

Protocolo HTTP

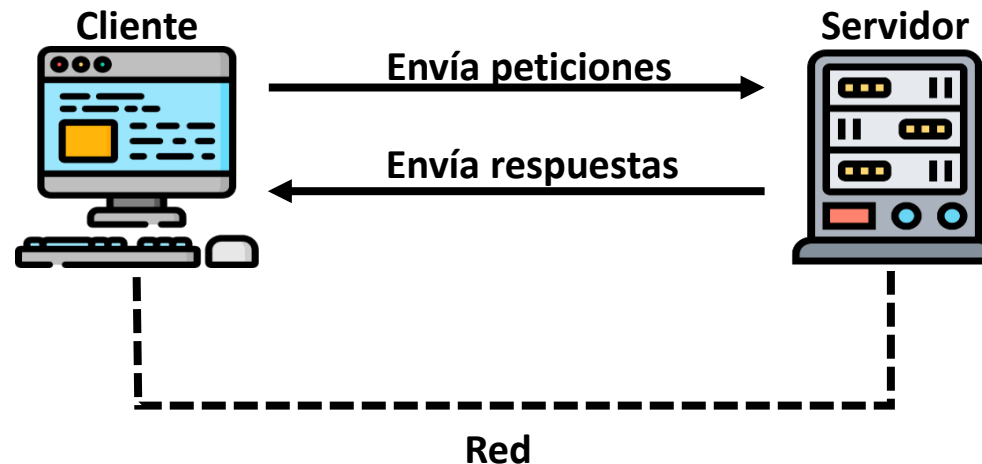
Miguel Sánchez Santillán – sanchezsmiguel@uniovi.es

Sistemas Distribuidos e Internet

Grado en Ingeniería del Software – 2023/2024

Protocolo HTTP - Definición

- **HTTP (HyperText Transfer Protocol)** es el **protocolo de aplicación** utilizado para el intercambio de información en la Web mediante el **paso de mensajes**.
- Está basado en el **Modelo Cliente/Servidor**. Un cliente (**user-agent**) envía una petición (**Request**) a un servidor y éste le responde (**Response**).



HTTP – Mensajes de Petición I

- Los mensajes en HTTP **se componen de texto plano** (ASCII).
- Se envían de forma abierta (HTTP/1.1) o en tramas binarias (HTTP/2).
- Los mensajes pueden ser de **tipo petición** (request) o de **tipo respuesta** (response).
 - La estructura del mensaje varía según el tipo.

HTTP – Mensajes de Petición II

- Estructura de un mensaje de petición:



HTTP – Métodos de Petición I

- Las peticiones comienzan con un método de petición:
 - **GET**: Para recuperar un recurso (READ).
 - **POST**: Para crear un nuevo recurso (CREATE).
 - **PUT**: Para modificar completamente un recurso* (UPDATE).
 - **PATCH**: Envía exclusivamente los datos a modificar del recurso.
 - **DELETE**: Elimina un recurso (DELETE).
 - **Otros**: HEAD, OPTIONS, TRACE...
- HTTP es extensible y permite la creación de nuevos métodos.

HTTP – Métodos de Petición II

- Desencadenamos una petición GET en el navegador (user-agent) si*:
 - Escribimos una URL en la barra de direcciones.
 - Hacemos clic en un enlace de un sitio web.
 - Enviamos un formulario de tipo GET:

```
<form action="GreetingServlet" method="get">  
  <label for="nombre">Nombre:</label>  
  <input type="text" name="nombre" id="nombre"/>  
  <input type="submit" value="Enviar"/>  
</form>
```

- **action** → Ruta a la que se enviará el formulario.
- **method** → Cómo enviamos la información (get o post).
- Pares de **name=value** → Datos que enviamos (nombre=lo_que_teclee_el_usuario).
 - Los pares viajan concatenados en la URL: ?param1=value1¶m2=value2...

HTTP – Métodos de Petición III

- Desencadenamos una petición POST en el navegador sí*:
 - Enviamos un formulario de tipo POST:

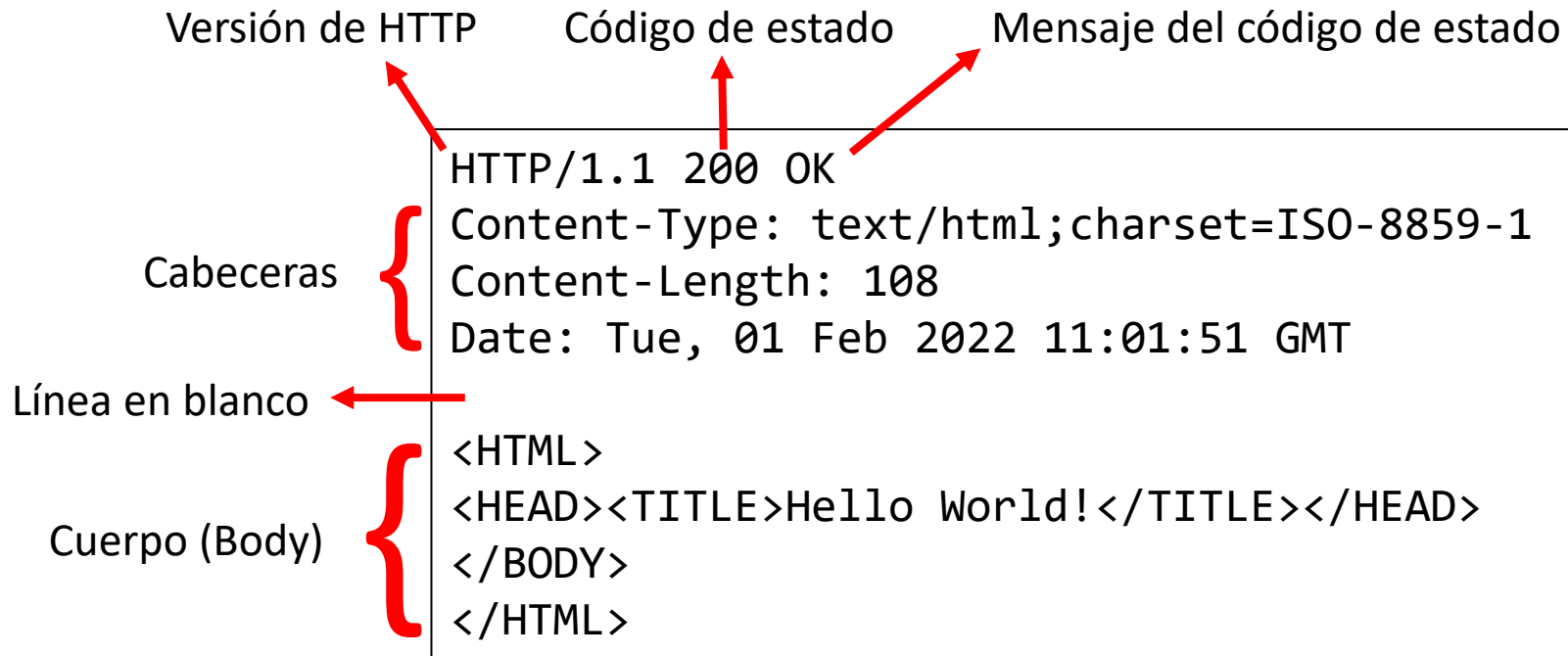
```
<form action="GreetingServlet" method="post">  
  <label for="nombre">Nombre:</label>  
  <input type="text" name="nombre" id="nombre"/>  
  <input type="submit" value="Enviar"/>  
</form>
```

- Los pares de name=value viajan en el cuerpo de la petición.

¿Qué ocurre con otros métodos como PUT o DELETE?

HTTP – Mensajes de Respuesta

- Estructura de un mensaje de respuesta:



- **Nota:** La respuesta puede no tener cuerpo → Tampoco tendrá línea en blanco.

HTTP – Códigos de Respuesta

- Los códigos de respuesta se agrupan en cinco categorías:
 - **1XX: Respuestas informativas** → 101 Switching Protocols, ...
 - **2XX: Respuestas exitosas** → 200 OK, 204 No content, ...
 - **3XX: Redirecciones** → 301 Moved Permanently, ...
 - **4XX: Errores del cliente** → 404 Not Found, 403 Forbidden, ...
 - **5XX: Errores del servidor** → 500 Internal Server Error, ...

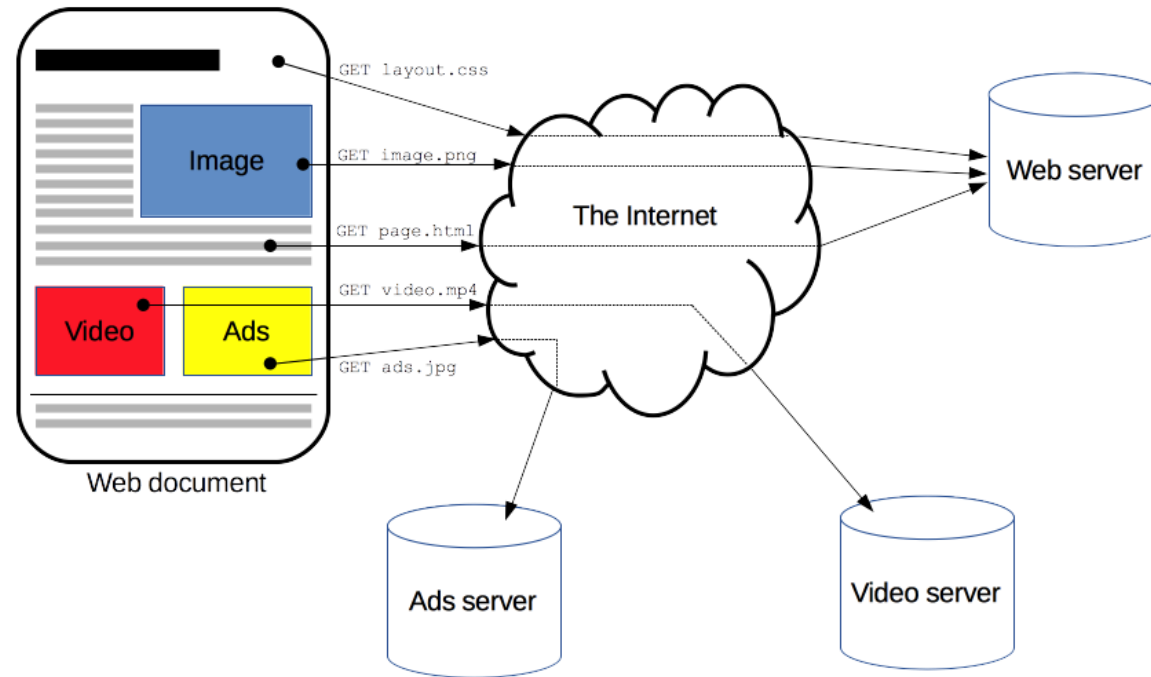
HTTP – Cabeceras en los mensajes

- Existen diferentes tipos de cabeceras, algunas destacables:
 - **Content-Type**: Indica al user-agent el **tipo MIME del documento enviado** (text/html, image/png, application/pdf,...).
 - **Content-Length**: Longitud en bytes del cuerpo.
 - **Cache-Control**: Cómo gestionar la caché de un determinado recurso.
 - **Set-Cookie**: Se envía en respuestas con el fin de crear una cookie.
 - **Cookie**: Se envía en peticiones para transmitir los datos almacenados en una cookie.

Más cabeceras en <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>

HTTP- Solicitud de una página web

- Cuando el navegador solicita una página web, **recibe la página Y:**
 - Desencadena **una petición para cada uno de los recursos asociados** a la misma.



Fuente: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview>

HTTP – Stateless o sin estado

- El protocolo **no establece vínculo entre dos peticiones consecutivas de un mismo cliente**. Posibles soluciones:

- **URL-Rewriting**: Estado como pares de clave-valor en la URL:

`http://localhost:8080/productos?pagina=4&busqueda=libros`

- **Campos ocultos** en formularios:

`<input type="hidden" name="session" value="af2b5"/>`

- **Cookies**: Almacenar datos en texto plano en el cliente, viajando en cada petición (cabeceras Set-Cookie y Cookie).
- **Sesión**: El servidor mantiene el estado con un objeto. El objeto se vincula a cada agente mediante una cookie con un identificador único.

cURL – Peticiones HTTP en línea de comandos

- cURL: Herramienta de línea de comandos para transferencia de archivos.
- Soporta múltiple protocolos: HTTP, HTTPS, IMAP, LDAP, FTP...
- URL: <https://curl.se/>

cURL – Ejemplos de peticiones

- Una petición básica GET enviando parámetro en la URL:

```
curl --verbose http://localhost:8080/sdi-sesion1/GreetingServlet?name=Pepe
```

- cURL ofrece la posibilidad de enviar cookies:

```
curl --cookie "username=admin" http://localhost:8080/sdi-sesion1/GreetingServlet
```

- Establecer un user-agent determinado:

```
curl --user-agent "nadie" http://localhost:8080/sdi-sesion1/GreetingServlet
```

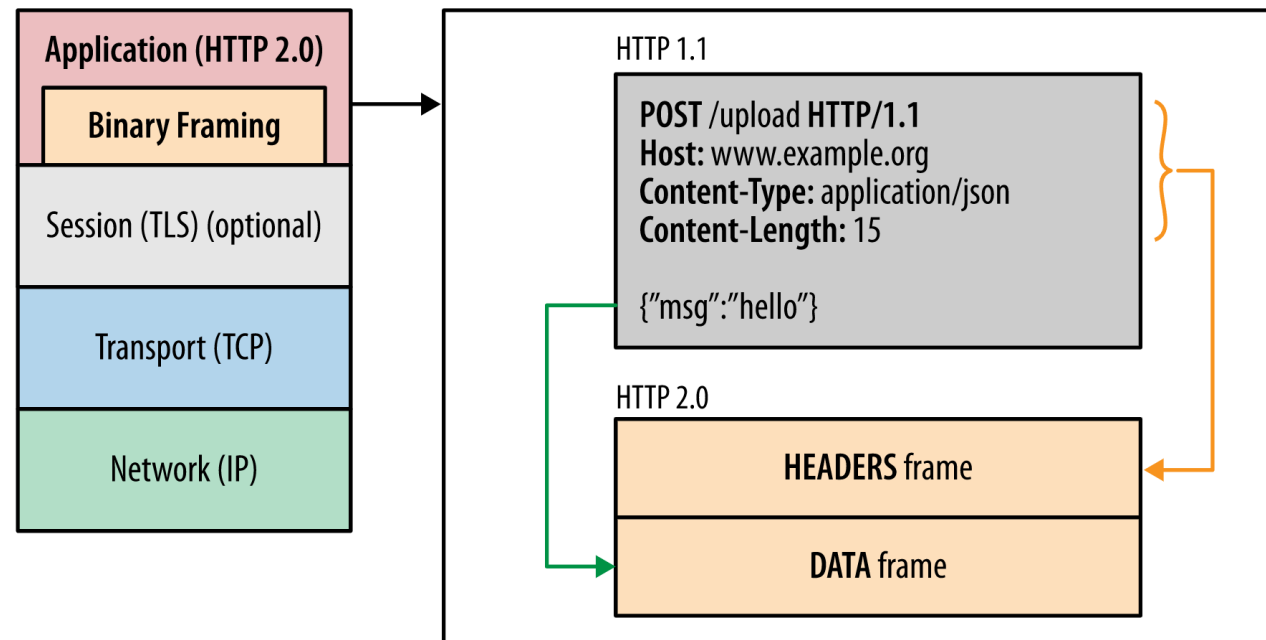
- Realizar peticiones POST:

```
curl --verbose -d "name=Pepe" http://localhost:8080/sdi-sesion1/GreetingServlet
```

- Pruebas POST contra <https://www.hashemian.com/tools/form-post-tester.php>

HTTP – Comparativa de versiones I

- HTTP/1.1 envía en texto plano, **HTTP/2** crea un **enmarcado binario** para encapsular y transferir la petición.
 - Por lo tanto, HTTP/2 tiene mejor rendimiento.



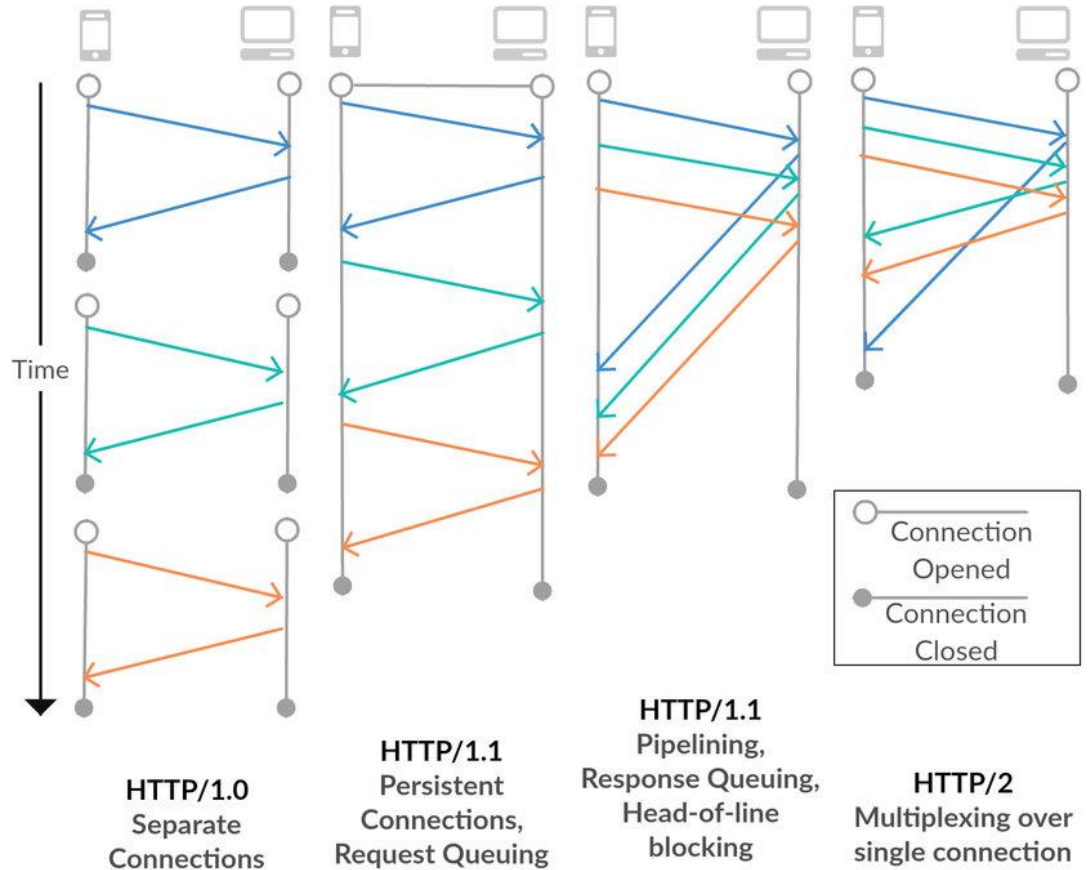
HTTP – Comparativa de versiones II

HTTP/1.1 produce **bloqueos en la cola de respuesta**.

HTTP/2 utiliza **multiplexado**.

HTTP/1 usa inlining, spriting...; para reducir el número de peticiones, generando **recursos no cacheables**.

HTTP/2 incluye **Server Push**. El servidor puede enviar recursos al cliente antes de que los solicite. **HPACK**, además, comprime las cabeceras.



Manzoor, J., Drago, I., & Sadre, R. (2016, October). The curious case of parallel connections in http/2. In *2016 12th International Conference on Network and Service Management (CNSM)* (pp. 174-180). IEEE.

HTTP - Comparativa de versiones III

- Para visualizar ese beneficio de rendimiento de HTTP/2 vs HTTP/1.1 podemos consultar:

<https://imagekit.io/demo/http2-vs-http1>

<http://www.http2demo.io/>

- Artículo para profundizar en:

<https://evertpot.com/h2-parallelism/>

HTTP/3

- **HTTP/2 (y anteriores) se asienta sobre TCP**, que garantiza que los mensajes (divididos en paquetes) llegan al receptor correctamente.
 - Si se sufre una pérdida de paquetes, se bloquean los recibidos hasta recibir el resto correctamente.
- **HTTP/3 se asienta sobre QUIC** (Quick **UDP** Internet Connections).
 - UDP no se preocupa de si llegan o no llegan los paquetes.
 - La integridad de los datos ya no es responsabilidad del protocolo de transferencia.

<https://blog.cloudflare.com/http-3-vs-http-2>