códigofacilito

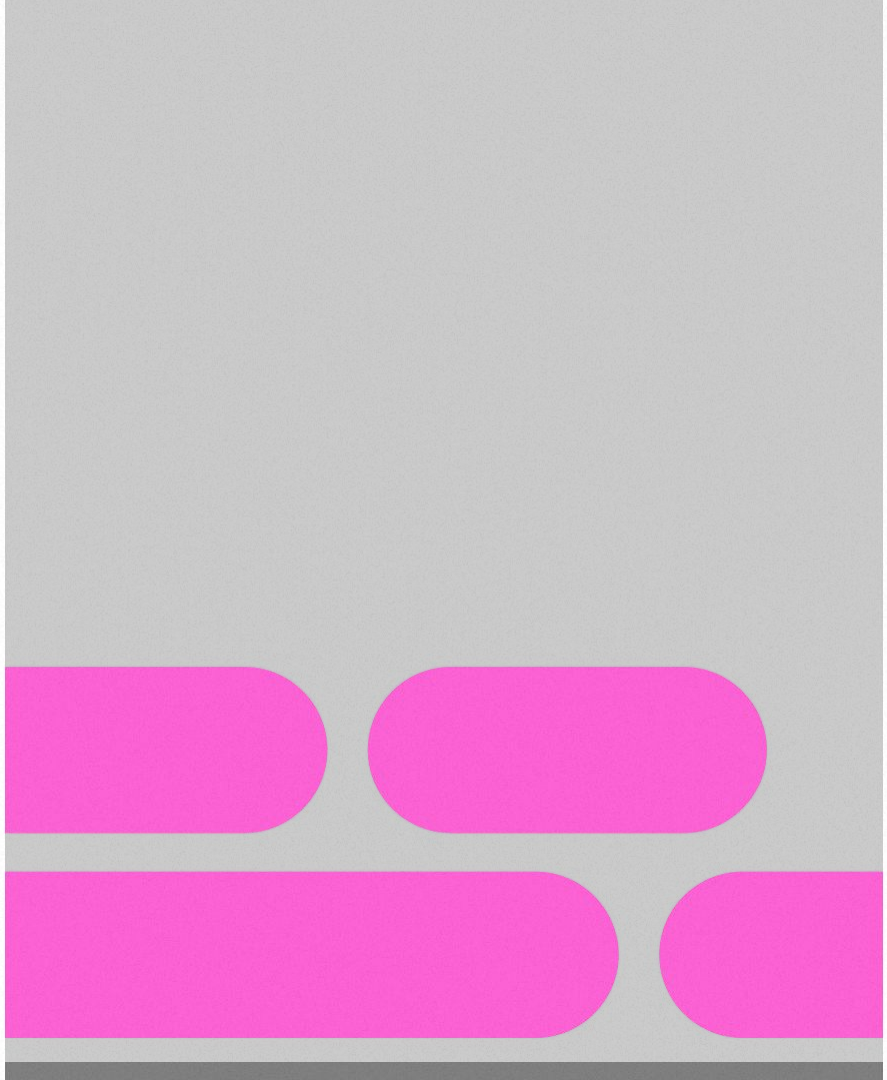
CQRS para Microservicios

Bootcamp - Backend Avanzado

Eduardo I. García Pérez.

Objetivos.

- ¿Qué es CQRS?
- Implementación.
- Ejercicio Práctico.



CQRS

Command query responsibility segregation

Es un patrón usado en la arquitectura para separar operaciones de lectura y escritura de nuestra base de datos.

Tradicionalmente usamos una solo DB para la lectura y escritura de datos. Lo que puede ser algo ineficiente.

Componentes (CQRS)

- Command: Operaciones que modifican estados (tablas)(Insert, Update & Delete).

- Query: Operaciones de obtención de datos.

Aplicamos el principio de responsabilidad única.

- Una tabla/modelo/database se encarga de realizar una y solo una acción.

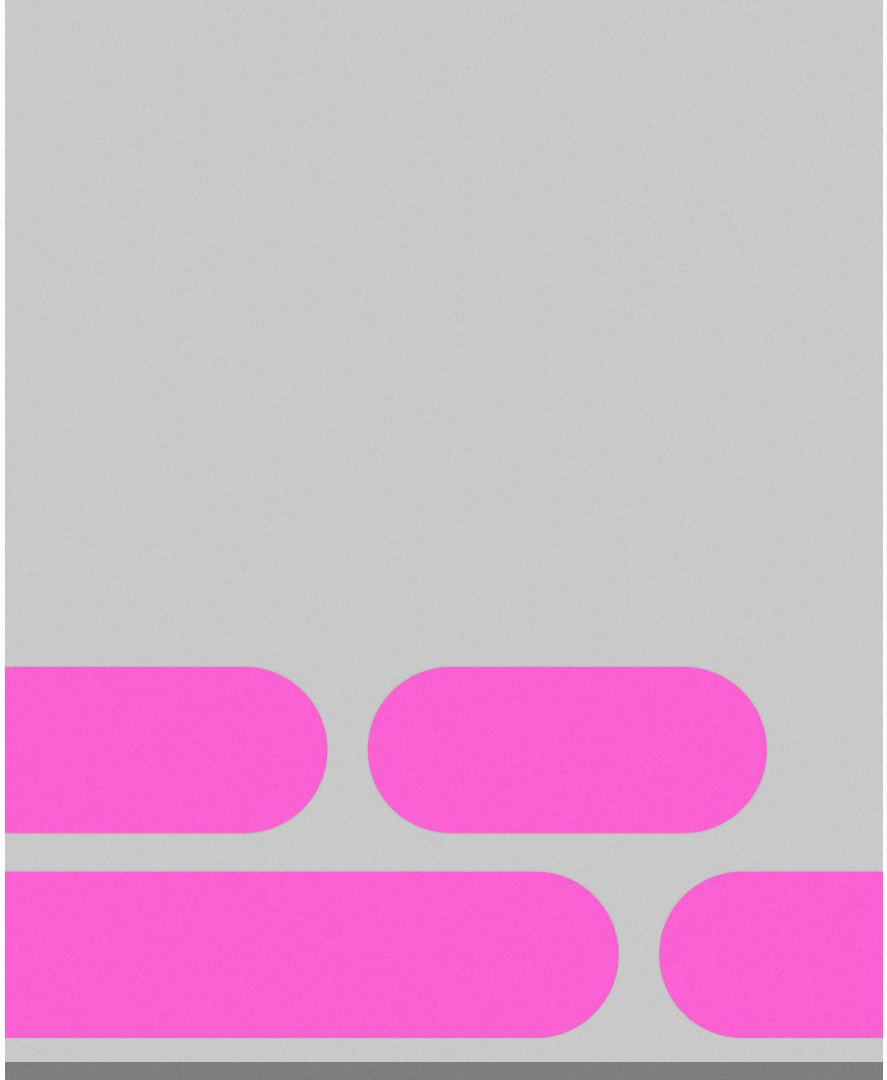
Pros. ✅

- Al separar los modelos es posible mejorar el performance de consultas y/o actualizaciones.
- Sin bloqueo de tablas.
- Fácil escalabilidad.
- Mejora de seguridad.*

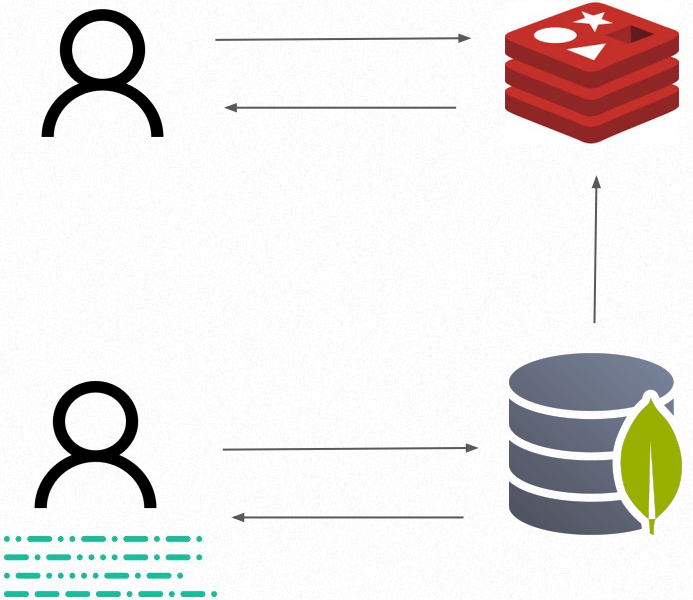
Cons. ❌

- Complejidad al mantener sync los datos.
- Complejidad operacional.
- Huecos de seguridad.*

Implementación.



Diferentes bases de datos.



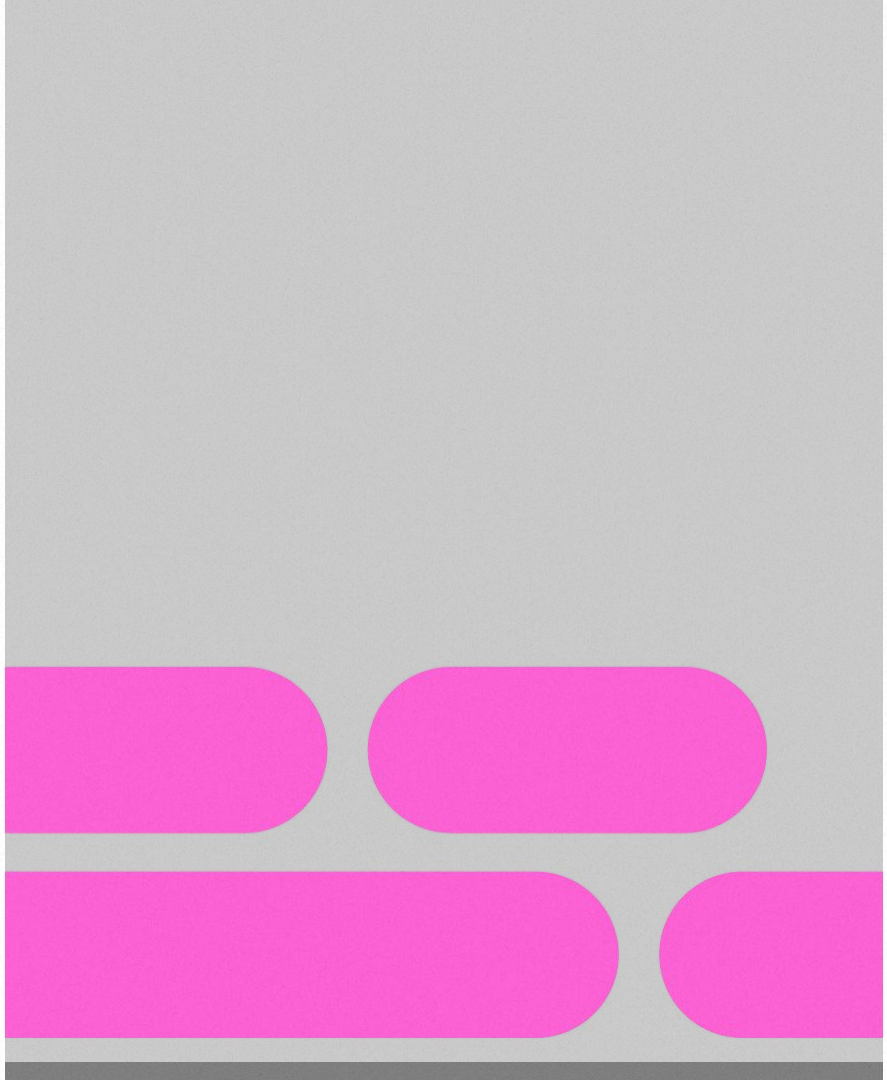
- pub-sub
- Firebase

- A

- ***API Call (Rest)***
- ***Event Drive (Pika, gRPC, KAFKA)***
- Stream in MongoDB
- Replication in MySQL
- JOBs

- B

Ejemplo práctico.



Conexión entre microservicios.

- Servicio Principal
- Servicio de Cursos.
- Servicio de mensajería.
- Servicio de lectura.

https://github.com/codigofacilito/api_call

Uso de Signals en Django

```python
from django.dispatch import receiver
from django.db.models.signals import post_save


@receiver(post_save, sender=Model)
def after_create(sender, instance, created, **kwargs):
    pass
```

códigofacilito

CQRS para Microservicios

Bootcamp - Backend Avanzado

Eduardo I. García Pérez.