UD2 Introducción a las arquitecturas web y servidores

5. Cómo funciona una petición web

Cada vez que accedemos a una página web —por ejemplo, al escribir `www.ejemplo.com` en el navegador— estamos realizando una **petición web**. Esta acción, que parece simple, activa una cadena compleja de procesos que involucra múltiples tecnologías y servicios de red. Comprender cómo funciona esta petición es clave para entender la arquitectura web y los elementos que intervienen en el despliegue de aplicaciones.

Veamos paso a paso qué sucede en una petición web típica:

5.1. El navegador interpreta la URL

Cuando el usuario introduce una dirección como `https://www.ejemplo.com`, el navegador analiza su estructura:

- `https` → protocolo (especifica cómo se comunicará con el servidor)
- `www.ejemplo.com` → nombre del dominio
- `/productos` → ruta o recurso solicitado (opcional)

5.2. Resolución de nombre: DNS

Antes de poder comunicarse con el servidor, el navegador necesita conocer su dirección IP (por ejemplo, `203.0.113.5`). Para ello, consulta un **servidor DNS** (Domain Name System), que traduce el nombre del dominio (`www.ejemplo.com`) a la IP correspondiente.

Este proceso puede implicar varias búsquedas (en caché local, en el sistema operativo, en servidores DNS públicos o del proveedor de Internet), pero suele completarse en milisegundos.

5.3. Establecimiento de la conexión

Una vez obtenida la IP, el navegador inicia una conexión con el servidor web:

- Si el protocolo es **HTTP**, se abre una conexión TCP con el puerto 80.
- Si es **HTTPS**, se abre una conexión segura (TLS/SSL) normalmente por el puerto 443.

En el caso de HTTPS, antes de enviar cualquier dato, se realiza un **intercambio de claves criptográficas** para cifrar la comunicación y garantizar la autenticidad del servidor (gracias a su certificado digital).

5.4. Envío de la petición

El navegador envía una petición HTTP al servidor. Esta incluye:

- **Método** (GET, POST, PUT, DELETE, etc.)
- **Ruta** del recurso solicitado
- **Cabeceras (headers)** con información sobre el navegador, idioma, tipo de contenido aceptado, cookies, etc.

Ejemplo de petición GET:

```
GET /productos HTTP/1.1
Host: www.ejemplo.com
User-Agent: Mozilla/5.0
Accept: text/html
```

5.5. Procesamiento en el servidor

El **servidor web** (por ejemplo, Apache o Nginx) recibe la petición y decide cómo procesarla. Dependiendo del tipo de recurso solicitado, puede:

- Servir un archivo estático (HTML, imagen, CSS, etc.)
- Reenviar la petición a una aplicación backend (por ejemplo, a un servidor Node.js o PHP)
- Comprobar reglas de redirección, autenticación o control de acceso
- Consultar una base de datos

En el caso de una aplicación dinámica, el servidor backend genera contenido personalizado (por ejemplo, una lista de productos extraída de la base de datos).

5.6. Envío de la respuesta

El servidor devuelve al navegador una **respuesta HTTP**, que incluye:

- Código de estado (200 OK, 404 Not Found, 500 Error interno…)
- Cabeceras (tipo de contenido, longitud, caché, cookies, etc.)

- Cuerpo del mensaje (el contenido solicitado: HTML, JSON, imagen…)

Ejemplo de respuesta:

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 1024

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

5.7. Renderizado en el navegador

El navegador interpreta el contenido HTML recibido y comienza a construir la página visualmente. Si hay recursos adicionales (CSS, JavaScript, imágenes), realiza nuevas peticiones para descargarlos.

Además, si la aplicación usa JavaScript moderno (por ejemplo, con React), parte del contenido puede cargarse dinámicamente desde una API.

5.8. Interacción posterior

Una vez cargada la página, el usuario puede interactuar (hacer clic, enviar formularios, cambiar filtros...). Muchas de estas acciones generan **nuevas peticiones web asincrónicas** mediante JavaScript (por ejemplo, con `fetch` o `axios`), sin recargar toda la página. Esto se conoce como AJAX (Asynchronous JavaScript and XML) o, en términos modernos, como aplicaciones SPA (Single Page Applications).

Resumen visual del flujo

```
[Usuario]
   ↓ (URL)
[Navegador]
   ↓ (DNS)
[Servidor DNS]
   ↓ (IP)
[Navegador]
   ↓ (HTTP/HTTPS)
[Servidor Web (Apache/Nginx)]
   ↓ (si necesario)
[Servidor de Aplicaciones]
   ↓
[Base de datos]
   ↑
[Servidor Web]
   ↑ (respuesta HTML/JSON)
[Navegador]
   ↓ (render)
[Usuario ve la web]
```

Este flujo básico puede complicarse según la arquitectura: puede haber múltiples servidores intermedios, sistemas de balanceo de carga, redes de entrega de contenido (CDNs), autenticación con tokens, compresión, almacenamiento en caché, etc. Aun así, este esquema sigue siendo la base sobre la que funcionan todas las aplicaciones web.