

- **Hipertexto**: formato de información que permite moverse de una parte a otra de un documento o entre documentos mediante conexiones internas entre estos documentos, también conocidos como hiperenlaces o enlaces.
- **Identificadores de recursos**: identificadores únicos que permiten localizar un recurso en la red, como una URL (Uniform Resource Locator) o una URI (Uniform Resource Identifier).
- **Modelo cliente-servidor**: modelo de comunicación en el que un cliente (navegador web) envía solicitudes al servidor web para obtener recursos, como páginas web, imágenes o videos. El servidor web almacena estos objetos web y los hace accesibles a través de una URL.
- **Lenguajes de marcas**: lenguaje que utiliza caracteres o códigos embebidos en texto para indicar la estructura, semántica o recomendaciones para la presentación de una página web. La Web utiliza HTML (HyperText Markup Language), que permite la creación de páginas web que pueden contener texto, imágenes, videos, formularios y otros elementos multimedia.

Los componentes de la Web son:

- **Página web**: se compone de un archivo HTML base y otros objetos como imágenes.
- **Navegador**: el agente de usuario utilizado para acceder a la Web.
- **Servidor web**: almacena los objetos web y los hace accesibles a través de una URL.
- **Protocolo HTTP (Hypertext Transfer Protocol)**: permite la comunicación entre el navegador y el servidor web para la transmisión de los recursos.

URI y URL

Un URI (Uniform Resource Identifier) es un identificador que permite acceder a un recurso en la Web, como una página web, una imagen, un archivo de audio o cualquier otro tipo de recurso.

Por ejemplo:
http://www.udc.es/lista.html?urlmenu=/servizos/#Final

- **Esquema**: especifica el protocolo utilizado para acceder al recurso. Por ejemplo, http, ftp, https, mailto, etc. El esquema se coloca antes del símbolo "://". En este ejemplo http.

- **Parte jerárquica**: se divide en dos partes: autoridad y ruta. La autoridad identifica el nombre (o dirección IP) del servidor que aloja el recurso y puede incluir el número de puerto o información de control de acceso. La ruta es la dirección de la ubicación del recurso en el servidor, similar a los directorios en una estructura de archivos. En este ejemplo www.udc.es/lista.html
- **Consulta**: información adicional que se puede incluir en la URL, como variables y sus valores, para enviar los campos de un formulario o para realizar una consulta más detallada al servidor. En este ejemplo urlmenu=/servizos/.
- **Fragmento**: es opcional y se utiliza para identificar una subsección específica del recurso, como una sección específica de una página web. En este ejemplo `#Final`.

Se llama URL (Uniform Resource Locator) es una URI sin el fragmento.

HTTP

Una RFC (Request for Comments) es un tipo de documento técnico utilizado en ingeniería de redes y sistemas de computación. Las RFC se utilizan para describir estándares, protocolos, procedimientos y tecnologías relacionadas con Internet, redes de computadoras y sistemas de comunicaciones.

HTTP (HyperText Transfer Protocol) es el protocolo estándar utilizado para la transferencia de datos en la Web. Fue especificado en varias RFCs, incluyendo la RFC 1945 (HTTP/1.0), RFC 2616 (HTTP/1.1) y RFC 7540 (HTTP/2).

El protocolo HTTP se basa en un modelo cliente-servidor, en el cual el cliente (por lo general un navegador web) realiza una petición al servidor para obtener información y el servidor (donde se aloja la información solicitada) responde con una respuesta HTTP. Esta respuesta puede incluir la información solicitada o un mensaje de error en caso de que la información no esté disponible o no se pueda acceder a ella.

HTTP utiliza el protocolo TCP (Transmission Control Protocol) como base para la transmisión de datos. TCP es un protocolo orientado a conexión, lo que significa que antes de enviar datos, se establece una conexión entre el cliente y el servidor. Esto garantiza que los datos lleguen al otro extremo sin modificaciones y que se reciba una confirmación de que los datos fueron recibidos correctamente.

Es un protocolo sin estado, lo que significa que el servidor no almacena información sobre las peticiones anteriores del cliente. Cada petición es independiente y única, por lo que el servidor no tiene información sobre la historia de las solicitudes realizadas por el cliente.

HTTP tiene varias versiones, siendo las más conocidas HTTP/1.0, HTTP/1.1 y HTTP/2. Cada versión ha introducido mejoras y nuevas funcionalidades, siendo la versión HTTP/2

la más reciente. Sin embargo, a pesar de las mejoras, HTTP sigue siendo compatible con versiones anteriores, lo que permite que los clientes y servidores puedan seguir comunicándose sin problemas, incluso si utilizan versiones distintas del protocolo. Sin embargo, HTTP/2 no es compatible en la transmisión.

HTTP/1.0

HTTP/1.0 utiliza conexiones no persistentes, lo que significa que después de la transmisión de un objeto, la conexión TCP se cierra. En resumen, cuando un cliente HTTP (por ejemplo, un navegador web) desea acceder a una URL (por ejemplo http://www.fic.udc.es/presentacion), inicia una conexión TCP con el servidor web (www.fic.udc.es) en el puerto 80. Luego envía un mensaje de petición solicitando el objeto /presentacion (el archivo por defecto que se encuentra en /presentacion). El servidor web recibe la petición, busca el objeto, lo encapsula en un mensaje HTTP de respuesta y lo envía. Después de eso, el servidor web finaliza la conexión TCP. El cliente HTTP recibe la respuesta y también finaliza la conexión TCP.

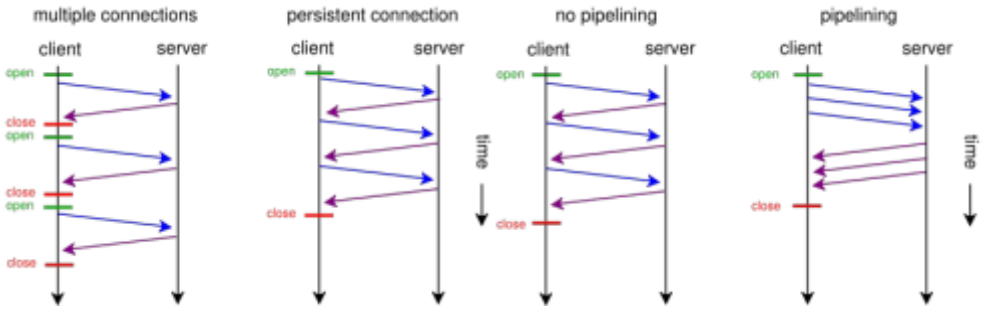
Las conexiones podrían ser en paralelo, es decir, algunos navegadores pueden abrir varias conexiones simultáneas al mismo servidor para recuperar varios objetos (por ejemplo, imágenes, hojas de estilo CSS, scripts JavaScript) al mismo tiempo. Una de las desventajas de las conexiones no persistentes es que se necesita una conexión nueva para cada objeto solicitado, lo que puede aumentar el tiempo de espera y retrasar la carga de la página web. Además, cada vez que se establece una conexión, hay un retardo de dos veces el RTT (Round-Trip Time): el tiempo que tarda en establecerse la conexión y el tiempo que tarda en recibir el objeto solicitado. Esto puede aumentar aún más el tiempo de carga de la página.

HTTP/1.1

HTTP/1.1 utiliza conexiones persistentes, lo que significa que la conexión TCP se mantiene abierta después de la transmisión de un objeto, lo que permite que el servidor web pueda esperar nuevas peticiones y respuestas. Sin embargo, si no hay actividad en la conexión durante un cierto tiempo, el servidor web cerrará la conexión TCP. Esto se hace para evitar que los recursos del servidor se agoten debido a la gran cantidad de conexiones abiertas simultáneamente.

En las conexiones persistentes, el cliente HTTP puede enviar varias solicitudes en serie sin tener que esperar a recibir una respuesta antes de enviar la siguiente solicitud. Esto se puede hacer sin pipeline (también conocido como no pipelining), donde el cliente envía una solicitud y espera a recibir la respuesta antes de enviar la siguiente solicitud. También puede hacerse con pipeline, donde el cliente envía solicitudes tan pronto como encuentra una referencia a un objeto, sin esperar a recibir la respuesta previa.

Con pipeline, el cliente puede enviar varias solicitudes en paralelo y el servidor puede responder en el orden en que las recibe. Esto puede mejorar la velocidad de carga de la página web al permitir que se recuperen varios objetos al mismo tiempo. Sin embargo, es importante tener en cuenta que no todos los servidores web o navegadores admiten el pipeline, y en algunos casos, puede haber problemas de compatibilidad.



HTTP/2

HTTP/2 es la versión más reciente del protocolo HTTP. Fue estandarizada en mayo de 2015 en el RFC 7540 y se basa en el protocolo SPDY de Google. Aunque mantiene los mismos métodos, códigos de estado y demás elementos del protocolo HTTP, cambia la forma en que se envían los datos. Las mejoras que trae HTTP/2 son:

- **Multiplexación completa de la conexión TCP**: varios objetos web pueden ser descargados de manera asíncrona a través de una sola conexión TCP, lo que resuelve el problema del head-of-line (HOL) que existía en HTTP/1.1. En HTTP/1.1, la descarga de un objeto podía bloquear la descarga de otros objetos, lo que ralentizaba la carga de la página.
- **Protocolo está en formato binario**: permite una mejor compresión y más eficiente transmisión de los datos. Esto ayuda a reducir el tiempo de carga de la página y a mejorar el rendimiento general.
- **Compresión de cabeceras (HPACK)**: reduce el tamaño de las cabeceras que se envían en cada solicitud y respuesta, lo que ayuda a reducir el ancho de banda necesario para transmitir los datos.
- **Server push**: permite al servidor enviar objetos no solicitados por el cliente para que se almacenen en caché. Esto ayuda a reducir el tiempo de carga de la página, ya que los objetos ya están disponibles en caché cuando el navegador los necesita.

Mensajes HTTP

Ejemplo de petición:

```
GET /index.html HTTP/1.1 -> línea de petición
Host: www.fic.udc.es -> líneas de cabecera
```

```
User-agent: Mozilla/4.0 -> líneas de cabecera
\b -> línea en blanco
<body> -> cuerpo de entidad
```

- **Línea de petición (obligatoria)** : está formada por el método, el URL del objeto al que se hace referencia y la versión HTTP. El método puede ser:
  - **GET**: el navegador solicita un objeto. El servidor responde con un mensaje HTTP y con el objeto.
  - **HEAD**: el servidor responde con un mensaje HTTP, pero sin incluir el objeto solicitado (sólo la cabecera).
  - **POST**: se utiliza para enviar datos a un servidor web con el fin de procesarlos y/o almacenarlos. Se suele utilizar cuando el usuario rellena un formulario.
  - **PUT**: se utiliza para actualizar o crear un recurso en un servidor web.
  - **DELETE**: se utiliza para borrar un recurso en un servidor web.

Ejemplo de respuesta:

```
HTTP/1.1 200 OK -> línea de estado
Date: Sat, 1 Jan 2000 12:00:15 GMT -> líneas de cabecera
Server: Apache/1.3.0 (Unix) -> líneas de cabecera
Last-Modified: Fri, 24 Dic 1999 13:03:32 GMT -> líneas de cabecera
Content-Length: 6821 -> líneas de cabecera
Content-Type: text/html -> líneas de cabecera
\b -> línea en blanco
<body> -> cuerpo de entidad
```

- **Línea de estado**: está formada por la versión, el código de estado y la frase. El código de estado puede ser:
  - **Informativo**: 100 Continue
  - **Éxito**: 200 OK
  - **Redirecciones**: 300 Moved Permanently
  - **Error del cliente**: 404 Not Found
  - **Error del servidor**: 500 Internal Server Error
- **Date**: fecha y hora en la que se envió la respuesta.
- **Server**: servidor que ha lanzado la petición.
- **Last-Modified**: fecha y hora en la que el objeto fue modificado por última vez.
- **Content-Length**: número de bytes del objeto enviado.
- **Content-Type**: tipo de objeto indicado mediante un MIME type (Multipurpose Internet Mail Extensions type).

Los MIME type pueden ser:

- **Discretos**: se utilizan para enviar un único documento de un único tipo.
  - application/pdf, application/zip, application/octet-strem
  - audio/mpeg
  - image/png, image/jpeg, image/csv
  - video/mp4
- **Multipart**: permite encapsular múltiples archivos de posiblemente distintos tipos en una única transacción.

Cookies HTTP

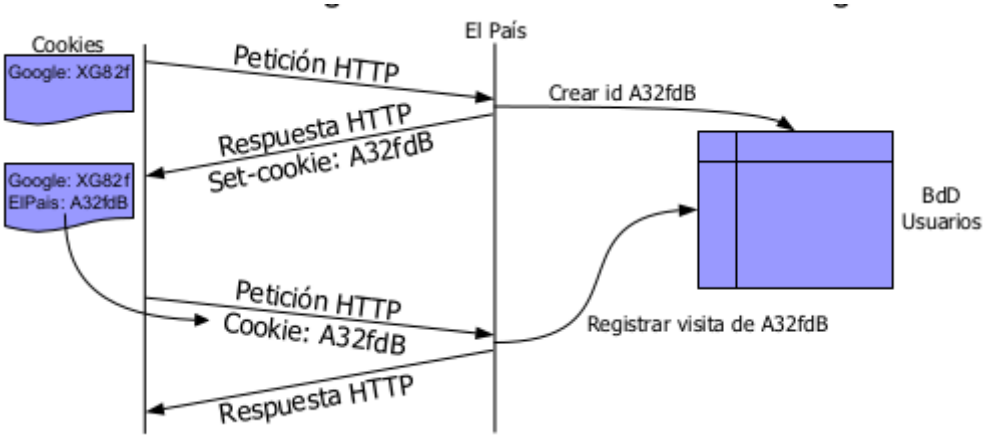
HTTP es un protocolo sin estado, lo que significa que cada solicitud HTTP que se envía a un servidor web se procesa de manera independiente, sin conexión o conocimiento del estado anterior. Esto significa que, por defecto, HTTP no tiene memoria y no puede recordar la información sobre sesiones anteriores.

Sin embargo, para muchas aplicaciones web, es importante poder recordar información sobre sesiones anteriores, como la información de inicio de sesión del usuario, las preferencias del usuario, el contenido del carrito de compras, entre otros. Es aquí donde entran en juego las cookies de HTTP.

Las cookies son un mecanismo que permite a un servidor web guardar información en el navegador del usuario, que luego se puede utilizar para identificar y rastrear al usuario en sesiones futuras. Las cookies se definen en la especificación RFC 2965 del protocolo HTTP.

Cuando un usuario visita un sitio web, el servidor web envía una cookie al navegador del usuario en la respuesta HTTP. La cookie es un pequeño archivo de texto que contiene información, como un identificador de sesión o preferencias del usuario, y se almacena en el disco duro del usuario. Cuando el usuario vuelve a visitar el sitio web en una sesión posterior, el navegador envía la cookie al servidor web en cada solicitud HTTP, lo que permite al servidor web identificar al usuario y recordar información sobre la sesión anterior.

Las cookies de HTTP tienen una serie de propiedades que pueden ser configuradas por el servidor web, como la fecha de expiración, el alcance (por ejemplo, si se puede compartir entre subdominios) y la seguridad (por ejemplo, si se deben transmitir solo a través de una conexión segura HTTPS).



GET condicional

Cuando un cliente realiza una solicitud HTTP GET a un servidor web para obtener un recurso, como una imagen o una página web, el servidor web puede incluir información adicional en la respuesta HTTP para ayudar a la caché del cliente a determinar si su copia del recurso en caché está actualizada.

El problema es que la copia de un objeto en caché puede ser obsoleta si el servidor web ha actualizado el recurso desde que el cliente lo obtuvo por última vez. Esto puede resultar en que el cliente vea una versión obsoleta del recurso, lo que puede provocar problemas de compatibilidad o seguridad.

La solución a este problema es usar una solicitud GET condicional, que incluye un encabezado HTTP If-Modified-Since que indica la fecha en que el cliente obtuvo por última vez el recurso. Si el servidor web detecta que el recurso ha sido modificado desde la última vez que el cliente lo obtuvo, enviará una nueva versión del recurso en la respuesta HTTP. De lo contrario, el servidor web puede responder con un código de estado HTTP 304 Not Modified, lo que indica que la copia en caché del cliente todavía es válida.

Por ejemplo:

```
GET /images/udc.gif HTTP/1.1 //petición
User-agent: Mozilla/4.0
```

```
HTTP/1.1 200 OK //respuesta
Date: Sat, 1 Jan 2000 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Fri, 24 Dic 1999 13:03:32 GMT
Content-Type: image/gif

```

```
\b
(datos)...
```

```
GET /images/udc.gif HTTP/1.1 //petición 2
User-agent: Mozilla/4.0
If-modified-since: Fri, 24 Dic 1999 13:03:32 GMT
```

```
HTTP/1.1 304 Not Modified //resppuesta 2
Date: Wed, 5 Jan 2000 20:30:43 GMT
Server: Apache/1.3.0 (Unix)
```