Fernando Sonego

Sobre mi

Soy de Buenos Aires, Argentina. En la actualidad me desarrollo como Head of Solutions y Arquitecto de Soluciones en Andreani, una empresa enfocada a la logística.

Tengo una amplia experiencia en desarrollo de Arquitectura Distribuidas, software en la nube y tecnologías Microsoft.

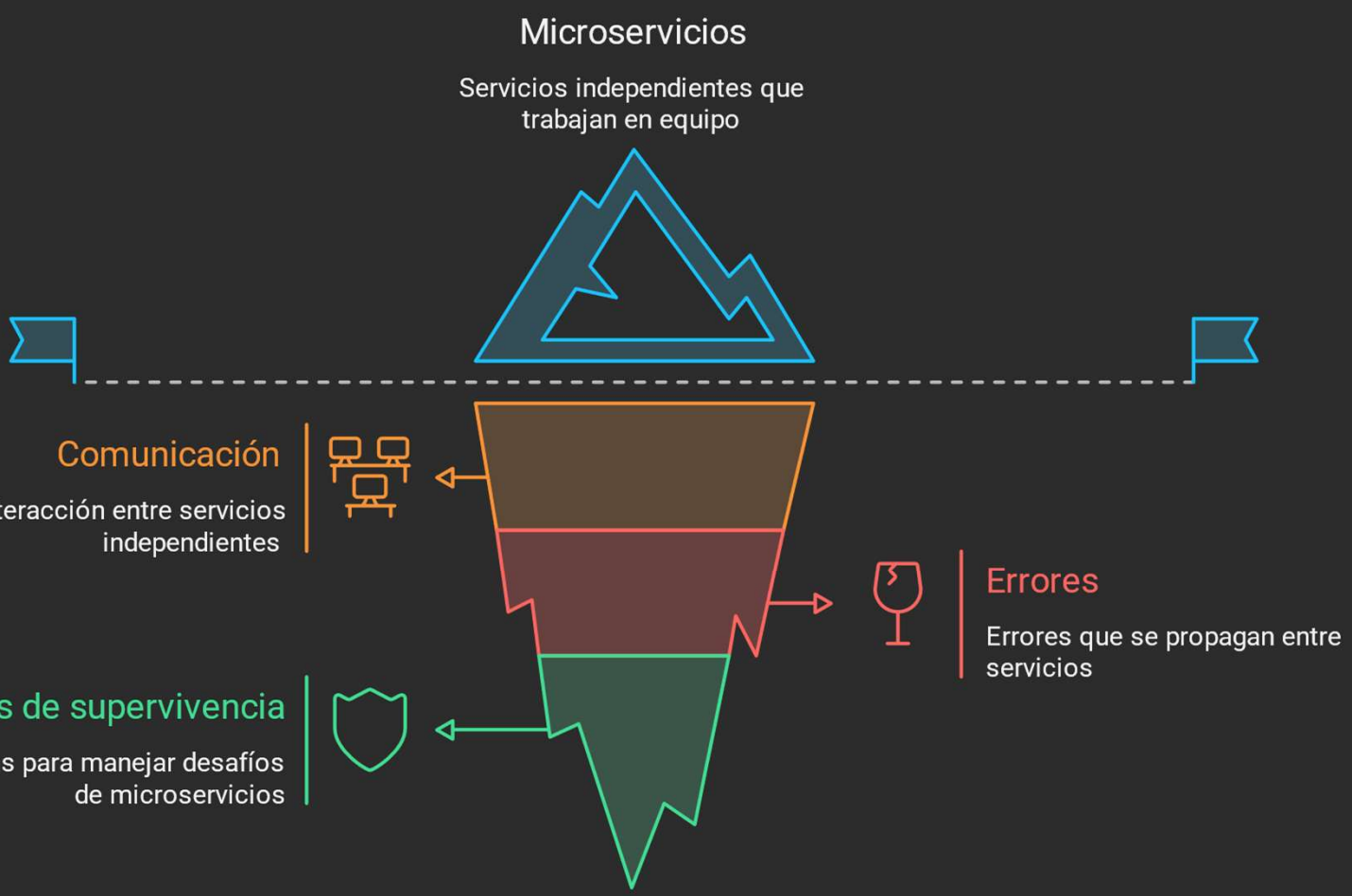
@fernandosonego

https://withoutdebugger.com/

MVP
Microsoft®
Most Valuable
Professional

¿Qué son los microservicios y por qué todo el mundo habla de ellos?

Microservicios: Más de lo que se ve a simple vista.



Comparando Monolitos y Microservicios

El monolito es cómodo al principio,
pero duele con el tiempo.

Los microservicios son más trabajo al principio… pero te salvan cuando tenés que crecer.

Problemas típicos de microservicios sin una estrategia clara

Frankenstein

Caso real

¿Qué se hizo?

Separación de Servicios

Se separaron los servicios: usuarios, productos, pagos, stock, notificaciones.

Comunicación Interna

Aún se llamaban entre sí como si vivieran en el mismo proceso.

Base de Datos Compartida

Todos usaban la misma base de datos compartida, un grave error.

Falta de Trazabilidad

No había trazabilidad ni se implementó un registro centralizado.

¿Qué paso?

Día no especificado

Cambio en el servicio de productos rompe los pagos

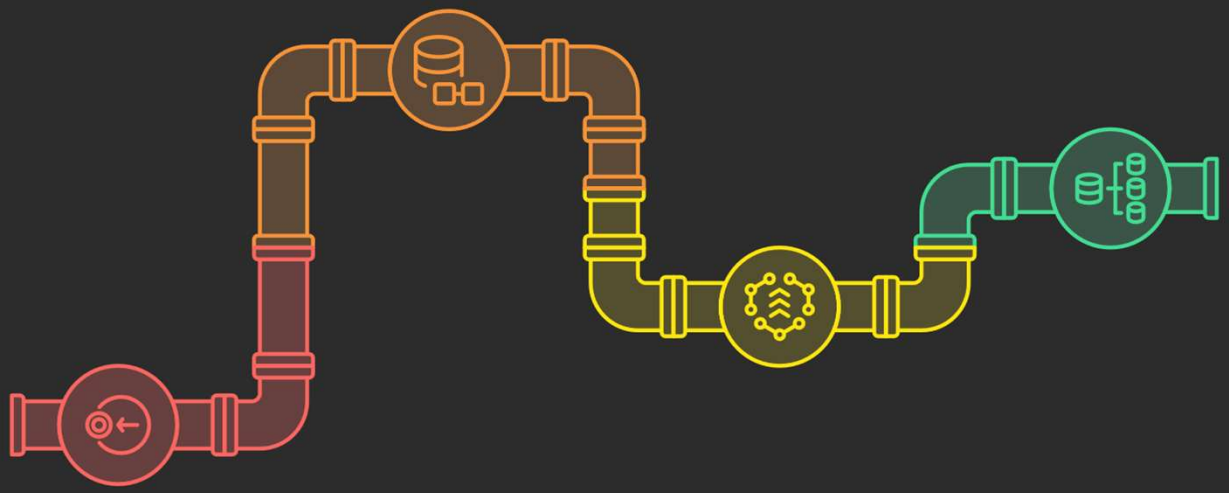
Dos días después

Los usuarios se quejan de errores aleatorios

Dos días después

Las revisiones de logs por separado no logran identificar el problema

¿Cómo lo resolvieron?



01

Revertir Cambios

Deshacer cambios recientes para estabilizar el sistema

02

Implementar Logging Distribuido

Establecer un sistema de registro centralizado usando OpenTelemetry

03

Agregar API Gateway

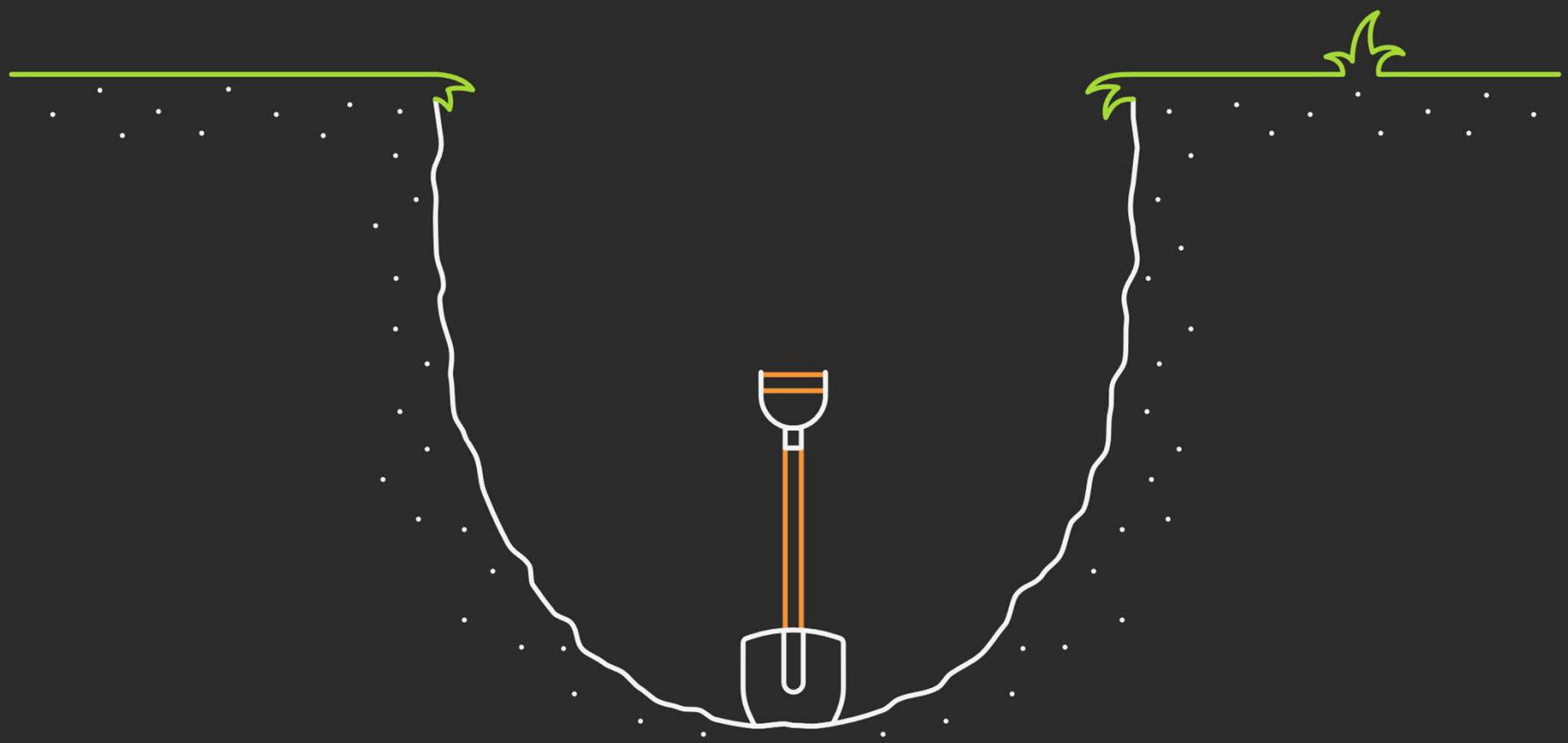
Introducir un punto de entrada único para todas las llamadas de servicio

04

Aplicar Database per Service

Aislar bases de datos para cada servicio para mejorar la autonomía

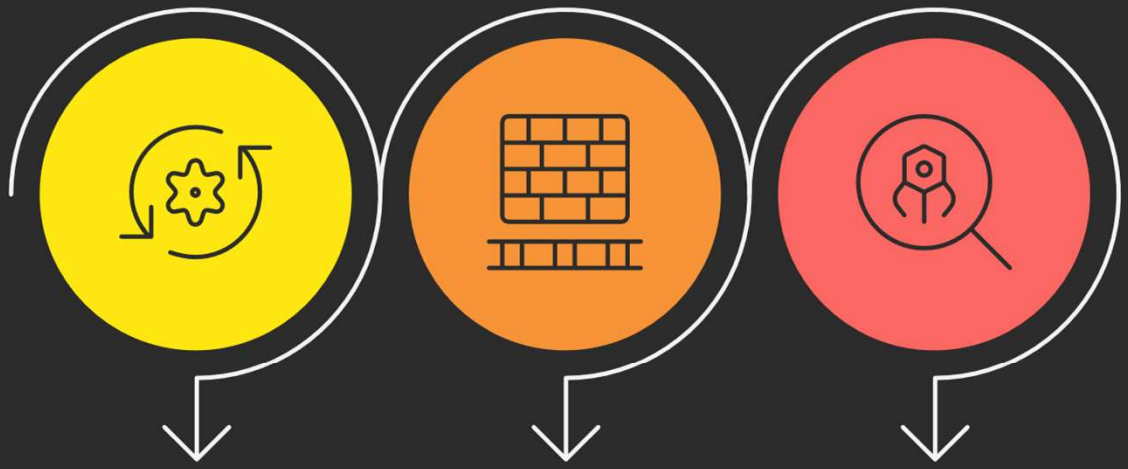
¿Por qué hablar de “supervivencia”?



Conceptos Base

Muy, pero muy Necesarios

Características Clave



Autónomo

Puede desarrollarse, desplegarse y escalarse de manera independiente.

Aislado

Tiene su propia base de datos o fuente de información, evitando el almacenamiento compartido.

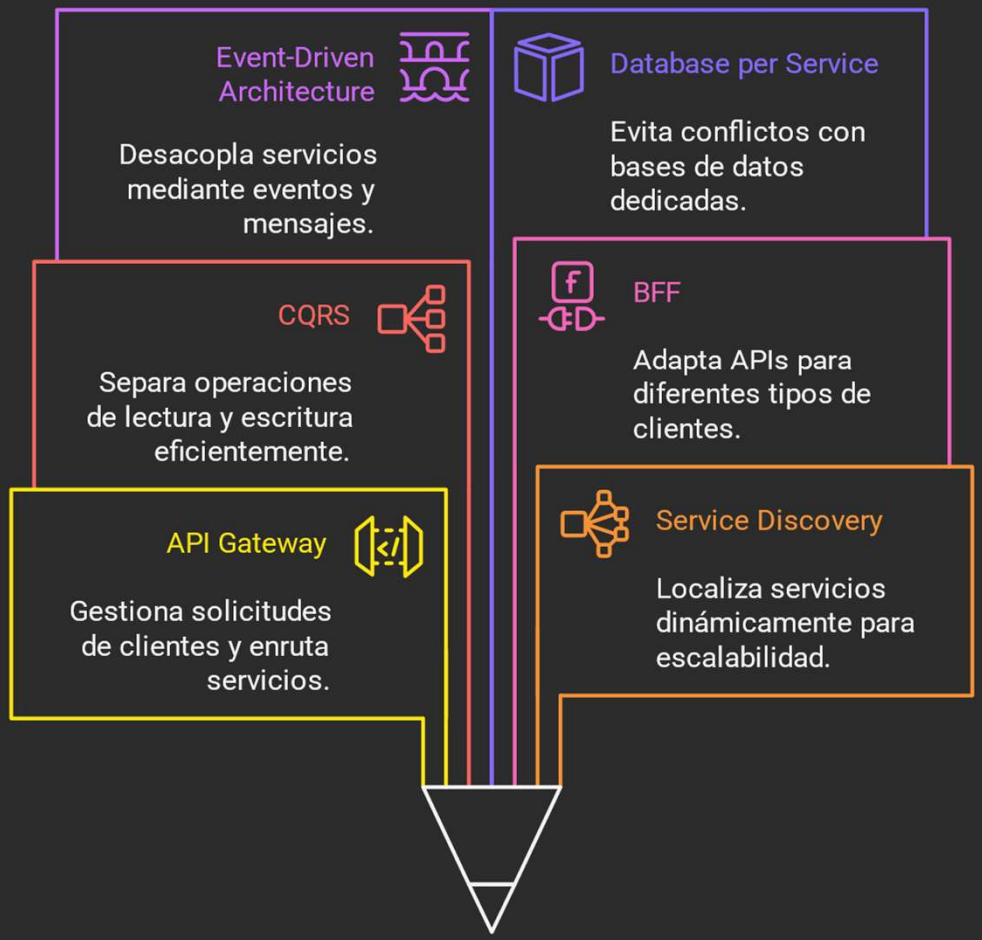
Específico a un dominio

Se encarga de una única responsabilidad clara, como usuarios o pagos.

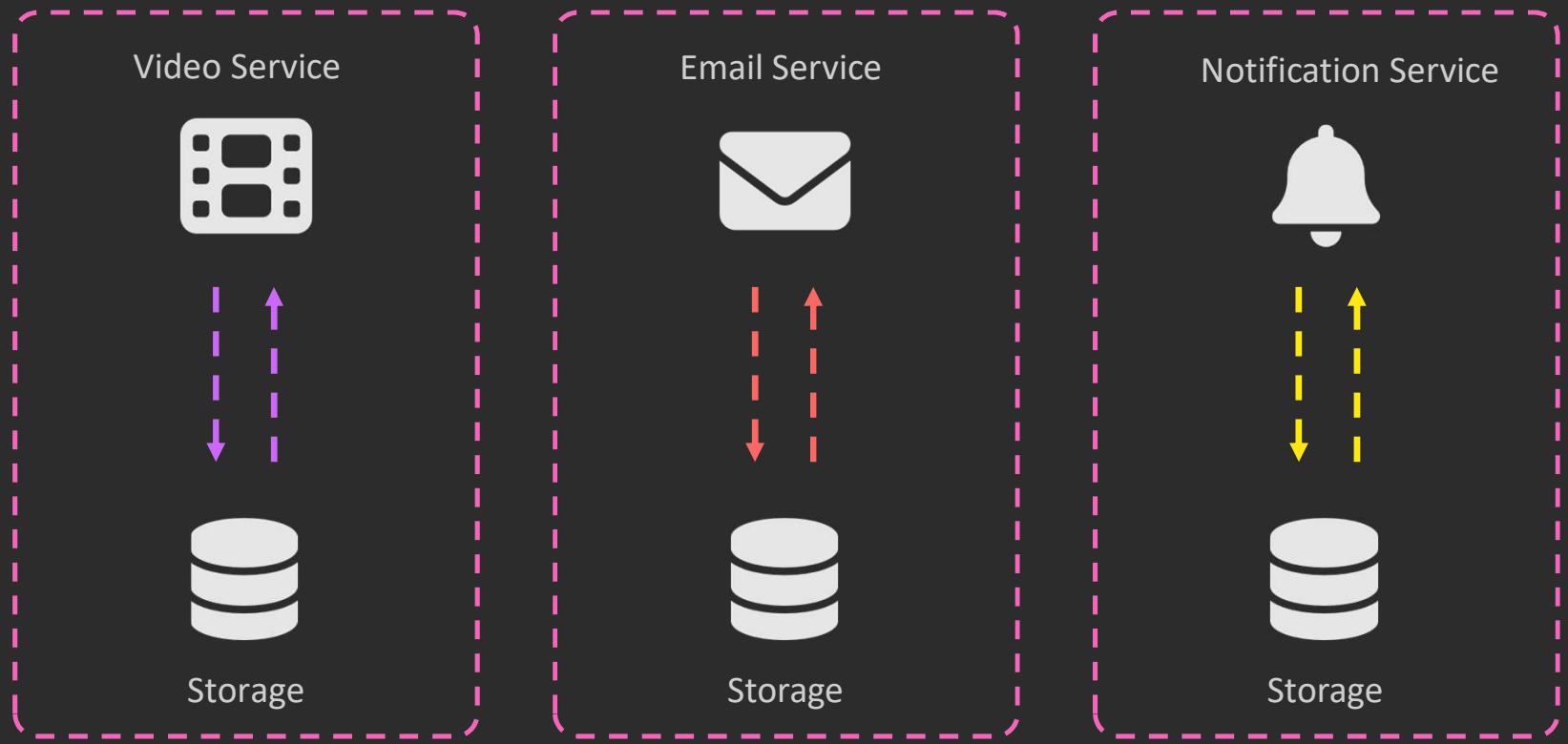
Patrones para microservicios

Esenciales

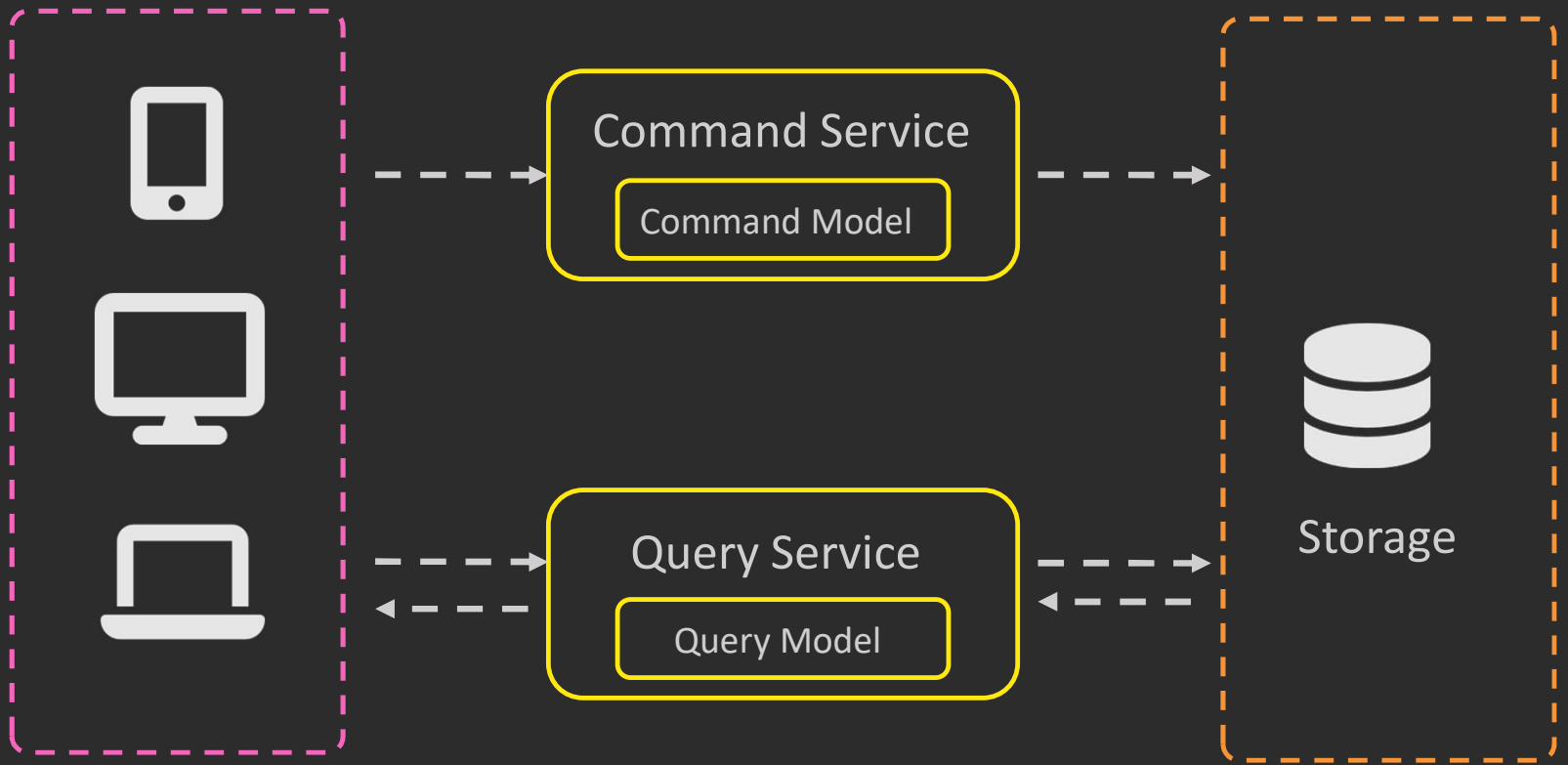
Construyendo Microservicios Escalables



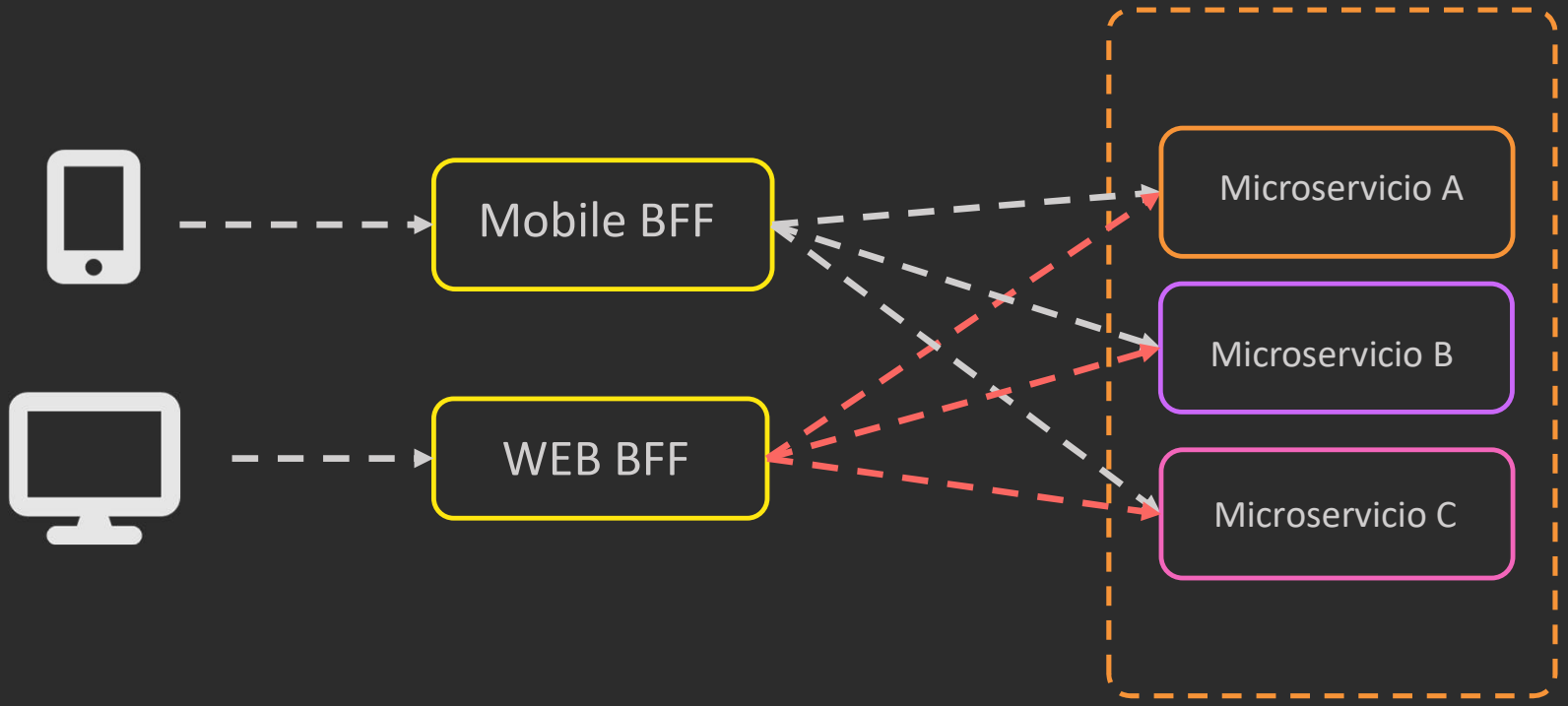
Database Service



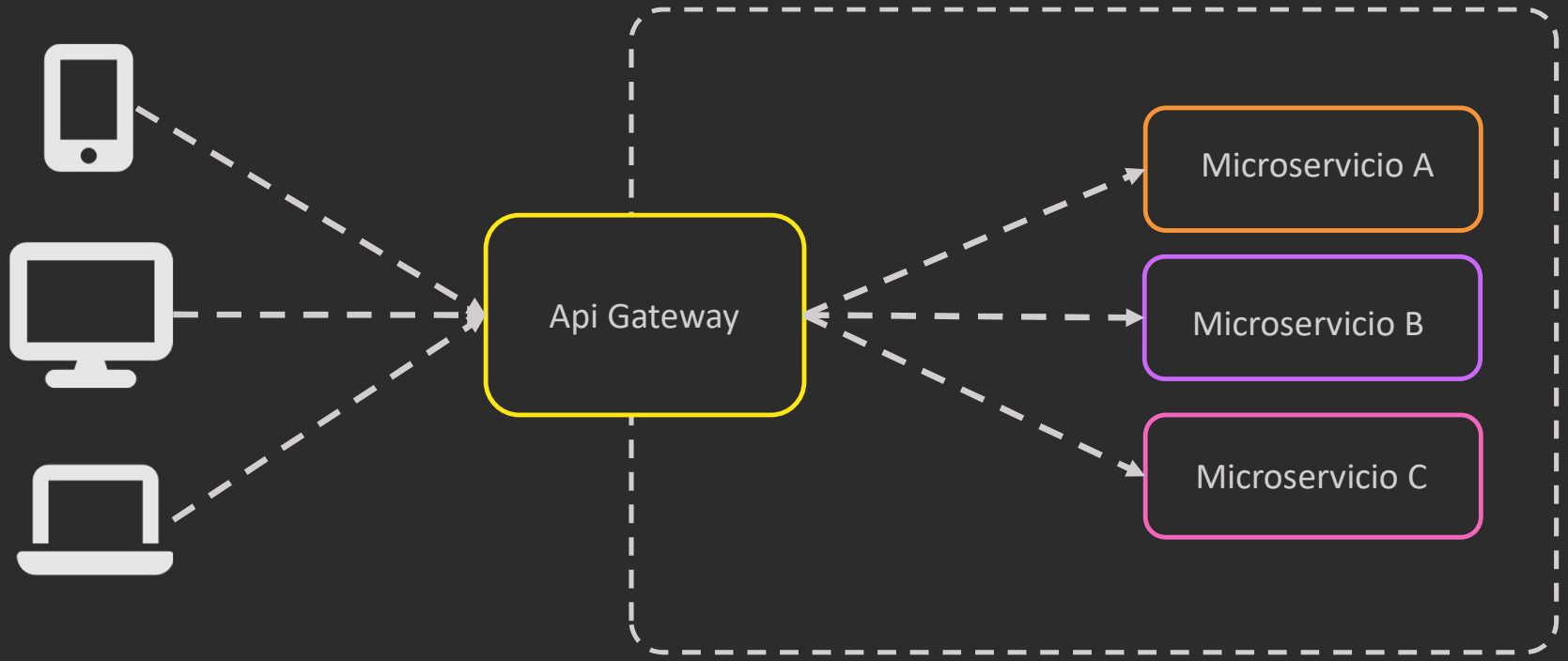
CQRS | Command Query Responsibility Segregation



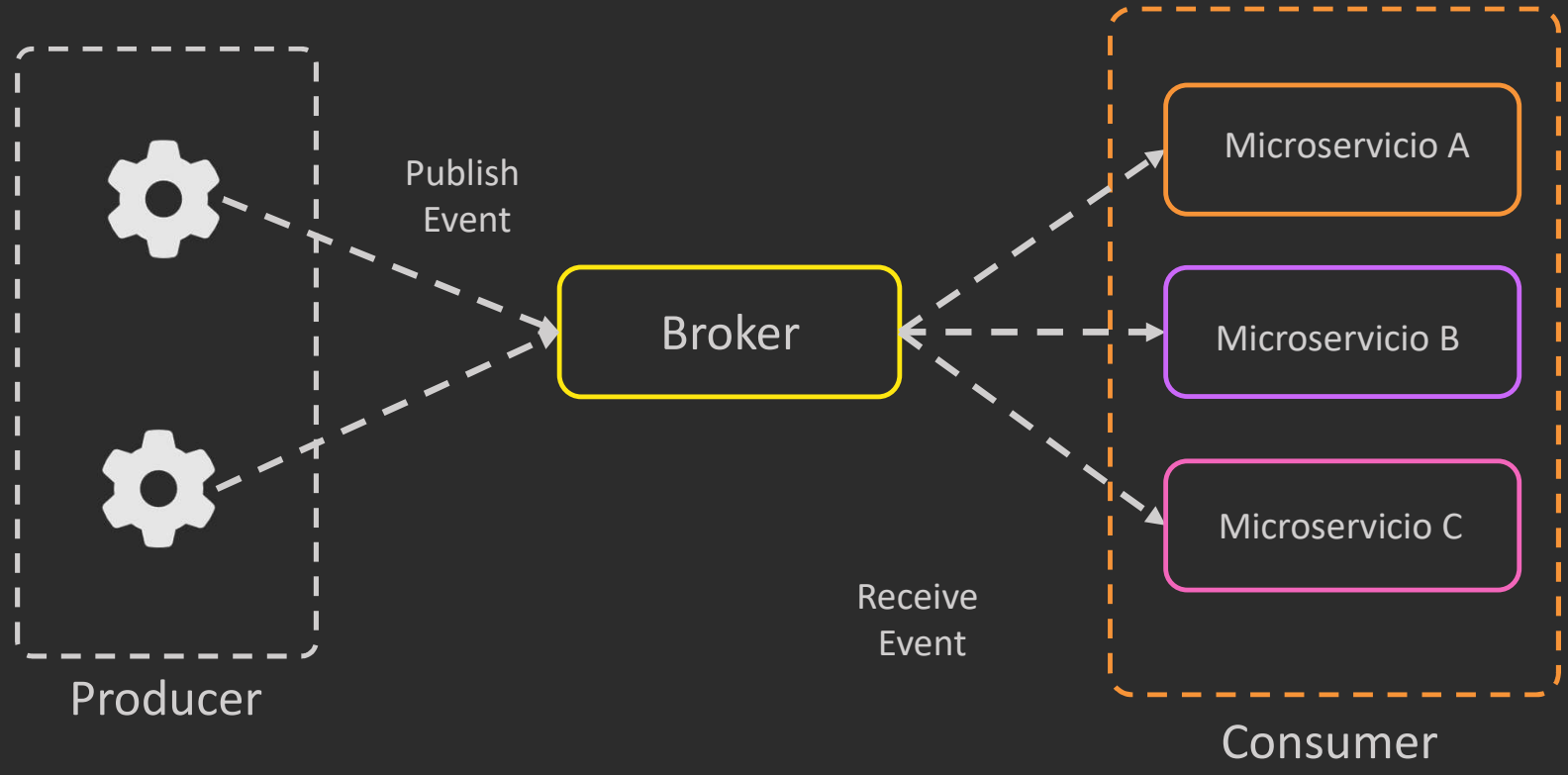
Banckends for Frontends (BFF)



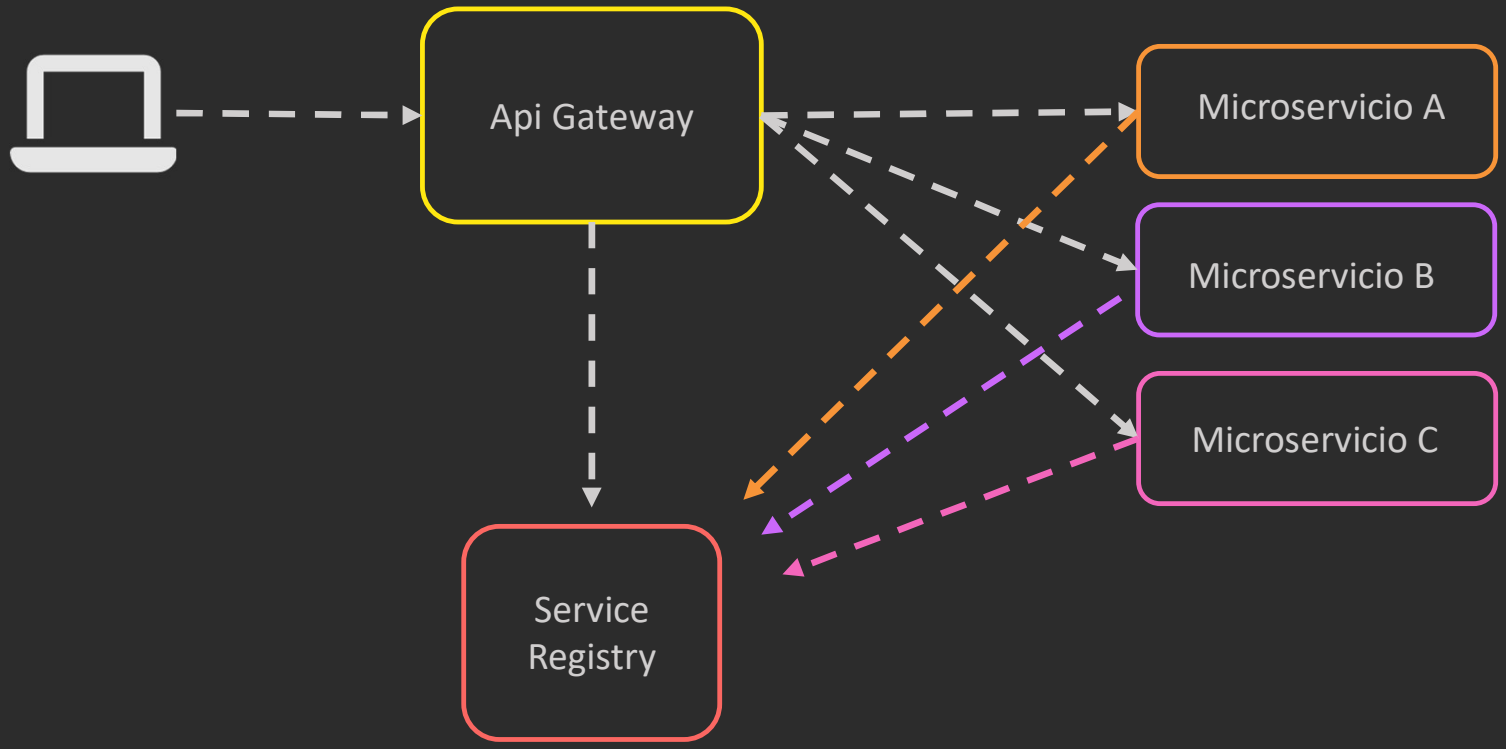
API Gateway


Api Gateway
Microservicio A
Microservicio B
Microservicio C

Event Driven



Service Discovery



Dos Casos reales

Buenas Practicas

Caso real #1: La base de datos compartida... del terror

Una fintech separó sus microservicios… pero todos seguían usando la misma base de datos.

Resultado:

- Un cambio de esquema en el servicio de pagos rompió el servicio de notificaciones.
- El servicio de soporte no podía actualizar nada sin coordinar con 3 equipos.
- En plena madrugada, un trigger mal configurado causó una cascada de errores.

¿Solución?
Implementaron Database per Service, empezaron a comunicar cambios por eventos, y cada equipo recuperó su autonomía.

Caso real #2: El evento que nadie escuchó

Un equipo implementó eventos para avisar cuando se creaban pedidos. Pero un bug en el servicio de notificaciones hacía que ignorara los mensajes si llegaban muy rápido.

Resultado:

- clientes no recibían confirmación, se duplicaban reclamos y soporte explotaba.

¿Solución?

Agregaron retry con backoff, monitoring, y una Dead Letter Queue para revisar mensajes fallidos.

Mejores Prácticas de Microservicios

Característica	Diseño y Límites	Observabilidad	Pruebas y Despliegue	Mantenimiento
 Definición de Alcance	Define responsabilidades del servicio	No aplicable	No aplicable	No aplicable
 Compartición de Datos	Comparte datos a través de eventos	No aplicable	No aplicable	Documenta contratos de servicio
 Registro	No aplicable	Implementa registro distribuido	No aplicable	No aplicable
 Seguimiento de Solicitudes	No aplicable	Usa IDs de correlación	No aplicable	Evita fallas en cascada
 Pruebas	No aplicable	No aplicable	Automatiza pruebas por servicio	No aplicable
 Despliegue	No aplicable	No aplicable	Usa despliegues independientes	No aplicable

No es magia,
es estrategia

¡Gracias!

@fernandosonego

https://**withoutdebugger**.com/