Webpage Request Lifecycle.

Luis Francisco Contreras González

Webpage Request Lifecycle.

Cuando escribimos una URL en el navegador y presionamos Enter, comienza un proceso llamado Webpage Request Lifecycle.

Este ciclo describe todo lo que pasa desde que se ingresa la dirección hasta que vemos la página cargada en la pantalla.

Componentes de una URL

https://www.algoexpert.io:443/frontend?max-price=10#content

Esquema/Protocolo: https

Subdominio: www

Dominio: algoexpert

TLD (Top-Level Domain): .io

Puerto: 443

Ruta (Path): /frontend

Parámetros (Query): ?max-price=10

Fragmento (Fragment): #content

Sistema de Nombres de Dominio (DNS)

Convierte los nombres de dominio en direcciones IP.

algoexpert.io → 35.202.194.70

Siempre revisa primero la caché local antes de consultar servidores externos.

Ciclo de Vida de una Petición Web

Entrada de la URL → El usuario escribe la dirección.

Resolución DNS → Convierte dominio a IP.

Conexión con el servidor → Se establece la comunicación (puerto, protocolo).

Solicitud HTTP/HTTPS → El navegador envía una petición.

Respuesta del servidor → Devuelve HTML, CSS, JS, imágenes.

Renderizado en navegador → Se construye el DOM y se muestra la página.

HTTP – Hypertext Transfer Protocol

Protocolo de red para enviar solicitudes y respuestas en la web.

Una solicitud HTTP tiene:

- Línea de solicitud (método, ruta, versión).
- Encabezados (headers).
- Cuerpo (body) (opcional).

Ejemplo de solicitud GET:

- GET / HTTP/1.1
- host: www.algoexpert.io
- accept: text/html

Respuesta HTTP

HTTP/1.1 200 OK

content-type: text/html

<!DOCTYPE html>

<html>

...

</html>

HTTPS – Hypertext Transfer Protocol Secure

Extensión de HTTP que añade seguridad.

Usa TLS (Transport Layer Security) sobre TCP.

Los datos entre cliente y servidor se transmiten cifrados.

Requiere certificados digitales confiables.

Ejemplo

http:// → comunicación sin cifrar

https:// → comunicación cifrada y segura

Paso a paso

Escribes https://www.algoexpert.io/frontend.

DNS traduce el dominio → 35.202.194.70.

Conexión → Puerto 443.

Solicitud:

GET /frontend HTTP/1.1

Respuesta:

HTTP/1.1 200 OK

El navegador renderiza la página.

Códigos de Estado HTTP

Los códigos de estado HTTP indican el resultado de una petición entre el cliente y el servidor.

Se clasifican en 5 grupos principales:

- 1xx – Informativos

Indican que la petición fue recibida y el proceso continúa.

Ejemplo: 100 Continue.

Códigos de Estado HTTP

- 2xx – Éxito

La petición se completó correctamente.

200 OK: Respuesta exitosa.

201 Created: Recurso creado.

Códigos de Estado HTTP

- 3xx – Redirecciones

Indican que el recurso fue movido.

301 Moved Permanently.

302 Found.

Códigos de Estado HTTP

- 4xx – Errores del cliente

Problema en la petición hecha por el cliente.

400 Bad Request: Solicitud malformada.

401 Unauthorized: Requiere autenticación.

403 Forbidden: Acceso denegado.

404 Not Found: Recurso no encontrado.

Códigos de Estado HTTP

- 5xx – Errores del servidor

El servidor no pudo procesar la petición.

500 Internal Server Error.

502 Bad Gateway.

503 Service Unavailable.

Métodos HTTP

Los métodos HTTP definen la acción que se quiere realizar sobre un recurso.

- GET

Obtiene información del servidor.

No modifica datos.

Ejemplo: Ver una página web.

Métodos HTTP

- POST

Envía datos al servidor para crear un recurso nuevo.

Ejemplo: Registrar un usuario.

Métodos HTTP

- PUT

Reemplaza completamente un recurso existente.

Ejemplo: Actualizar todos los datos de un perfil.

Métodos HTTP

- PATCH

Modifica parcialmente un recurso.

Ejemplo: Cambiar solo el email de un perfil.

Idempotencia en HTTP

Una operación es idempotente si al ejecutarla una o múltiples veces, el resultado es siempre el mismo.

Garantiza que repetir la misma petición no cambia el estado del servidor más de una vez.

Muy importante en APIs para evitar efectos secundarios.

Métodos idempotentes

GET → Leer información (no altera nada).

PUT → Reemplaza un recurso completo.

Hacer la misma actualización varias veces deja el mismo resultado.

DELETE → Eliminar un recurso (borrarlo varias veces produce el mismo estado: ya no existe).

HEAD y OPTIONS → También son idempotentes.

Métodos NO idempotentes

POST → Crear un recurso nuevo.

Repetirlo varias veces puede generar duplicados.

PATCH → Puede o no ser idempotente, depende de la implementación.

Ejemplo: “sumar +1 a un contador” no es idempotente.

Ejemplo sencillo:

PUT /user/1/email = "juan@mail.com"

→ Da el mismo resultado si se ejecuta 1 o 100 veces.

POST /user con datos de usuario

→ Cada ejecución crea un usuario nuevo.