UD2 Introducción a las arquitecturas web y servidores

3. Componentes básicos: cliente, servidor, base de datos, DNS, etc

Para entender cómo funciona una arquitectura web, es esencial conocer los componentes principales que la conforman. Aunque las arquitecturas pueden variar en complejidad, casi todas comparten un conjunto de elementos básicos que colaboran entre sí para ofrecer una experiencia completa al usuario final.

A continuación, analizamos los componentes más comunes:

3.1 Cliente (cliente web)

El cliente es el dispositivo o software que inicia la comunicación con la aplicación web. En la mayoría de los casos, hablamos de un **navegador web** (como Chrome, Firefox o Safari), pero también pueden ser aplicaciones móviles, herramientas de línea de comandos o incluso otros servidores.

Funciones principales del cliente:

- Enviar peticiones HTTP al servidor.
- Mostrar la respuesta que llega (HTML, CSS, JS, imágenes…).
- Ejecutar código del lado cliente (JavaScript) para enriquecer la experiencia.

Ejemplo típico: cuando el usuario escribe una URL en el navegador, el cliente envía una petición al servidor y muestra la página recibida.

3.2 Servidor web

El **servidor web** es el programa (y máquina) que escucha peticiones entrantes del cliente y les da respuesta. Su trabajo puede ser tan simple como devolver archivos estáticos (HTML, imágenes, CSS) o tan complejo como actuar de intermediario entre el cliente y una aplicación más profunda.

Los servidores web más utilizados hoy en día son:

- **Apache HTTP Server**: el más veterano, modular y muy configurable.

- **Nginx**: moderno, eficiente con múltiples conexiones simultáneas, muy usado como proxy inverso.
- **LiteSpeed**, **Caddy** y otros también existen, aunque son menos comunes en entornos educativos.

Funciones típicas de un servidor web:

- Atender peticiones HTTP/HTTPS.
- Servir archivos estáticos.
- Redirigir peticiones a otros servidores o aplicaciones (proxy).
- Aplicar reglas de acceso, seguridad y compresión.

3.3 Servidor de aplicaciones

Un **servidor de aplicaciones** ejecuta la lógica de negocio de una aplicación web. Mientras que el servidor web sirve archivos o enruta peticiones, el servidor de aplicaciones procesa datos, accede a bases de datos y genera contenido dinámico.

Ejemplos comunes:

- **Apache Tomcat**: servidor de aplicaciones Java (servlets y JSP).
- **Node.js**: entorno de ejecución para aplicaciones JavaScript del lado servidor.
- **Django (Python), Spring Boot (Java), Express (Node.js)**: frameworks que suelen ejecutarse dentro o junto a un servidor de aplicaciones.

En arquitecturas modernas, el servidor web (como Nginx) suele actuar de intermediario, redirigiendo ciertas peticiones al servidor de aplicaciones.

3.4 Base de datos

Toda aplicación dinámica necesita guardar datos de forma persistente. Para ello, se utiliza un sistema de gestión de bases de datos (SGBD). Existen diferentes tipos:

- **Relacionales**: utilizan tablas estructuradas y SQL. Ej.: MySQL, PostgreSQL, Oracle.
- **No relacionales (NoSQL)**: más flexibles, basados en documentos, pares clave-valor, grafos… Ej.: MongoDB, Redis, Firebase.

El servidor de aplicaciones se conecta a la base de datos para:

- Recuperar datos (lectura).
- Guardar nuevos datos (escritura).

- Modificar o eliminar registros.
- Ejecutar consultas complejas (joins, agregaciones, etc.).

A este conjunto de operaciones se le conoce como **CRUD** (Create, Read, Update, Delete).

3.5 Sistema de nombres de dominio (DNS)

El **DNS** (*Domain Name System*) es uno de los pilares de internet. Su función es traducir **nombres de dominio** legibles (como `www.ejemplo.com`) en **direcciones IP** comprensibles para los equipos de red.

Cuando el cliente escribe una URL en el navegador, lo primero que hace su sistema operativo es consultar un servidor DNS para saber qué dirección IP corresponde a ese dominio. Solo entonces puede iniciarse la comunicación real con el servidor web.

Ejemplo:

- El usuario accede a `www.mitiendaonline.com`.
- El navegador consulta al DNS y obtiene la IP `185.67.89.22`.
- A partir de esa IP, se envía la petición HTTP al servidor correspondiente.

Importancia del DNS:

- Permite cambiar de servidor sin cambiar el dominio visible.
- Soporta configuraciones complejas (redirecciones, subdominios…).
- Es esencial en el despliegue profesional de aplicaciones.

3.6 Otros componentes

Además de los elementos principales, existen otros componentes que pueden intervenir en una arquitectura web moderna:

- **Proxy inverso**: servidor que actúa como intermediario entre el cliente y uno o varios servidores backend. Mejora la seguridad, la escalabilidad y el rendimiento. Ej.: Nginx actuando como proxy de una app Node.js.

- **CDN (Content Delivery Network)**: red de servidores distribuidos que alojan copias de archivos estáticos para entregarlos desde ubicaciones cercanas al usuario. Mejora la velocidad de carga y reduce el uso del servidor principal.

- **Balanceadores de carga**: distribuyen las peticiones entre varios servidores para repartir la carga y evitar saturación.

- **Firewalls y gateways**: controlan el tráfico, aplican reglas de seguridad y protegen la red de ataques externos.

Conclusión

Una arquitectura web moderna es un ecosistema complejo donde múltiples componentes trabajan de forma coordinada. Comprender qué papel juega cada uno de ellos es esencial para tomar decisiones de diseño, despliegue y mantenimiento de aplicaciones web. En la siguiente sección analizaremos en detalle **cómo funciona una petición web completa**, desde que el usuario escribe una URL hasta que recibe una página.