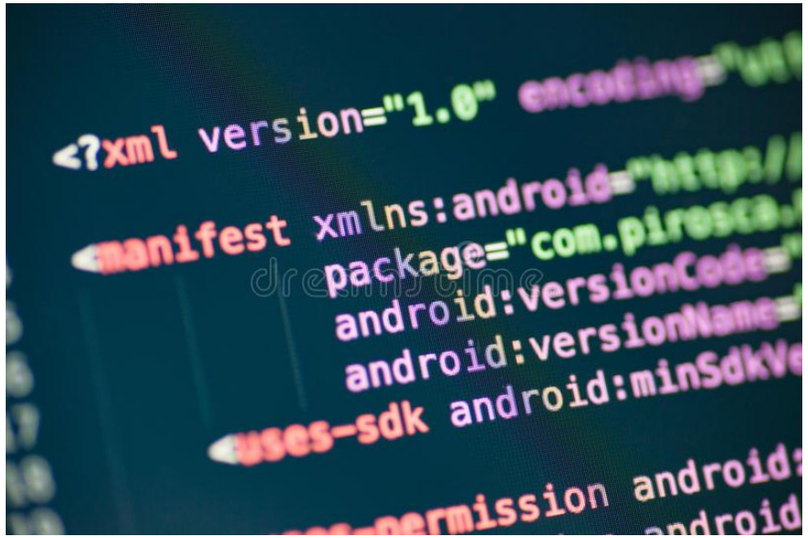
Lenguajes de marcas y sistemas de gestión de la información



UT02 – HTML

7 – Protocolo HTTP

IES Clara del Rey – Madrid

¿Cómo funciona la web?

Lo que normalmente denominamos “la web” es un sistema que permite la transferencia de distintos tipos de información.

Permite compartir distintos tipos de datos a través de Internet:

- Documentos de hipertexto (HTML)
- Imágenes
- Video
- Audio
- Programas
- Otros tipos de información

Componentes de la web

La web funciona apoyándose en una serie de elementos necesarios para que pueda prestar su servicio:

- Servidores: son los servicios (programas) que ponen algún tipo de información a disposición de los usuarios. Servidores web ,más usados: Apache web server, NGINX, IIS (Windows server).
- Clientes: las herramientas que se utilizan para acceder a esta información publicada por los servidores. Navegadores web: Chrome, Firefox, Edge, Opera, Vivaldi, etc.
- Tecnologías: una serie de lenguajes, protocolos y estándares necesarios para poder orquestar la comunicación entre el cliente y el servidor. HTML, CSS, JavaScript, HTTP, HTTPS, DNS, etc.

Elementos más relevantes

Los elementos más destacables en el funcionamiento de la web son:

- El lenguaje HTML. Permite definir la estructura de la información y establecer relaciones entre distintos recursos (páginas, imágenes, vídeos, audio, etc.). Se centra en la semántica, en el significado, no en la forma.
- El lenguaje CSS. Permite dar forma (y cierto grado de interactividad) a los elementos definidos en el HTML. Se encarga de definir colores, posición, tamaños y distribución de los contenidos.
- El lenguaje JavaScript. Permite añadir más interactividad y comportamiento a los documentos HTML.
- El protocolo HTTP. Es el que permite la transferencia de contenidos.

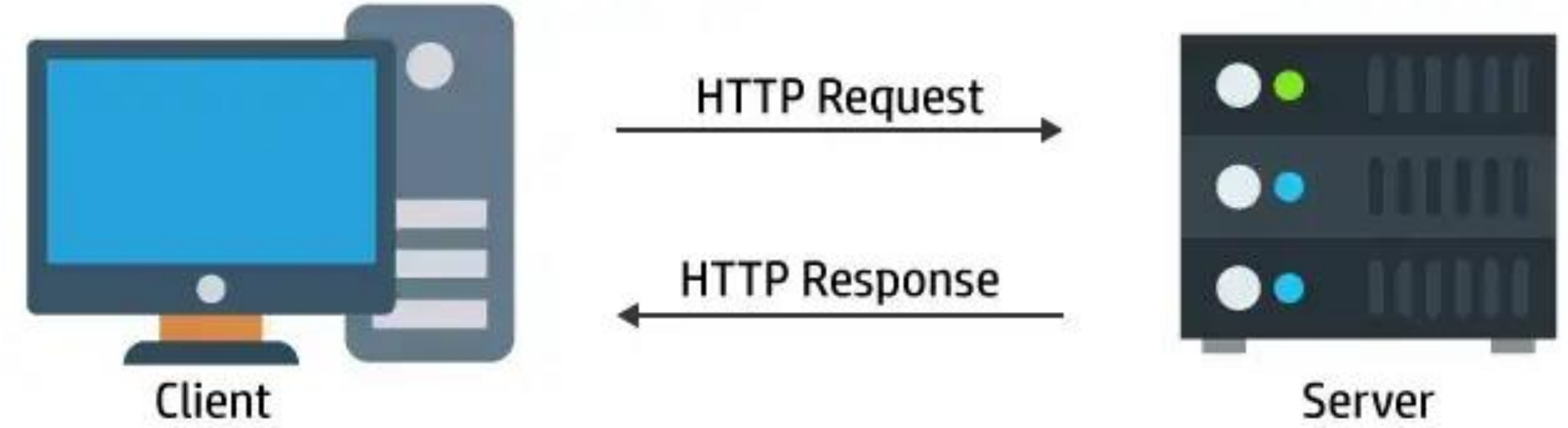
El protocolo HTTP

Es un protocolo especialmente diseñado para la transferencia de documentos hipermedia, como HTML, pero que se ha extendido con los años para poder transferir cualquier clase de medios.

El protocolo sigue el paradigma cliente-servidor, en el que:

- El servidor está a la espera de recibir conexiones.
- El cliente inicia la conexión y realiza una petición, en la que solicita ciertos datos (un documento HTML, una imagen, un fichero PDF...).
- Una vez realizada, el cliente espera la respuesta del servidor.
- El servidor responde a la petición con los datos solicitados.
- Se cierra la conexión.

El protocolo HTTP



Los mensajes del cliente al servidor se denominan peticiones (request).

Los mensajes del servidor al cliente se denominan respuestas (response).

Ambos mensajes son mensajes de texto, con una estructura definida, que se envían en texto plano (sin codificar). En HTTP 2, se codifican como una estructura binaria para ser más eficientes.

Estructura de una petición HTTP

Es un mensaje de texto que el cliente envía al servidor. Consta de:

- Línea de inicio. Una línea con tres elementos separados por espacios:
  - El método HTTP. Veremos los métodos más adelante.
  - El recurso solicitado, la página, fichero u otro recurso que se pide.
  - La versión de HTTP que se está usando para formar el mensaje.
- Cabeceras. Una línea por cabecera, terminadas con una línea en blanco. Cada línea de cabecera tiene dos partes separadas por “:”:
  - Nombre de la cabecera (User-Agent, Accept, Content-Length, etc.)
  - Valor de la cabecera.
- Cuerpo. Datos enviados al servidor. Las cabeceras Content-Type y Content-Length indican tipo de datos, y su tamaño.

Ejemplos de peticiones HTTP

Petición que pide la página inicial del sitio web www.nginx.com.

En la primera línea se ve el método (get), el recurso solicitado (/) y la versión del protocolo HTTP (HTTP/1.1)

El resto de las líneas son cabeceras para dar más información al servidor.

Esta petición no tiene cuerpo.

```
GET / HTTP/1.1
Host: www.nginx.com
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Accept:text/html
Accept-Encoding: gzip, deflate
Accept-Language: es-ES,es;q=0.9,en;q=0.8
```

Ejemplos de peticiones HTTP

Petición que envía un formulario a una página de login en el servidor www.pruebas.com. Esta petición tiene un cuerpo que incluye las credenciales enviadas para validar. El cuerpo se separa de las cabeceras por una línea vacía.

```
POST /login HTTP/1.1
Host: www.pruebas.com
Connection: keep-alive
Accept:text/html
Accept-Encoding: gzip, deflate
Accept-Language: es-ES,es;q=0.9,en;q=0.8

username=prueba-usuario&password=prueba-password
```

Estructura de una respuesta HTTP

Es también un mensaje de texto, pero se envía del servidor al cliente, como respuesta a una petición. Consta de:

- Línea de estado. Una línea con tres elementos separados por espacios:
  - Versión del protocolo.
  - Código de estado. Veremos algunos códigos más adelante.
  - Texto de estado. Una explicación del código de estado.
- Cabeceras. Siguen la misma estructura que las peticiones.
- Cuerpo. Datos enviados desde el servidor al cliente. Las cabeceras Content-Type y Content-Length indican tipo de datos, y su tamaño.

Ejemplos de respuestas HTTP

Respuesta que devuelve la página web inicial de NGINX cuando se solicita por HTTP. Es la respuesta a la petición de ejemplo anterior.

```
HTTP/1.1 307 Internal Redirect
Location: https://www.nginx.com/
Cross-Origin-Resource-Policy: Cross-Origin
Non-Authoritative-Reason: HSTS
```

Es una respuesta un poco especial. En lugar de responder con la página solicitada, redirige a la página inicial, pero con el protocolo HTTPS.

Ejemplos de respuestas HTTP

Respuesta que devolvería la página si admitiera el protocolo HTTP:

```
HTTP/1.1 200 OK
Content-type: text/html
Content-length: 353
--- Otras cabeceras ---

<!DOCTYPE html>
 <html lang="en-US">
 <head>NGINX<script>
 --- Más HTML---
```

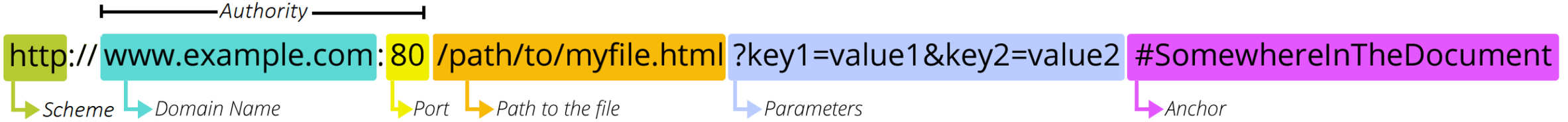
El cuerpo de la respuesta es una página HTML.

Está separado de las cabeceras por una línea en blanco.

Este HTML es el que el navegador interpreta para mostrarnos la página web.

URL

Uniform Resource Locator, especifica dónde se puede encontrar un recurso en Internet. Un recurso es una página web, una imagen, un fichero, un video, etc. Las URL Están formadas por:



Esquema: el protocolo que hay que usar para acceder al recurso.

Autoridad / dominio: el servidor al que hacemos la petición.

Puerto: el puerto TCP. Opcional. Si no aparece: 80 HTTP y 443 HTTPS.

Path: la ubicación del recurso dentro del servidor. Opcional.

Parámetros: Opcionales. Más información proporcionada al recurso.

Ancla o marcador: un punto concreto dentro del recurso.

Métodos HTTP

Indican qué operación se desea realizar sobre el recurso.

- GET: obtener el recurso. Las peticiones GET solo deberían recuperar datos, no enviar datos al servidor (aunque veremos que es posible). Es el método que se usa al hacer clic en enlaces.
- POST: enviar el recurso al servidor, modificándolo o provocando otros efectos. Este es el método que se suele usar en formularios.
- HEAD: obtiene información del recurso, pero no el recurso en sí. Obtiene sólo la línea de estado y las cabeceras, no el cuerpo.
- PUT: reemplazar el recurso con los datos del cuerpo de la petición.
- DELETE: eliminar el recurso.
- Otros: CONNECT, TRACE, OPTIONS, PATCH

Códigos de respuesta HTTP

El servidor web responde a todas las peticiones con un código de respuesta. Este código es un número que se agrupa en tramos:

- 100 – 199: Informativo. Indican que se está procesando la petición y que responderá más adelante con otro mensaje.
- 200 – 299: Éxito. La petición se ha podido completar. Éxito significa cosas diferentes según la petición. Para un GET es que se ha podido devolver el recurso. Para un POST es que se ha podido procesar la información.
- 300 – 399: Redirección. El recurso ya no está en esa dirección, hay que ir a otra.
- 400 – 499: error de cliente. Error en los datos de la petición.
- 500 – 599: error de servidor. Error de servidor al procesar la petición.

Ejemplos de códigos HTTP

- 200: OK. El más habitual. El que se suele devolver cuando pedimos una página web y no hay problemas al obtenerla.
- 404: Not found. El servidor no encuentra el recurso que estamos solicitando. Es un error de cliente porque el cliente ha pedido una URL que no representa un recurso del servidor. Es una URL incorrecta.
- 301: Moved permanently. El recurso ya no está en esa ubicación, se ha movido para siempre a otra URL. Se acompaña de la cabecera “location” para indicar la nueva ubicación.
- Otras redirecciones: 302 Found, 303 See other, 307 Temporary redirect, 308 Permanent redirect,
- 304: Not modified. El recurso no se ha modificado, puede usarse la versión cacheada, si ya la tenía. No incluye cuerpo, solo cabeceras.