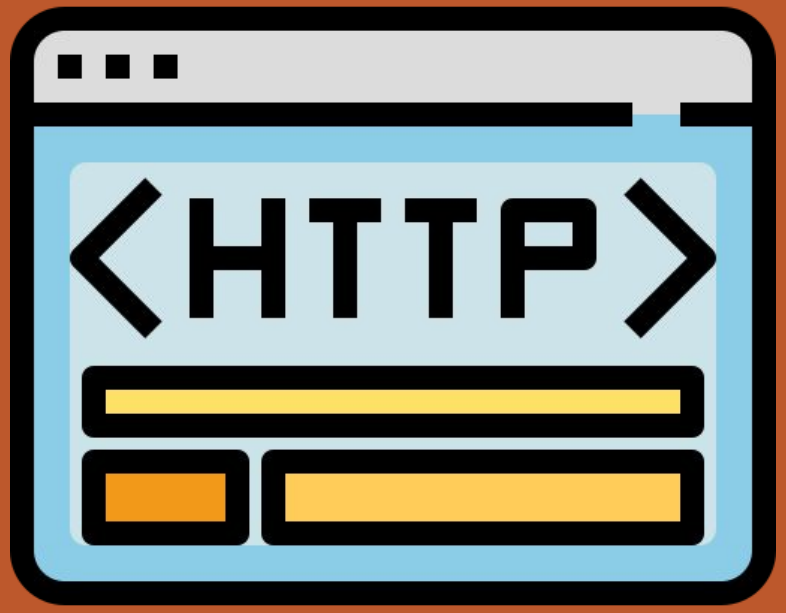
HTTP

TEMA 5 - DAW



Hyper Text Transfer Protocol

Diseñado para trasferir hipertexto y otra información entre clientes y servidores web

Capa de aplicación

Varias versiones conviviendo, actualmente HTTP/2

- HTTP/3.0 en pruebas

Web	Correo-e		FTP	News
HTTP	POP3	SMTP	FTP	NNTP
TCP/IP				
Red física				

Características:

- Modelo cliente/servidor
- Hasta la versión HTTP/2 usa TCP y el puerto 80
- HTTP/3 usa UDP
- No cifrado (la versión cifrada de HTTP es HTTPS)

Protocolo

- Esquema petición-respuesta

  Cliente envía un mensaje de petición y el servidor envía mensaje de respuesta

- No tiene estado

  El servidor cierra la conexión tras la respuesta

  A menudo las webs necesitan un estado: cookies

- Basado en mensajes

  En texto plano: más legibles, pero mayor tamaño

Escenario típico

1. Se crea una conexión HTTP

2. Cliente envía un mensaje de petición HTTP
   - Recurso que el cliente quiere
   - Información que quiere dar al servidor

3. Servidor devuelve mensaje de respuesta HTTP
   - Indica si la petición es exitosa
   - El contenido demandado, si procede

Mensajes

- En texto plano
- Estructura general:

<línea inicial>	Indica la naturaleza del mensaje. Diferente en mensajes de petición o de respuesta
<cabeceras>	Una o más cabeceras con metadatos que incluyen información esencial a la transacción. Formato: **<header-name>: <header-value>**
<CRLF>	Línea vacía, incluso si no hay cabeceras
[<cuerpo del mensaje>]	Sólo si procede. En respuestas, suele incluir el archivo o recurso pedido por el cliente, llamado **entidad**

Mensajes de petición

La primera línea tiene esta forma:

<METHOD> <request-uri> <HTTP-VERSION>

- **Method**: acción que se quiere hacer
  - Siempre en mayúsuculas
  - GET, HEAD, POST, etc
- **Request-uri**: el recurso sobre el que se quiere llevar a cabo la acción
  - **URL completa o solo path,** el host se pone en una cabecera de tipo **Host** (única cabecera obligatoria)
- **HTTP-Version**: indica al servidor la versión HTTP que usa el cliente

- Ejemplo: **`GET /index.html HTTP/1.0`**
  `Host: www.example.com`

```
<línea inicial>
<cabeceras>
<CRLF>
[<cuerpo del mensaje>]
```

Mensajes de respuesta

Cada petición genera una respuesta indicando resultado y entidad

<HTTP-VERSION> <status-code> <reason-phrase>

- **HTTP-Version**: indica al cliente la versión HTTP que usa el servidor
  - Siempre menor o igual a la del cliente
- **Status-code**: tres números que indican el resultado de la petición
- **Reason-phrase**: texto adicional que explica el resultado de forma más legible

```
<línea inicial>
<cabeceras>
<CRLF>
[<cuerpo del mensaje>]
```

Ejemplo de mensajes

PETICIÓN

```
GET /index.html HTTP/1.1
Host: www.example.com
Referer: www.google.com
User-Agent: Mozilla/5.0 (X11;
  Linux x86_64; rv:45.0)
  Gecko/20100101 Firefox/45.0
Connection: keep-alive
[Línea en blanco]
```

RESPUESTA

```
HTTP/1.1 200 OK
Date: Fri, 31 Dec 2003 23:59:59 GMT
Content-Type: text/html
Content-Length: 1221

<html>
  <head> .... </head>
  <body> .... </body>
</html>
```

Métodos de petición (I)

- **GET**: Solicita un documento al servidor.
  - Se pueden enviar datos en la URL
- **HEAD**: Similar a GET, pero sólo pide las cabeceras HTTP.
  - Comprobar enlaces
  - Para consultar información sobre el fichero (fecha de modificación, tamaño, tipo de servidor, tipo de documento solicitado) antes de solicitarlo.
- **POST**: Manda datos al servidor para su procesado.
  - Similar a GET, pero además envía datos en el cuerpo del mensaje.
  - La URL corresponde a un página dinámica que trata los datos enviados.

Métodos de petición (II)

- **PUT**: Almacena el documento enviado en el cuerpo del mensaje.
  - La URL hace referencia a la entidad a almacenar
  - Poco usado por razones de seguridad
- **DELETE**: Elimina el documento referenciado en la URL.
  - La URL hace referencia a la entidad a eliminar
  - Poco usado por razones de seguridad
- **TRACE**: Rastrea los intermediarios por los que pasa la petición.
- **OPTIONS**: Averigua los métodos que soporta el servidor.

GET

Sintaxis: **`GET <URL> <VERSION>`**

Solicita el recurso nombrado en la URL

- Recurso estático o dinámico (con o sin parámetros)

Variantes (para reducir el trafico en la red):

- GET condicional: Baja el recurso sólo bajo ciertas condiciones
  - Añadiendo las cabeceras:
    - **`If-Modified-Since`**, **`If-Match`**, **`If-Range`**, etc.
- GET parcial: descarga sólo ciertas partes del recurso
  - Añadiendo la cabecera:
    - **`Range: bytes=...`**

GET - Ejemplos

```
GET http://www.uv.es/index.html HTTP/1.1
Host: www.uv.es
If-Modified-Since: Fri, 1 Feb 2004 13:53:40 GMT
```

```
HTTP/1.0 304 Not Modified
Date: Thu, 1 Mar 2004 13:55:13 GMT
Content-Type: text/html
Expires: Fri, 30 Apr 2004 13:55:13 GMT
```

```
GET /Default.htm HTTP/1.1
Host: www.microsoft.com
Range: bytes=0-80
```

```
HTTP/1.1 206 Partial content
Server: Microsoft-IIS/5.0
Content-Type: text/html
Content-Length: 81
Content-Range: bytes 0-80/19618

<HTML> ....
```

POST

Sintaxis:

```
POST <URL> <VERSION>
```

Proporciona datos al recurso nombrado en la URL.

Los datos son enviados en el cuerpo del mensaje.

Códigos de respuesta:

- 200 OK
- 204 No Content
- 201 Created (cabecera location)

POST - Ejemplos

```
POST /test.cgi HTTP/1.0
Host: www.teco.edu:8080
User-Agent: Mozilla/4.7 (compatible; MSIE 5.0; Windows 5.0)
Content-Type: application/x-www-form-urlencoded
Content-Length: 39

name=Marie&path=%2F&ort=Karlsruhe&submit=Submit+Request
```

```
HTTP/1.0 200 OK
Date: Wed, 27 Oct 1999 14:13:43 GMT
Server: Apache/1.2.1
Content-Type: text/html
Content-Length: 380

<html><head>
<title>CGI-Script</title> ...
```

Códigos de estado

1xx: Mensaje informativo.

2xx: Éxito

- 200 OK
- 201 Created
- 202 Accepted
- 204 No Content

3xx: Redirección

- 300 Multiple Choice
- 301 Moved Permanently
- 302 Found
- 304 Not Modified

4xx: Error del cliente

- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found

5xx: Error del servidor

- 500 Internal Server Error
- 501 Not Implemented
- 502 Bad Gateway
- 503 Service Unavailable

Cabeceras

HTTP/1.0: 16 cabeceras, ninguna obligatoria.

HTTP/1.1: 46 cabeceras, "**`Host`**:" obligatoria en las peticiones

Formato de las cabeceras

- **`Nombre:`** `bsp` **`VALOR`** `CRLF`
- Mismo formato que las cabeceras de correo y News (MIME).

Clasificación:

- **Genéricas**: cliente y servidor
- **Exclusivas de la petición** (información del cliente)
- **Exclusivas de la respuesta** (información del Servidor)
- **Entidad del cuerpo del mensaje**

Se recomienda incluir en las peticiones al menos **`User-Agent`** y en las respuestas **`Server`** y **`Last-Modified`**.

Algunas cabeceras (I)

Genéricas

- **`Date:`** fecha de creación del mensaje
- **`Cache-Control:`** controla el comportamiento de la caché

De la Petición:

- **`Accept:`** tipos MIME aceptados por el navegador
- **`Host:`** nombre y puerto del servidor al que se dirige la petición
- **`User-Agent:`** identificación del programa del cliente
- **`Cookies:`** nombres y valores de las cookies.

Algunas cabeceras (II)

De la Respuesta:

- **`Server:`** versión del software del servidor
- **`Location:`** localización real del recurso. Permite redireccionar
- **`WWW-Authenticate:`** solicita autenticación
- **`Set-Cookies:`** envía cookie al cliente

De la Entidad:

- **`Content-Type:`** tipo MIME de la entidad (text/html, image/png, etc
- **`Content-Length:`** longitud de la entidad (importante en la petición)
- **`Last-Modified:`** fecha de última modificación
- **`Expires:`** fecha tope de validez en caché

MÁS INFORMACIÓN

Ejemplo en el navegador

Abre el Inspector (F12), pestaña Red

Cachés

Esencial en HTTP

- Usuarios suelen pedir los mismos recursos varias veces
- Una petición web desencadena peticiones de otras muchas entidades

Ventajas

- Reduce ancho de banda utilizado
- Mejora tiempo de respuesta

El cacheo se puede hacer en cliente o en puntos intermedios

- Cuanto más cerca del usuario, más eficiencia (velocidad y ancho de banda)
- Cuanto más lejos del usuario, más usuarios se benefician de la caché

Cookies

Son información que el navegador guarda a solicitud del servidor

- Asociadas a un nombre de dominio y path
- Tienen fecha de caducidad

HTTP no almacena estado entre peticiones sucesivas

- Las cookies pueden usarse para eso: inicios de sesión, personalización, etc
- Libera carga al servidor

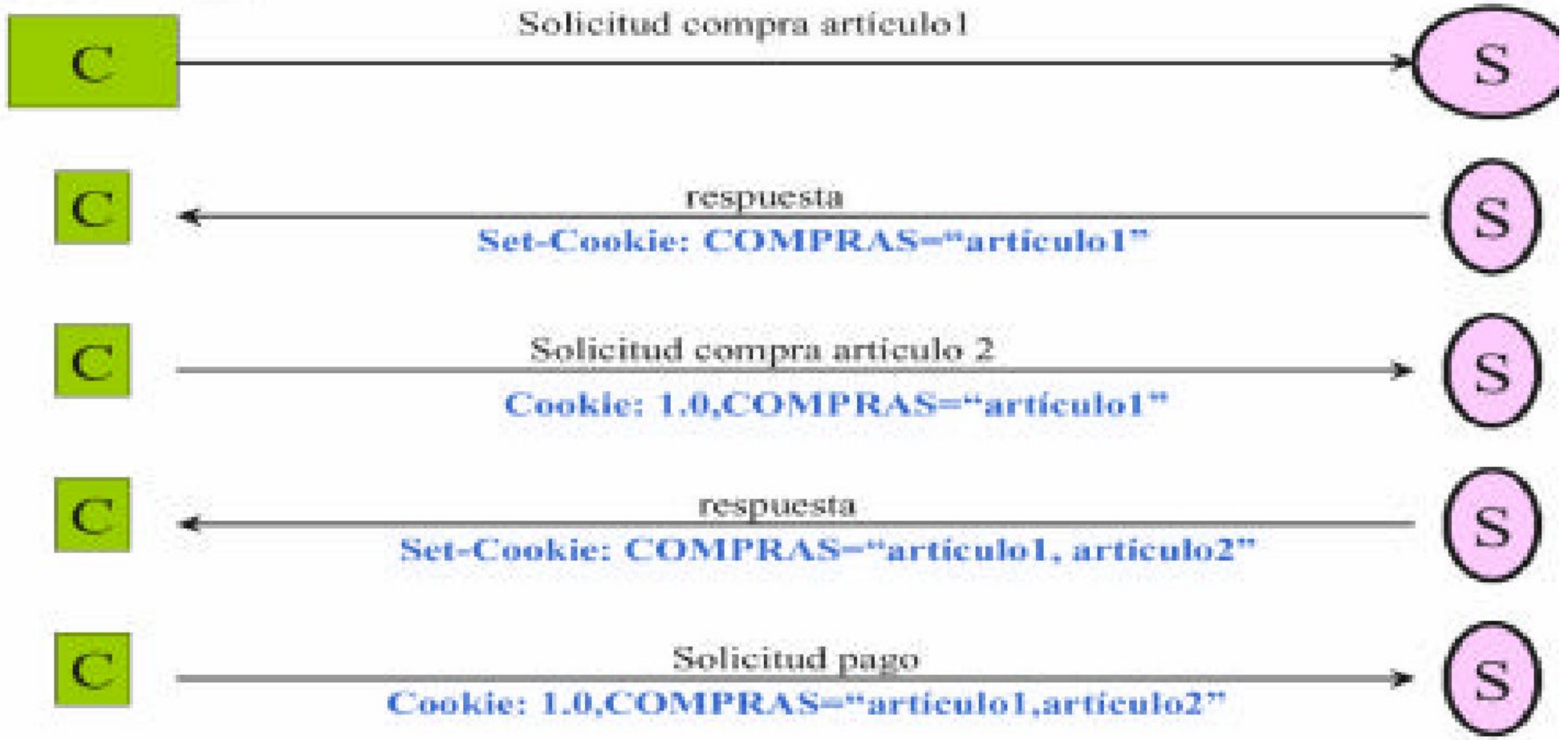
Servidor envía cabecera **`Set-Cookie`** con información a guardar

- Navegador almacena información

Cuando el navegador solicita una URL:

- Busca las cookies asociadas a esa URL
- Las envía en la cabecera **`Cookie`**

Cookies - Ejemplo



Ejemplo en el navegador

Abre el Inspector (F12), pestaña Almacenamiento