Telefónica

Arquitectura Software

Arquitecturas monolíticas y microservicios. Coreografía y orquestación

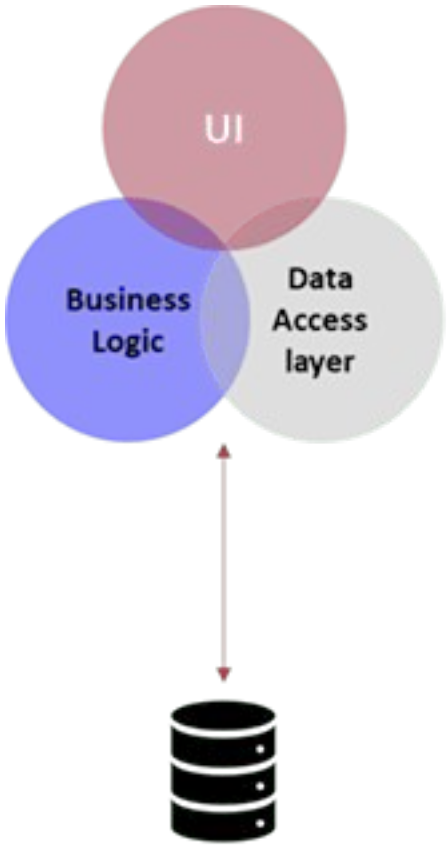
06/07/23 – Canal Online – Telefónica

Telefónica

Arquitecturas Monolíticas vs Microservicios

dd/mm/aa – Nombre de unidad

Arquitecturas Monolíticas



MONOLITHIC ARCHITECTURE

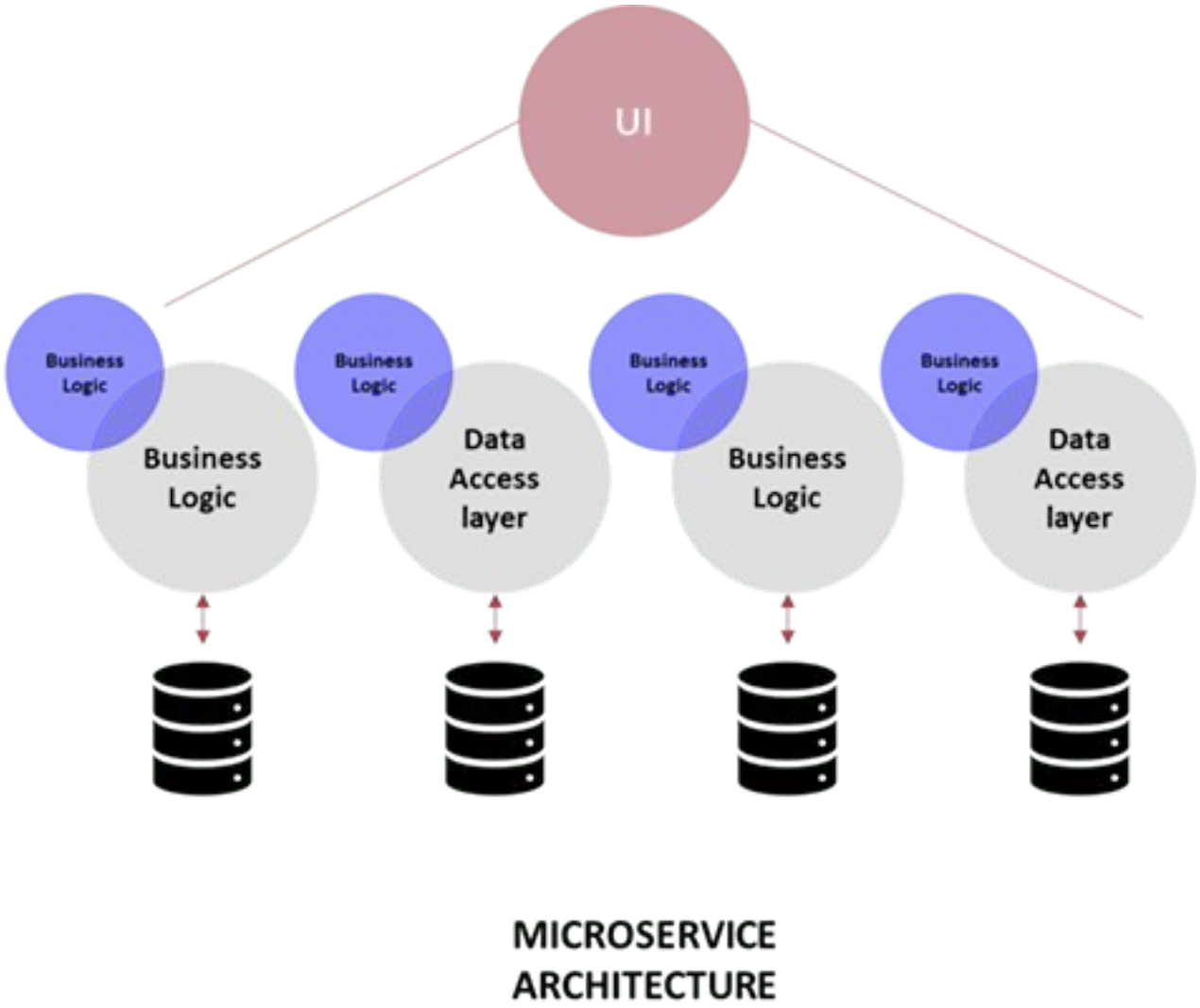
Pros:

1. **Simplicidad**: Más fáciles de desarrollar, probar y desplegar debido a que todo el código está en un solo lugar.
2. **Rendimiento**: Las aplicaciones monolíticas tienden a tener un rendimiento más rápido debido a que no hay comunicación a través de redes o llamadas a servicios externos.
3. **Fácil escalabilidad**: Al ser una aplicación única, se puede escalar verticalmente agregando más recursos a la máquina que la ejecuta.

Contras:

1. **Acoplamiento**: Los componentes de una aplicación monolítica están fuertemente acoplados, lo que dificulta realizar cambios o actualizaciones en partes específicas sin afectar el resto de la aplicación.
2. **Mantenimiento**: A medida que la aplicación crece, el mantenimiento puede volverse más complicado debido a la falta de separación entre los diferentes componentes.
3. **Escalabilidad limitada**: La escalabilidad horizontal, es decir, agregar más instancias de la aplicación, puede ser más difícil debido a las dependencias entre los componentes.

Arquitecturas Microservicios



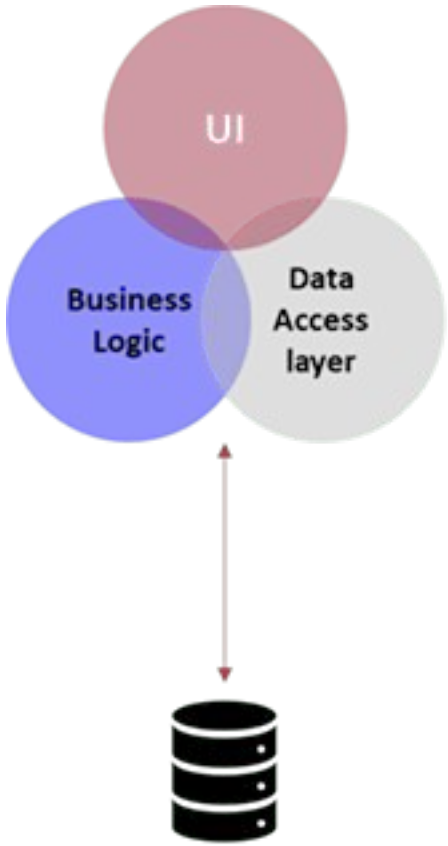
Pros:

1. **Escalabilidad flexible**: Cada microservicio se puede escalar de forma independiente según sea necesario, lo que permite un mejor aprovechamiento de los recursos y una mayor capacidad de respuesta ante cambios en la carga de trabajo.
2. **Desarrollo ágil**: Los equipos pueden trabajar de forma independiente en diferentes microservicios, lo que facilita la implementación continua y la adopción de metodologías ágiles.
3. **Mantenibilidad**: Al tener componentes más pequeños y específicos, el mantenimiento y la depuración se vuelven más manejables, y es más fácil realizar actualizaciones o mejoras en partes específicas de la aplicación.

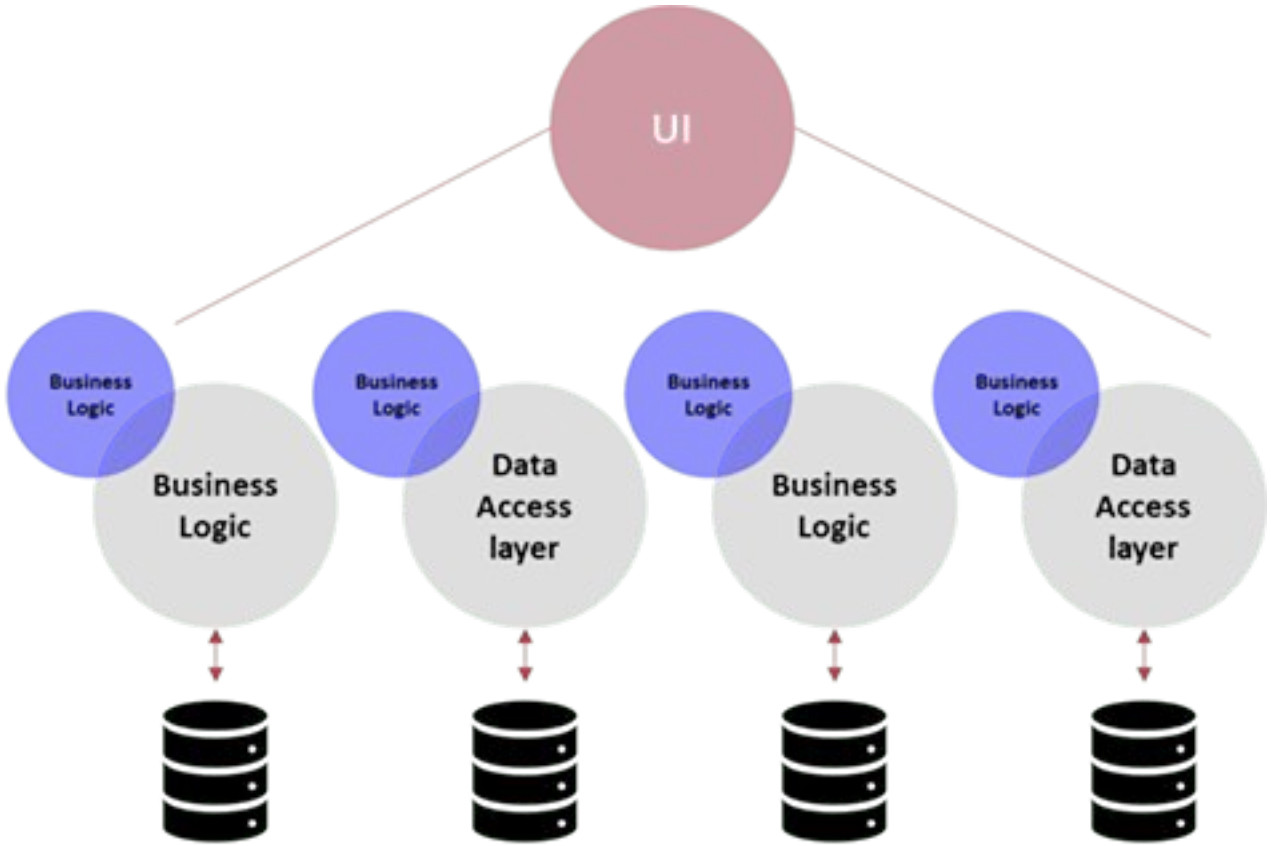
Contras:

1. **Complejidad en la gestión**: Al tener múltiples microservicios, se requiere una gestión y coordinación más sofisticada, lo que puede aumentar la complejidad operativa.
2. **Comunicación en red**: La comunicación entre microservicios se realiza a través de la red, lo que puede introducir latencia y puntos de falla adicionales.
3. **Pruebas más complejas**: Las pruebas de integración en una arquitectura de microservicios pueden ser más complicadas debido a la necesidad de simular o replicar el entorno distribuido.

Arquitecturas Monolíticas y Microservicios



MONOLITHIC ARCHITECTURE



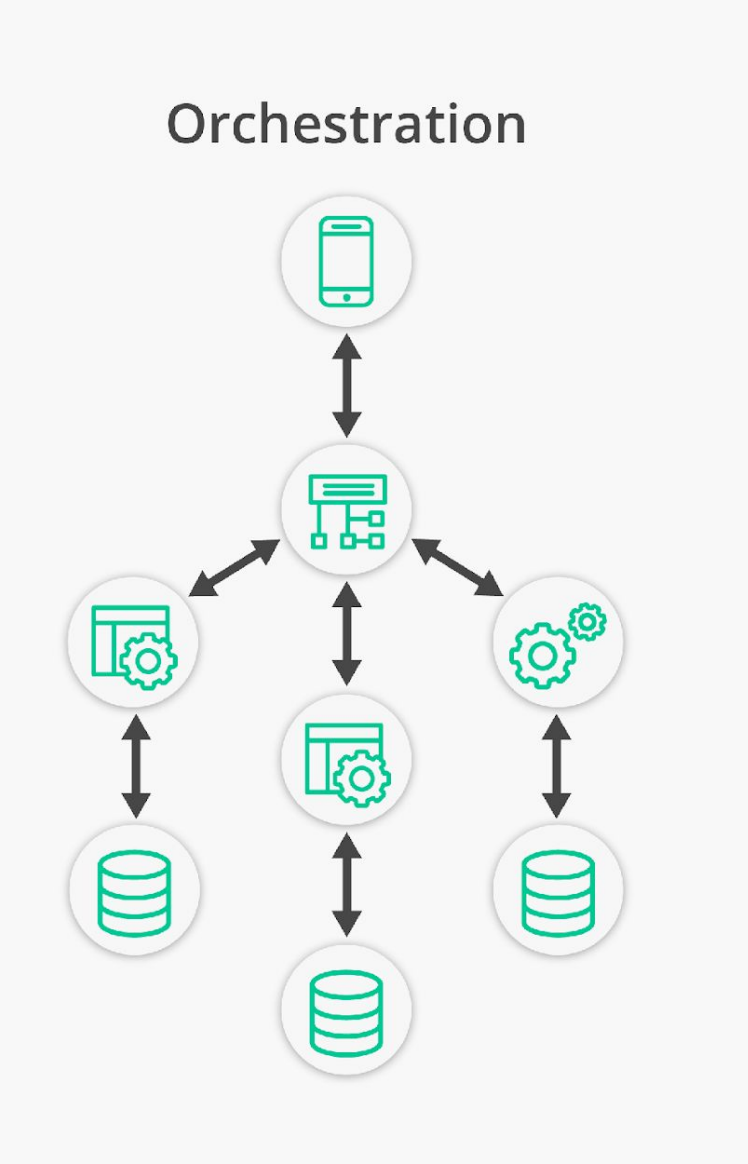
MICROSERVICE ARCHITECTURE

Telefónica

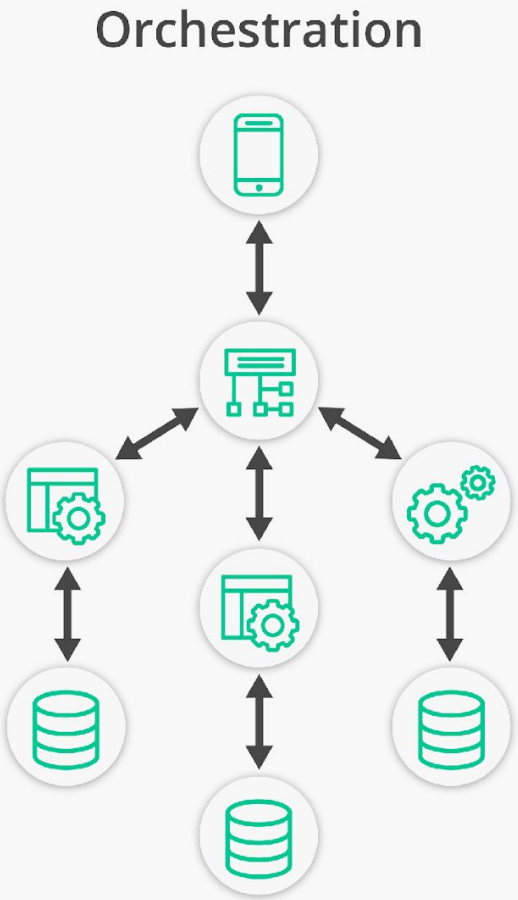
Orquestación vs Coreografía

dd/mm/aa – Nombre de unidad

Orquestación



Orquestación



Características

El **ORQUESTADOR** coordina y controla el flujo de trabajo de los diferentes componentes del sistema.

- **Centralización**: Hay un componente centralizado (orquestador) que coordina y dirige las actividades de los demás.
- **Control del flujo**: El orquestador controla el flujo de trabajo y toma decisiones sobre la ejecución de los pasos.
- **Acoplamiento más fuerte**: Los componentes dependen del orquestador y pueden estar diseñados para cumplir roles específicos en el proceso orquestado.

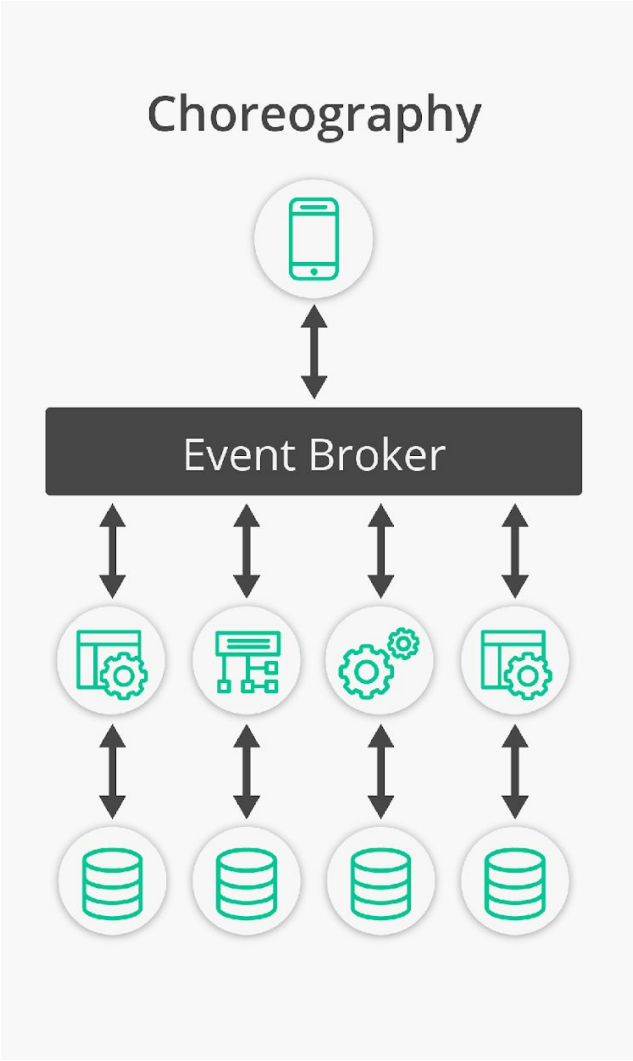
Beneficios:

1. **Mayor control y visibilidad**: Al tener un orquestador centralizado, es más fácil supervisar y controlar el flujo de trabajo.
2. **Mayor capacidad de administración**: El orquestador puede implementar lógica adicional, como la recuperación de errores o la gestión de transacciones.

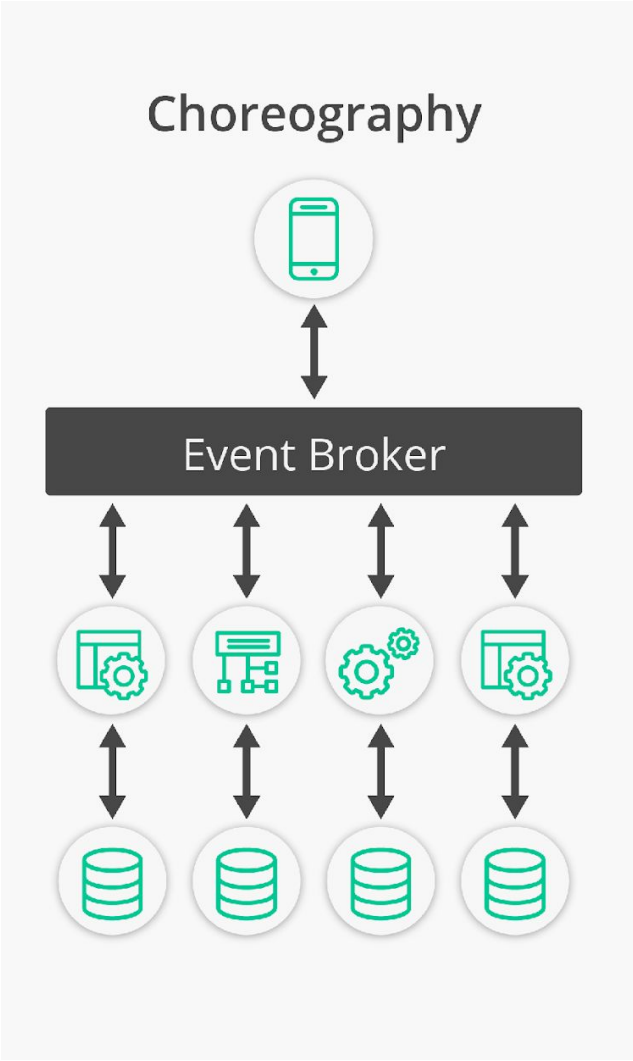
Desventajas:

1. **Acoplamiento más fuerte**: Los componentes dependen del orquestador, lo que puede generar una mayor dependencia y acoplamiento en el sistema.
2. **Menor autonomía de los componentes**: Los componentes no tienen un conocimiento completo del proceso general y dependen del orquestador para tomar decisiones.

Coreografía



Coreografía



Características

Cada componente es responsable de su propio comportamiento y colabora con otros componentes a través de mensajes o eventos. No hay un componente centralizado que controle el flujo de trabajo, sino que los componentes interactúan y toman decisiones basadas en los mensajes recibidos.

- **Descentralización**: No hay un componente centralizado que coordine y controle el flujo de trabajo.
- **Interacción basada en mensajes**: Los componentes se comunican a través de mensajes o eventos para coordinar su comportamiento.
- **Autonomía de los componentes**: Cada componente es responsable de su propia lógica y toma decisiones basadas en los mensajes recibidos.

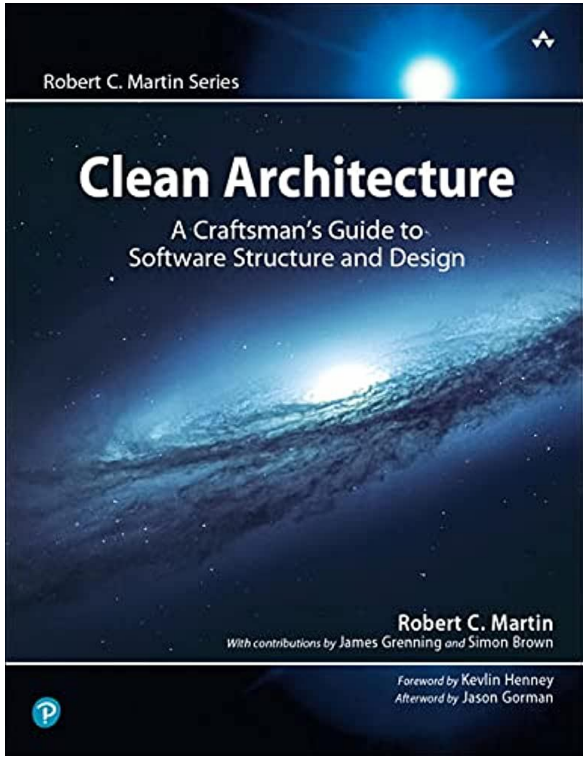
Pros

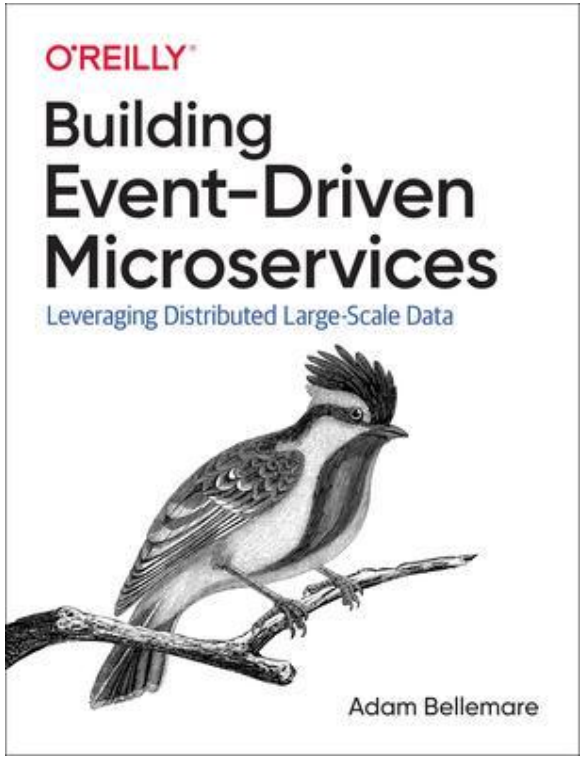
1. **Desacoplamiento**: Los componentes pueden evolucionar de manera independiente y no están fuertemente acoplados a un orquestador centralizado.
2. **Mayor escalabilidad**: La coreografía permite una mayor escalabilidad horizontal, ya que los componentes pueden ser distribuidos y escalados de manera independiente.

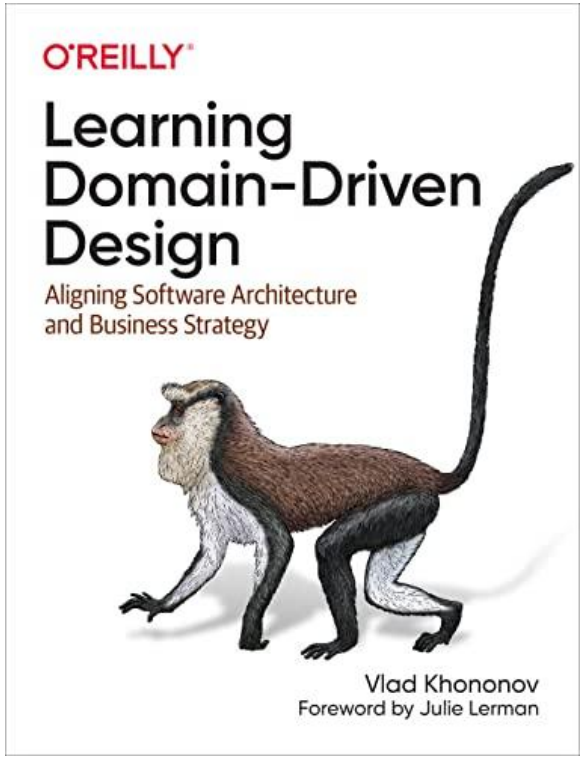
Contras:

1. **Mayor complejidad**: La coordinación y el seguimiento del flujo de trabajo pueden ser más complejos, ya que no hay un control centralizado.
2. **Menor visibilidad**: Al no tener un orquestador central, puede ser más difícil rastrear y supervisar el flujo de trabajo completo.

Técnicas de diseño





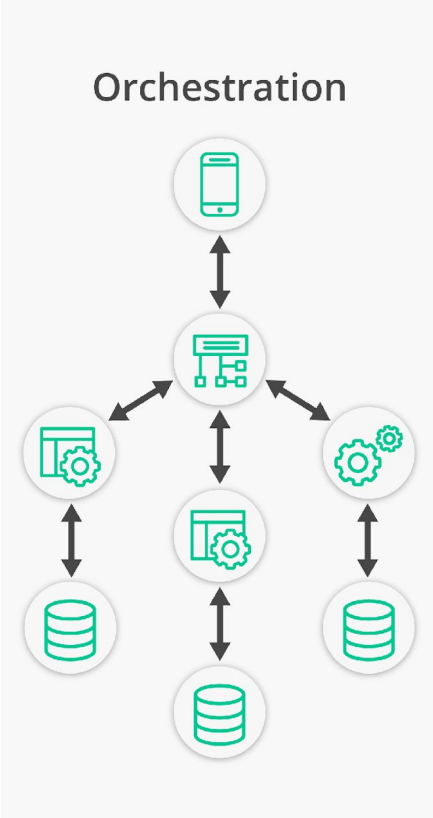


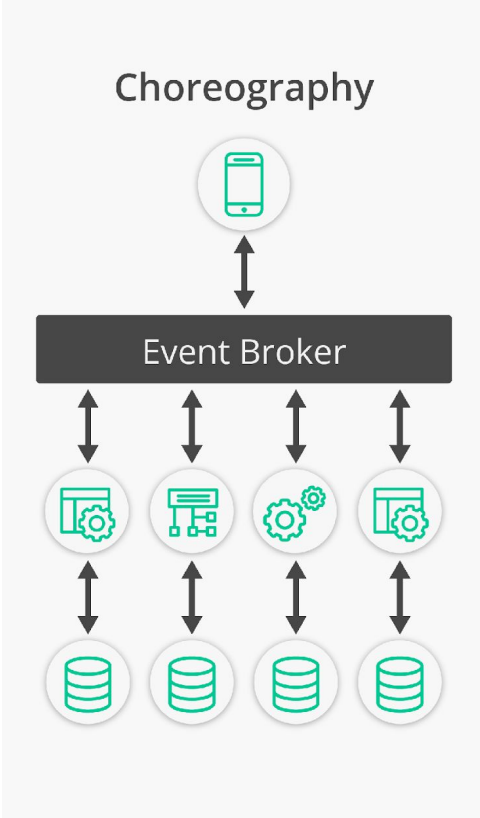
Telefónica

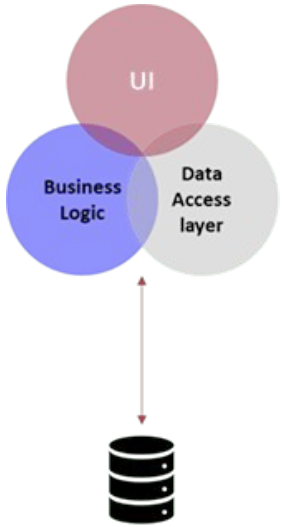
Resumen

dd/mm/aa – Nombre de unidad

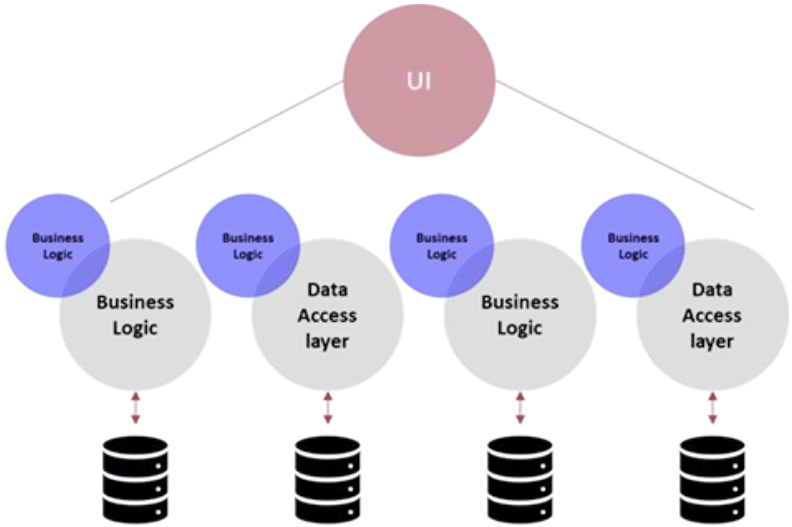
Resumen







MONOLITHIC ARCHITECTURE



MICROSERVICE ARCHITECTURE

Telefónica

Reto Tecnológico

dd/mm/aa – Nombre de unidad

Descripción

Telefónica quiere pilotar el uso de TELEGRAM como canal de comunicación con sus clientes.

El equipo de 1004 (*Call Center* de Telefónica) necesita una solución software que permita a los agentes (comerciales y de ayuda) usar un canal como TELEGRAM como vía de comunicación con todas las personas que quieran contratar productos y servicios de la marca Movistar.

También quieren usar este mismo canal para dar soporte y ayuda a los clientes que lo necesiten.

Reto Tecnológico

Detalle

Reto Tecnológico

Requisitos

Solución basada en microservicios

La solución presentada deberá contemplar los principales elementos de una arquitectura de microservicios (Deseable DDD).

Solución orientada a eventos

La solución presentada deberá contemplar los principales elementos de una arquitectura orientada a eventos.

Solución totalmente apificada

Todos los microservicios/piezas creadas deberán estar *apificadas* permitiendo futuras integraciones con otros sistemas

Infraestructura GCP

La solución deberá estar planteada para ser desplegada en el Cloud de Google. (Google Cloud Platform)

Frontal Web para Agentes

Debe tener un frontal web básico, la interfaz que usarían los agentes del Call Center de telefónica para comunicarse con los clientes a través de TELEGRAM

Recolección de datos en tiempo real

Debe recopilar datos de comportamiento para posterior análisis y reporting en tiempo real

Escalabilidad

La solución debe ser escalable y elástica

Reto Tecnológico

Entregables

Diseño de solución
Diagrama conceptual y/o diagramas de arquitectura/diseño que representen la solución software planteada

Software Backend
Código Back

Swaggers
Al menos los principales swaggers da las piezas principales (No es necesario que estén todos)

Frontal Web para Agentes
Código frontal

Reto Tecnológico

Valorable

Software funcional integrado correctamente con Telegram, GCP y Frontal Web

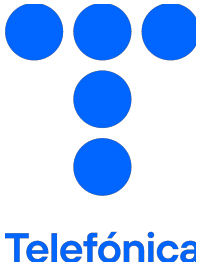
Software desplegado en GCP

Arquitectura escalable

Clean Code

Swaggers

Frontal Web para Agentes


Telefónica