UNIVERSIDAD AUTONOMA DE CHIAPAS

Facultad de Contaduría y administración Campus |

Alumno: Diego González Carpio

Materia: Taller de desarrollo 4

Actividad: Act.1.2 Arquitectura Orientada a Servicios

Investigación sobre Arquitectura Orientada a Servicios y Arquitectura de Microservicios

1. Introducción

La Arquitectura Orientada a Servicios (SOA, por sus siglas en inglés, Service-Oriented Architecture) y la Arquitectura de Microservicios son dos enfoques populares en el diseño y desarrollo de aplicaciones de software. En esta investigación, exploraremos ambas arquitecturas, sus características, beneficios, técnicas de integración de microservicios, balanceo de carga, despliegue, comparación con la arquitectura monolítica y buenas prácticas para diseñar arquitecturas de microservicios.

2. Arquitectura de Microservicios

2.1 Características

La Arquitectura de Microservicios se caracteriza por descomponer una aplicación en un conjunto de servicios pequeños e independientes. Cada servicio se enfoca en una única funcionalidad y puede ser desarrollado, desplegado y escalado de forma independiente. Algunas de las características clave son:

- Descomposición en servicios pequeños: Divide la aplicación en componentes independientes y manejables.
- Independencia tecnológica: Cada servicio puede utilizar diferentes tecnologías y lenguajes de programación.
- Despliegue independiente: Facilita la implementación continua y actualización de servicios sin afectar otros.
- Escalabilidad: Permite escalar servicios de manera individual para satisfacer la demanda.

2.2 Beneficios

Los beneficios de la Arquitectura de Microservicios incluyen:

- Agilidad en el desarrollo: Facilita el desarrollo rápido y la implementación continua.
- Escalabilidad: Permite escalar servicios según la demanda, optimizando recursos.
- Resistencia a fallos: Los fallos en un servicio no afectan a otros, lo que mejora la tolerancia a fallos.
- Mejora la colaboración: Equipos pequeños pueden trabajar en servicios de forma independiente.
- Adopción de tecnologías emergentes: Facilita la incorporación de nuevas tecnologías en servicios específicos.

3. Técnicas de Integración de Microservicios

La integración efectiva de microservicios es esencial para el éxito de la arquitectura. Algunas técnicas comunes incluyen:

- API REST: Comunicación a través de endpoints HTTP utilizando el estilo REST.
- Mensajería: Uso de sistemas de mensajería como RabbitMQ o Kafka para comunicación asíncrona.
- Gestión de eventos: Implementación de patrones de publicación-suscripción para notificar cambios.
- Protocolos de comunicación: Uso de protocolos como gRPC para comunicación eficiente.

4. Balanceo de Carga

El balanceo de carga es crucial para garantizar la disponibilidad y el rendimiento en arquitecturas de microservicios. Se utilizan técnicas como Round Robin, Algoritmos de Pesos, y el uso de servicios dedicados como Load Balancers para distribuir el tráfico de manera uniforme entre los microservicios.

5. Despliegue

El despliegue en una arquitectura de microservicios puede ser complejo debido a la cantidad de servicios involucrados. Se utilizan herramientas de orquestación de contenedores como Kubernetes para gestionar el despliegue, la escalabilidad y la administración de contenedores.

6. Arquitectura Monolítica vs. Arquitectura de Microservicios

- Arquitectura Monolítica: Una única aplicación donde todas las funcionalidades están integradas en un solo código base. Menos escalable y flexible, pero más fácil de desarrollar e implementar.

- Arquitectura de Microservicios: Aplicación dividida en servicios independientes. Más compleja pero altamente escalable y flexible.

7. Buenas Prácticas para Diseñar Arquitecturas de Microservicio

- Límites de contexto claro: Definir límites de servicio claros y responsabilidades.

- Comunicación eficiente: Minimizar la comunicación síncrona entre servicios.

- Resiliencia: Diseñar servicios para ser resistentes a fallos y gestionar la recuperación.

- Seguridad: Implementar medidas de seguridad en cada servicio y en la comunicación entre ellos.

- Monitorización y registro: Utilizar herramientas de monitorización para mantener visibilidad de la salud del sistema.