Materia:

DISEÑO ELECTRÓNICO BASADO EN SISTEMAS EMBEBIDOS

Alumno:

Posadas Pérez Isaac Sayeg

Paniagua Rico Juan Julian

García Azzúa Jorge Roberto

Grado y grupo:

8°G

Profesor:

Garcia Ruiz Alejandro Humberto

Unidad 4 -Tarea 6:

Diferencia entre arquitectura monolítica y de microservicios

ARQUITECTURA MONOLÍTICA VS ARQUITECTURA DE MICROSERVICIOS

Introducción

La forma en que se estructura una aplicación tiene un impacto directo en su escalabilidad, mantenimiento, despliegue y evolución. Dos de los enfoques más relevantes son la **arquitectura monolítica** y la **arquitectura de microservicios**. Mientras la primera ha sido el modelo tradicional durante décadas, la segunda ha emergido como una solución moderna adaptada a las exigencias de sistemas distribuidos y aplicaciones en la nube. Comprender sus diferencias, ventajas y desventajas es esencial para tomar decisiones informadas en el desarrollo de software.

Desarrollo

Arquitectura Monolítica

La **arquitectura monolítica** se basa en el desarrollo de una única aplicación unificada donde todas las funcionalidades están agrupadas y ejecutadas en un solo bloque de código. Esta aplicación comparte una única base de datos y se despliega como una sola unidad.

Características clave:

- Toda la lógica de negocio está dentro del mismo proceso.
- Generalmente posee una sola base de datos centralizada.
- Cambios en cualquier parte del sistema requieren un redeploy completo.
- La comunicación entre componentes es interna (dentro del mismo código fuente).

Ventajas:

- Desarrollo inicial más simple y rápido.

- Facilidad de pruebas en etapas tempranas.
- Despliegue centralizado.

Desventajas:

- Escalabilidad limitada a nivel de aplicación completa.
- Mayor complejidad al crecer: código acoplado y difícil de mantener.
- Riesgo de fallos globales: un error puede afectar toda la aplicación.
- Barreras para el trabajo simultáneo de varios equipos.

Arquitectura de Microservicios

La **arquitectura de microservicios** consiste en dividir la aplicación en múltiples servicios pequeños e independientes, donde cada uno cumple una función específica del negocio y puede ser desarrollado, desplegado y escalado de manera autónoma.

Características clave:

- Cada microservicio se comunica mediante interfaces (REST, gRPC, mensajería).
- Pueden estar escritos en distintos lenguajes o tecnologías.
- Suelen estar contenedorizados (por ejemplo, en Docker).
- Utilizan bases de datos independientes o especializadas por servicio (base de datos por microservicio).

Ventajas:

- Escalabilidad individual por servicio.

- Mejor mantenimiento: código más limpio y modular.
- Facilita el trabajo de múltiples equipos en paralelo.
- Mejor tolerancia a fallos.

Desventajas:

- Mayor complejidad operativa (infraestructura, orquestación).
- Necesidad de gestionar la comunicación y consistencia entre servicios.
- Requiere estrategias de monitoreo, logging y seguridad más sofisticadas.

Tabla Comparativa

Aspecto	Monolítica	Microservicios
Estructura	Una sola aplicación integrada	Conjunto de servicios independientes
Despliegue	Se despliega todo como una unidad	Cada servicio se despliega por separado
Escalabilidad	Escalado completo de la aplicación	Escalado individual por servicio
Base de datos	Compartida entre todos los módulos	Base de datos por servicio o distribuida
Dependencias	Alta interdependencia entre módulos	Bajo acoplamiento entre servicios
Fallo en un módulo	Puede afectar toda la aplicación	Afecta solo al servicio en cuestión

Tecnologías	Generalmente un único stack tecnológico	Puede usarse un stack distinto por servicio
Complejidad operativa	Menor (pero se incrementa al crecer la app)	Mayor (requiere orquestadores, monitoreo, seguridad distribuida)
Curva de aprendizaje	Más sencilla al inicio	Mayor al comienzo, pero más flexible a largo plazo

Ejemplo práctico

Imaginemos una aplicación de banca en línea con funciones como autenticación, gestión de cuentas, transferencias, historial de transacciones y notificaciones.

- **En una arquitectura monolítica**, todas estas funciones están integradas en una única aplicación. Si se necesita actualizar el módulo de transferencias, todo el sistema debe ser redeployado. Además, si hay un fallo en la autenticación, toda la aplicación podría volverse inoperativa.

- **En una arquitectura de microservicios**, cada función se implementa como un servicio independiente. El servicio de autenticación puede estar desacoplado del de transferencias, y cada uno puede escalar según su carga. Si el servicio de notificaciones falla, los demás servicios siguen funcionando normalmente.

Conclusión

La arquitectura monolítica sigue siendo útil en sistemas pequeños o cuando se busca simplicidad inicial. Sin embargo, a medida que una aplicación crece en tamaño y complejidad, la arquitectura de microservicios ofrece una solución más robusta, escalable y mantenible. Elegir entre una u otra arquitectura depende del tamaño del proyecto, los recursos disponibles, la experiencia del equipo y los objetivos de crecimiento del sistema. En muchos casos, una estrategia progresiva que parte de un monolito modular y evoluciona hacia microservicios puede ser una solución equilibrada.

Bibliografía

1. Newman, S. (2015). *Building Microservices*. O'Reilly Media.

2. Fowler, M. (2015). *Monolith First*. martinfowler.com.

3. Thönes, J. (2015). *Microservices*. IEEE Software, 32(1), 116–116.

4. Dragoni, N., et al. (2017). *Microservices: Yesterday, Today, and Tomorrow*. Springer.

5. Richards, M. (2015). *Software Architecture Patterns*. O'Reilly Media.