# Introducción al protocolo HTTP

Despliegue de Aplicaciones Web

José Juan Sánchez Hernández

Curso 2024/2025

# Índice


**1 ¿Qué es el protocolo HTTP? 1**

**2 Cliente-Servidor 2**

**3 ¿Qué puerto utiliza HTTP? 3**

**4 Peticiones y Respuestas 4**

4.1 Ejemplo de petición (Request) . . . . . 4

4.2 Ejemplo de respuesta (Response) . . . . . 4

4.3 Ejercicio 1 . . . . . 5

4.4 Ejercicio 2 . . . . . 5

**5 Métodos HTTP 6**

**6 URL y Recursos 7**

**7 Códigos de estado de las respuestas HTTP 8**

7.1 Ejercicio: . . . . . 9

**8 Encabezados HTTP 10**

**9 Versiones del protocolo HTTP 11**

9.1 Evolución del protocolo HTTP . . . . . 12

**10 Referencias 14**

**11 Licencia 15**

# Índice de figuras

# Índice de cuadros

# 1 ¿Qué es el protocolo HTTP?

**HTTP** es un protocolo de la capa de aplicación que establece las reglas que deben seguir un **cliente** y un **servidor** para comunicarse en la **web** (World Wide Web). Su principal objetivo es permitir que los usuarios soliciten y recuperen recursos, como páginas web, imágenes y archivos, desde servidores en la web.

# 2 Cliente-Servidor

En las comunicaciones HTTP existen dos roles principales:

- **Cliente:** Es el software o equipo que realiza peticiones a un servidor. *Ejemplo:* Un navegador web, como Chrome o Firefox.
- **Servidor:** El servidor HTTP es un software o equipo que almacena y entrega recursos solicitados por los clientes. *Ejemplo*: Los servidores web, como Apache, Nginx o Microsoft IIS.

# 3 ¿Qué puerto utiliza HTTP?

El protocolo HTTP utiliza el puerto `80`, mientras que el protocolo HTTPS utiliza el puerto `443`.

# 4 Peticiones y Respuestas

La comunicación en el protocolo HTTP se basa en un modelo basado en **peticiones** y **respuestas**:

- **Petición (Request):** El cliente envía una petición al servidor para recuperar un recurso específico. La petición incluye información como el método HTTP seleccionado (`GET`, `POST`, etc.), la URL del recurso y los encabezados HTTP necesarios.
- **Respuesta (Response):** El servidor recibe la petición y responde con la información solicitada, junto con un código de estado HTTP (Ejemplo: `200 OK` o `404 Not Found`) y encabezados adicionales. La respuesta puede incluir datos en HTML, imágenes, archivos JSON, etc.

## 4.1 Ejemplo de petición (Request)

```
GET /index.html HTTP/1.1
Host: www.iescelia.org
User-Agent: MiNavegador/1.0
Accept: text/html
```

En este ejemplo aparecen los siguientes elementos:

- **`GET`** es el método HTTP utilizado para solicitar un recurso.
- **`/index.html`** es la ruta del recurso que se está solicitando en el servidor.
- **`HTTP/1.1`** indica la versión del protocolo HTTP que se está utilizando.
- **`Host: www.iescelia.org`** especifica el nombre del host al que se hace la petición.
- **`User-Agent: MiNavegador/1.0`** proporciona información sobre el agente de usuario o el navegador que realiza la petición.
- **`Accept: text/html`** indica que el cliente prefiere recibir el recurso en formato HTML.

## 4.2 Ejemplo de respuesta (Response)

```
HTTP/1.1 200 OK
Date: Miércoles, 28 de septiembre de 2023 13:30:00 GMT
Server: ServidorEjemplo/1.0
Content-Type: text/html
Content-Length: 1234

<!DOCTYPE html>
<html>
```

```
<head>
    <title>Página de Ejemplo</title>
</head>
<body>
    <p>Esta es una página de ejemplo.</p>
</body>
</html>
```

En la respuesta de ejemplo aparecen los siguientes elementos:

- **`HTTP/1.1 200 OK`**: indica que el recurso solicitado se encontró y se entregó correctamente.
- **`Date: Miércoles, 28 de septiembre de 2023 13:30:00 GMT`**: proporciona la fecha y la hora en que se generó la respuesta.
- **`Server ServidorEjemplo/1.0`**: indica el software del servidor web y su versión.
- **`Content-Type: text/html`**: indica que el tipo de contenido de la respuesta es HTML.
- **`Content-Length: 1234`**: indica el número de bytes del contenido de la respuesta.
- El resto del contenido del mensaje es el documento HTML que ha solicitado el cliente.

## 4.3 Ejercicio 1

Utiliza la herramienta `curl` para realizar una petición HTTP a la página web del IES Celia Viñas. ¿Qué información se muestra en la respuesta?

```
curl -v https://www.iescelia.org
```

El parámetro `-v` habilita el modo *verbose* del comando `curl` y muestra información más detallada sobre la petición y la respuesta.

Si queremos omitir el contenido de la respuesta, podemos utilizar el parámetro `-I`. Este parámetro hará que sólo se muestren las cabeceras.

```
curl -Iv https://www.iescelia.org
```

## 4.4 Ejercicio 2

Utiliza las **herramientas para desarrolladores** que se se incluyen en los navegadores **Google Chrome** o **Mozilla Firefox** para realizar una petición HTTP a la página del IES Celia Viñas y examinar el contenido de las cabeceras (*Headers*)), peticiones (*Request*) y respuestas (*Response*).

# 5 Métodos HTTP

En el protocolo HTTP se definen diferentes métodos que indican la acción que se debe realizar en una petición. Algunos de los métodos más comunes son:

- **GET:** Solicita un recurso específico.
- **POST:** Envía datos al servidor para procesarlos, por ejemplo, para enviar formularios.
- **PUT:** Actualiza un recurso existente o crea uno nuevo si no existe.
- **DELETE:** Elimina un recurso.
- **HEAD**: Solicita únicamente la cabecera de la respuesta HTTP.

# 6 URL y Recursos

Las **URL** (*Uniform Resource Locators*) son direcciones que identifican recursos en la web. Una URL típica consta de un esquema (como "`http://`" o "`https://`"), un dominio (como "`www.iescelia.org`") y una ruta que indica la ubicación del recurso en el servidor.

# 7 Códigos de estado de las respuestas HTTP

Cada respuesta HTTP incluye un código de estado que indica si la petición se pudo resolver con éxito o no. Existen cinco tipos de códigos de estado:

- **1xx - Respuestas informativas:**
  - `100 - Continue`: Indica que la petición ha sido recibida y el cliente puede continuar con su petición.
- **2xx - Respuestas correctas:**
  - `200 - OK`: La petición se completó con éxito.
- **3xx - Redirecciones:**
  - `301 - Moved Permanently`: Indica que el recurso solicitado se ha movido permanentemente a una nueva ubicación y se proporciona la URL actualizada.
- **4xx - Errores del cliente:**
  - `403 - Forbidden`: Indica que el servidor comprende la petición del cliente, pero no se permite su acceso.
  - `404 - Not Found`: El recurso solicitado no se encontró en el servidor.
- **5xx - Errores del Servidor:**
  - `500 Internal Server Error`: Hubo un error en el servidor al procesar la petición.

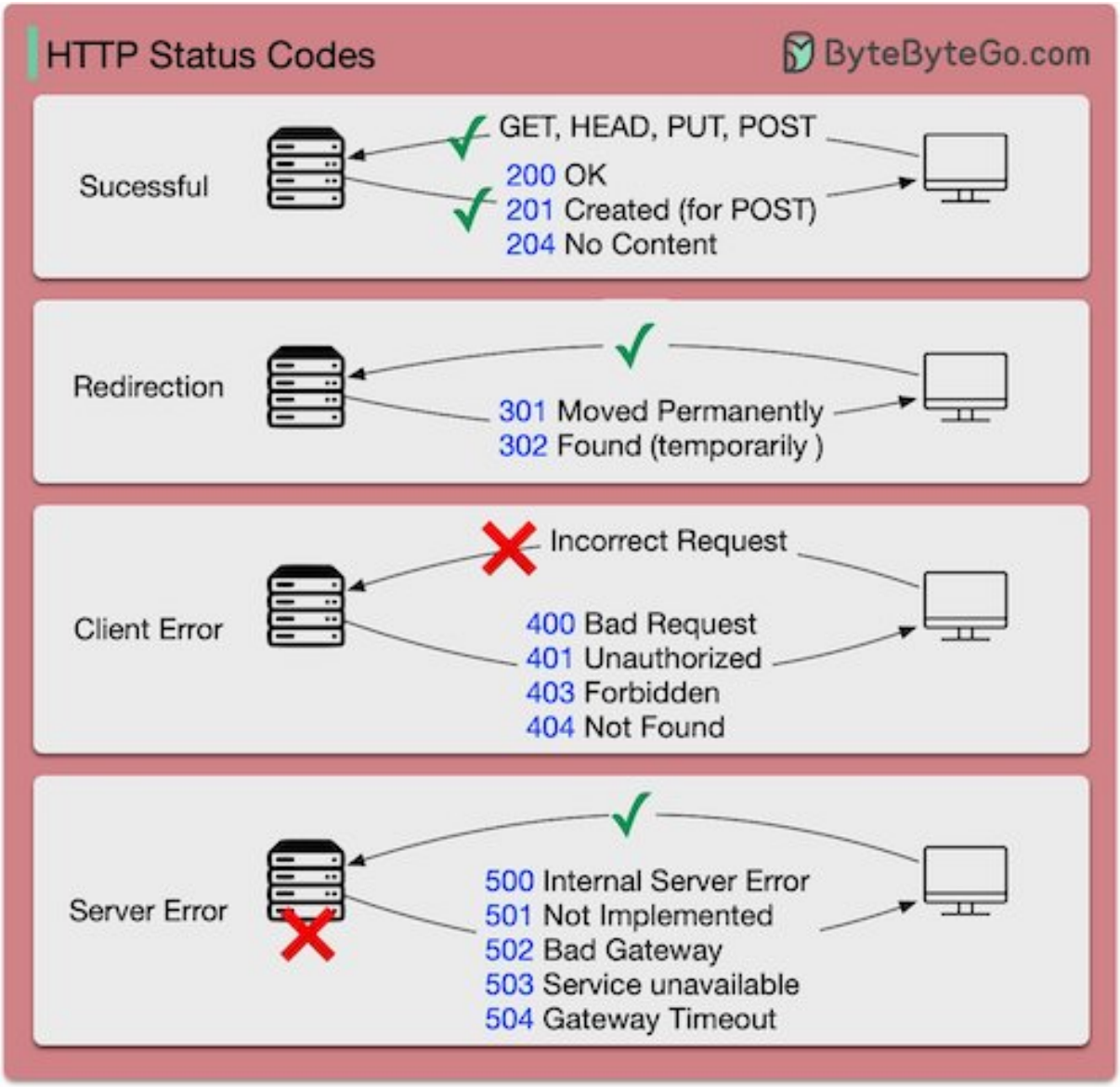


*Imagen 1: Imagen obtenida de ByteByteGo.com.*

## 7.1 Ejercicio:

Busca en Internet el significado del siguiente código de estado HTTP:

- 418.

Referencias:

- HTTP response status codes. MDN Web Docs.
- Códigos de estado HTTP. Wikipedia

# 8 Encabezados HTTP

El protocolo **HTTP** utiliza encabezados para transmitir información adicional tanto en las peticiones, como en las respuestas. Los encabezados pueden incluir información sobre el tipo de contenido, la longitud del contenido, la codificación, la autenticación, etc.

# 9 Versiones del protocolo HTTP

| Versión | Fecha de publicación | Características principales |
|---|---|---|
| HTTP/0.9 | 1991 | Se trata de una versión muy sencilla que sólo admite peticiones `GET` |
| HTTP/1.0 | 1996 | Se añaden más tipos de peticiones: `GET`, `POST`, `HEAD` y `PUT` |
| HTTP/1.1 | 1997 | Incluye mejoras de rendimiento, como la caché y conexiones persistentes |
| HTTP/2 | 2015 | Incluye mejoras de rendimiento, como la multiplexación y la compresión de cabeceras |
| HTTP/3 | 2018 | Utiliza el protocolo UDP para mejorar el rendimiento y la seguridad |

## 9.1 Evolución del protocolo HTTP

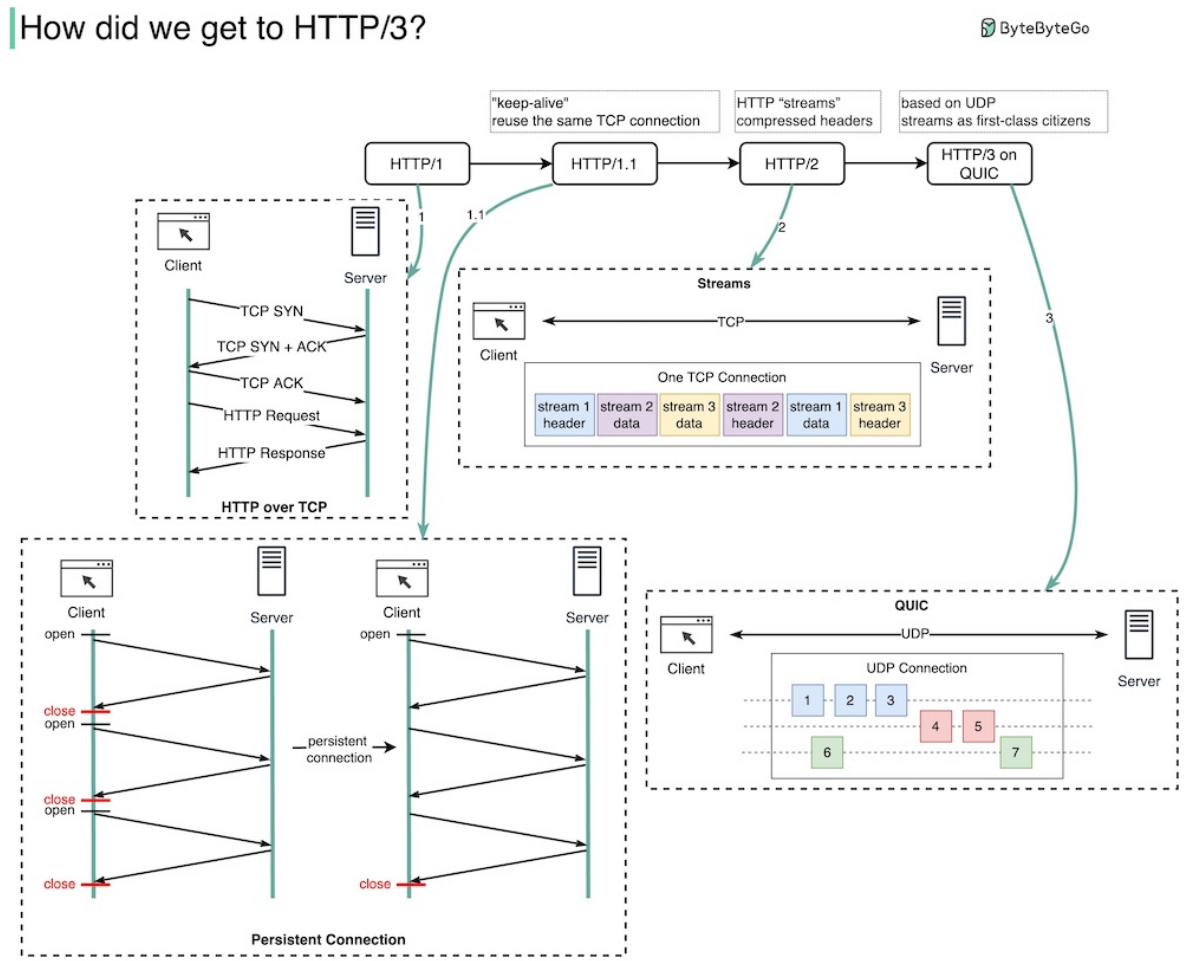


*Imagen 2: Imagen obtenida de ByteByteGo.com.*

- **HTTP 1.0**: Es necesario realizar una una conexión TCP separada para cada petición HTTP.
- **HTTP 1.1**: Permite utilizar conexiones persistentes, que consiste en mantener una conexión TCP abierta para que se pueda reutilizar. Esta versión no resuelve el problema de bloqueo HOL (Head-Of-Line).

  El bloqueo HOL ocurre cuando se agota el número de peticiones paralelas permitidas en el navegador, lo que hace que las peticiones posteriores deban esperar a que las anteriores se completen.
- **HTTP 2.0**: Resuelve el problema de bloqueo HOL a través de la multiplexación de peticiones, lo que elimina el bloqueo HOL en la capa de aplicación, aunque el bloqueo HOL todavía existe en la capa de transporte (TCP).

  HTTP 2.0 introdujo el concepto de *streams* de HTTP, que permite la multiplexación de diferentes intercambios de HTTP en la misma conexión TCP. Cada *stream* no necesita ser enviado en orden.
- **HTTP 3.0**: Utiliza QUIC en lugar de TCP como protocolo de transporte subyacente, eliminando así el bloqueo HOL en la capa de transporte.

Referencia:

- System Design 101. ByteByteGo

# 10 Referencias


- HTTP - Protocolo de transferencia de hipertexto. Wikipedia
- El protocolo HTTP. MDN Web Docs
- Modelo Cliente/Servidor. Wikipedia.
- HTTPS - Protocolo seguro de transferencia de hipertexto. Wikipedia
- Códigos de estado HTTP. Wikipedia
- URL (*Uniform Resource Locator*). Wikipedia
- HTTP/1 to HTTP/2 to HTTP/3. ByteByteGo
- System Design 101. ByteByteGo

# 11 Licencia

Esta página forma parte del curso Despliegue de Aplicaciones Web de José Juan Sánchez Hernández y su contenido se distribuye bajo una licencia Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional.