WebServices apunte

Un WebService es un **componente de software accesible a través de la red**, que permite que distintas aplicaciones, posiblemente escritas en lenguajes diferentes y ejecutándose en máquinas diferentes, puedan comunicarse entre sí mediante protocolos estándar.

Puede ser visto como una **funcionalidad compartida** que está alojada en un servidor y que puede ser **consumida remotamente** por otros sistemas.

¿Por qué usar Web Services?

- **Desarrollo independiente:** cada componente puede ser desarrollado por equipos distintos.
- **Despliegue independiente:** los servicios pueden actualizarse o desplegarse sin afectar al sistema completo.
- **Aislamiento de fallas:** un fallo en un servicio puede manejarse sin que colapse toda la aplicación.
- **Stack mixto:** los servicios pueden estar escritos en diferentes lenguajes o tecnologías.
- **Escalado granular:** se puede escalar solo el servicio necesario, en lugar de toda la aplicación.

Tipos de Servicios Web

1. SOAP (Simple Object Access Protocol)

- Basado en **XML**
- Requiere un formato rígido para solicitud y respuesta
- Protocolos como **WSDL** y **SOAP envelope** son necesarios
- Más formal, utilizado en entornos empresariales complejos

2. REST (Representational State Transfer)

- Estilo arquitectónico, no un protocolo
- Utiliza verbos HTTP: `GET`, `POST`, `PUT`, `DELETE`
- Usa formatos livianos como **JSON** o **XML**
- Más simple, flexible y ampliamente adoptado en desarrollo web

Estructura de una API RESTful

```
Cliente --GET/POST/PUT/DELETE--> API REST --> Base de datos
Cliente <-- JSON/XML ----------- API REST <-- Base de datos
```

Cada recurso (usuarios, productos, pedidos, etc.) se trata como una **entidad con estado** que puede ser manipulada a través de URLs y métodos HTTP.

Arquitecturas donde se usan Web Services

1. Monolítica

- Toda la lógica de negocio en un solo componente
- Fácil de desarrollar, difícil de escalar o mantener

2. SOA (Service-Oriented Architecture)

- Arquitectura orientada a servicios
- Mayor reutilización y modularidad
- La latencia puede aumentar por las múltiples capas

3. Microservicios

- Servicios pequeños, específicos y autónomos
- Independencia total entre componentes
- Permite escalado y despliegue granulares