Sistemas Distribuidos Tema 3: Servicios Web Distribuidos

Cosijopii García-García

UNISTMO

22 de abril de 2025

Contenido

¿Qué es un Servicio Web?

- Es una interfaz que permite la interoperabilidad entre aplicaciones a través de una red (típicamente Internet).
- Utiliza estándares abiertos como HTTP, XML/JSON, y protocolos como REST o SOAP.
- Se diseñan para ser independientes de plataformas y lenguajes.

Roles en la Arquitectura de Servicios Web

- **Proveedor del Servicio (Service Provider)**: Implementa y expone el servicio.
- **Consumidor del Servicio (Service Consumer)**: Llama al servicio y lo utiliza.
- **Registro del Servicio (Service Registry)**: Permite buscar y registrar servicios (ej. UDDI - obsoleto).

Pila de Protocolos Clásica (obsoleta en muchos casos)

- XML-RPC o SOAP para comunicación.
- WSDL para descripción del servicio.
- UDDI como registro de servicios.

Nota: Muchos de estos componentes han sido reemplazados por arquitecturas RESTful con JSON, OpenAPI (Swagger), y descubrimiento por medio de API Gateways.

Ciclo de Desarrollo de un Servicio Web

1. Diseño de interfaz (WSDL o OpenAPI).
2. Implementación del servicio (Java, Python, Node.js, etc.).
3. Despliegue (servidores como Apache, Nginx o contenedores Docker).
4. Pruebas y documentación (Postman, Swagger UI).

XML-RPC (obsoleto)

- Llamadas remotas usando XML sobre HTTP.
- Limitado en cuanto a tipos de datos y extensibilidad.
- Hoy en día está prácticamente en desuso.

SOAP - Simple Object Access Protocol

- Protocolo basado en XML para intercambio estructurado de datos.
- Aún usado en sistemas legados y empresas con fuerte enfoque en seguridad.
- Considerado pesado y complejo en comparación con REST/JSON.

WSDL - Web Services Description Language

- Lenguaje XML para describir servicios web SOAP.
- Automatiza la generación de clientes.
- Menos usado en arquitecturas REST. Reemplazado por OpenAPI.

UDDI - Universal Description, Discovery and Integration

- Estándar para registrar y descubrir servicios web.
- Nunca fue ampliamente adoptado en la práctica.
- Actualmente en desuso.

Tendencias Actuales

- **RESTful APIs** con JSON.
- **GraphQL** para consultas flexibles.
- **OpenAPI (Swagger)** para documentación y pruebas.
- **Microservicios**, contenedores (Docker) y orquestadores (Kubernetes).

Conclusiones

- La evolución de los servicios web refleja el cambio hacia arquitecturas ligeras y flexibles.
- Comprender tecnologías clásicas como SOAP/WSDL ayuda a entender los sistemas legados.
- El enfoque moderno prioriza REST, JSON y despliegue en la nube.

¡Gracias!

¿Qué es SOA?

- SOA (Service-Oriented Architecture) es un paradigma para organizar y utilizar capacidades distribuidas que pueden estar bajo el control de diferentes dominios.
- Promueve la reutilización de servicios, integración entre sistemas heterogéneos y agilidad empresarial.
- Basada en principios como interoperabilidad, reusabilidad y descubrimiento dinámico.

Evolución hacia SOA

- **Monolitos** $\rightarrow$ **Aplicaciones de tres capas** $\rightarrow$ **SOA** $\rightarrow$ **Microservicios**.
- SOA introdujo el desacoplamiento entre capas funcionales mediante contratos de servicios.
- Permite que diferentes aplicaciones (CRM, ERP, etc.) se comuniquen sin acoplamientos rígidos.

Multicapa vs. SOA

- Aplicaciones multicapa tradicionales:
  - Capa de presentación
  - Capa de lógica de negocio
  - Capa de datos
- En SOA, la lógica de negocio se divide en servicios interoperables.
- Las capas pueden residir en diferentes servidores o incluso en distintos proveedores.

Mensajería SOA

- Los servicios se comunican mediante mensajes.
- Protocolos comunes: SOAP, REST (más reciente), AMQP, JMS.
- Puede haber mensajería sincrónica (HTTP) o asincrónica (colas de mensajes).
- **Middleware** como Apache Kafka, RabbitMQ o ActiveMQ facilitan esta comunicación.

Enterprise Service Bus (ESB)

- ESB actúa como un "orquestador" que enruta, transforma y coordina servicios.
- Componentes comunes:
  - Adaptadores, ruteo, transformación de datos (XSLT), seguridad.
- Herramientas populares: MuleESB, Apache ServiceMix, WSO2.
- Actualmente muchas arquitecturas migran de ESB a microservicios con colas ligeras por la complejidad que puede introducir un ESB.

Pasos para implementar una aplicación SOA

1. Identificar servicios candidatos a partir de la lógica del negocio.
2. Diseñar contratos (interfaz WSDL o RESTful/OpenAPI).
3. Implementar servicios (por ejemplo, en Java, .NET, Node.js).
4. Utilizar un bus de servicios o mecanismo de descubrimiento.
5. Orquestar y monitorizar servicios.

SOA vs Microservicios

- **SOA:** Integración entre sistemas empresariales, mayor foco en orquestación.
- **Microservicios:** Enfocados en servicios pequeños, independientes, desplegables de forma autónoma.
- SOA se considera precursor de los microservicios. Muchas ideas aún son válidas.

Conclusiones del Tema 4

- SOA representa una evolución clave hacia sistemas distribuidos modulares y escalables.
- Aunque ha sido reemplazado en muchos contextos por microservicios, sigue siendo relevante en empresas con sistemas heredados o integración de grandes plataformas.
- La mensajería, contratos y orquestación siguen vigentes, aunque ahora bajo arquitecturas más ligeras.