CADS - Cloud Apps Delivery System:

Transacciones aisladas, sagas, eventos, y sus consumidores

Autor: Miguel García Sanguino
Tutor: Micael Gallego

CloudApps
MASTER

Universidad
Rey Juan Carlos

presentacion miquel

Introducción

CADS - Cloud Apps Delivery System

Se buscando un objetivo en el TFM, la independia de los actores en estructuras de microservicios cuando hay transacciones. Y a su vez podemos hablar de independencia de los servicios entre si y al mismo tiempo con el front end.

Objetivos: Independencia intra serivicios y con frontend

Middleware

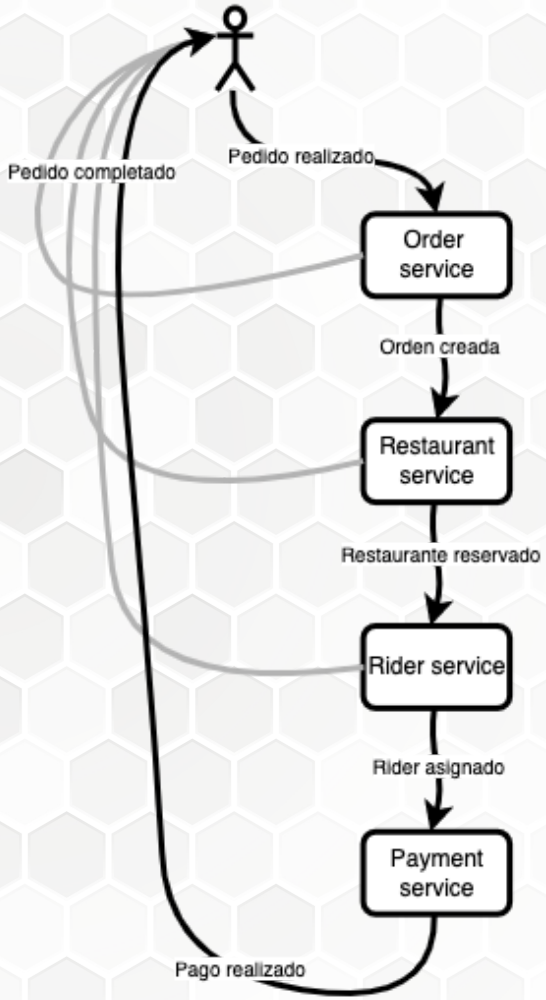
- sagas
- coreografia
- sin maquina de estados
- enfocado a eventos
- servicios independientes
- aislado

Frontend

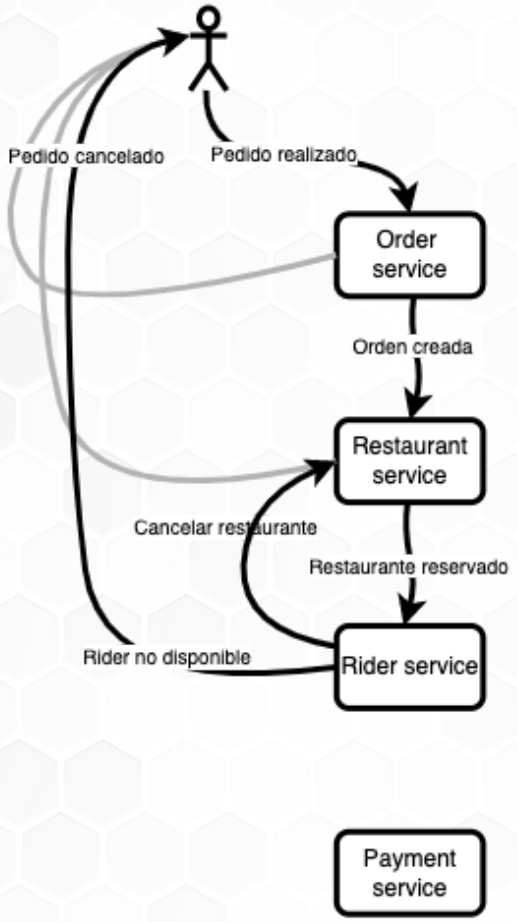
- independiente
- asincrono
- backend for frontend
- noficaciones online / offline

CADS - Cloud Apps Delivery System

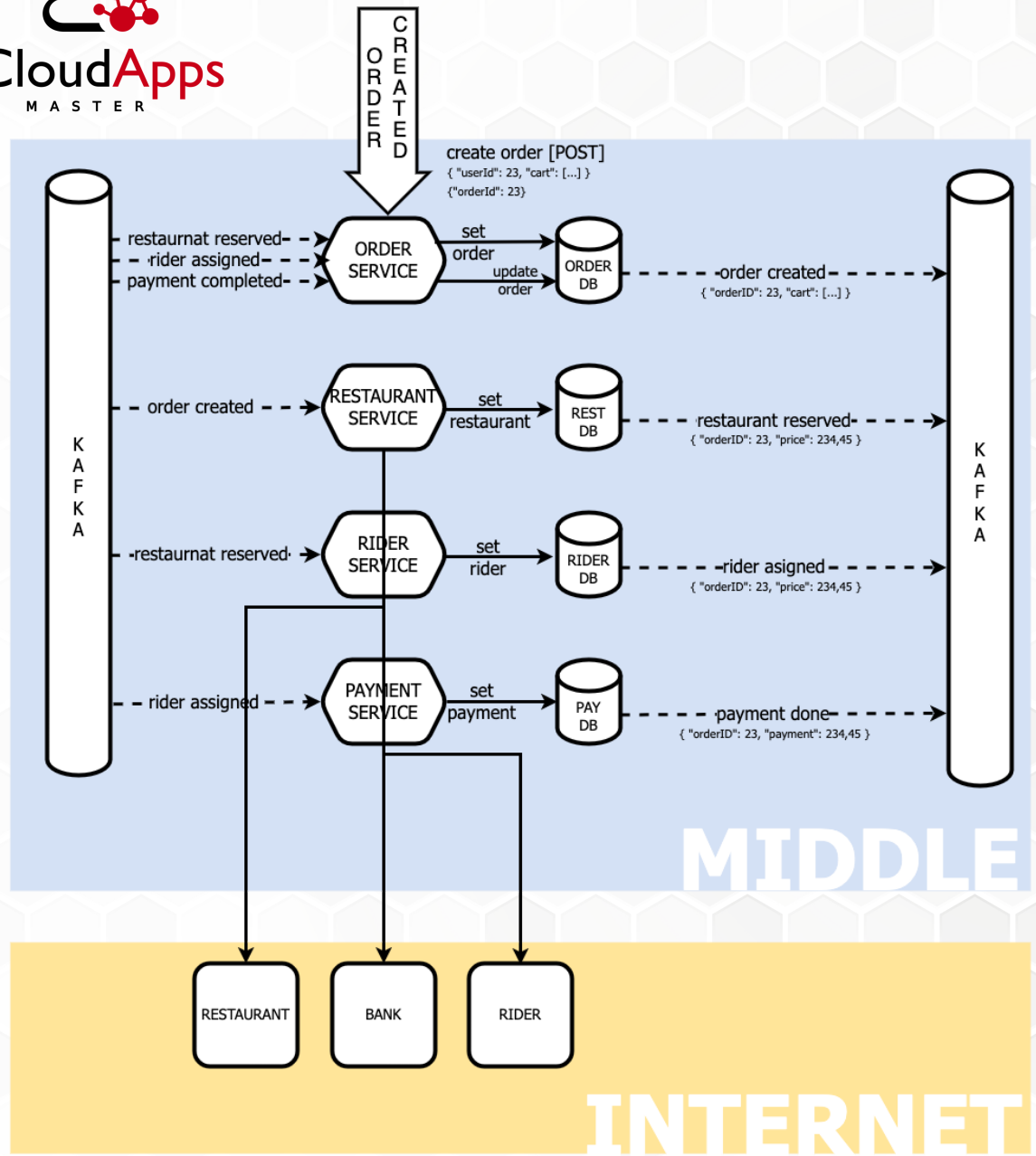
- Caso: pedido de comida online
- Reserva de restaurante
- Asignar un rider
- Realizar pago
- Rollback en caso de fallo
- Informar al usuario en cada paso



Completada



Cancelada

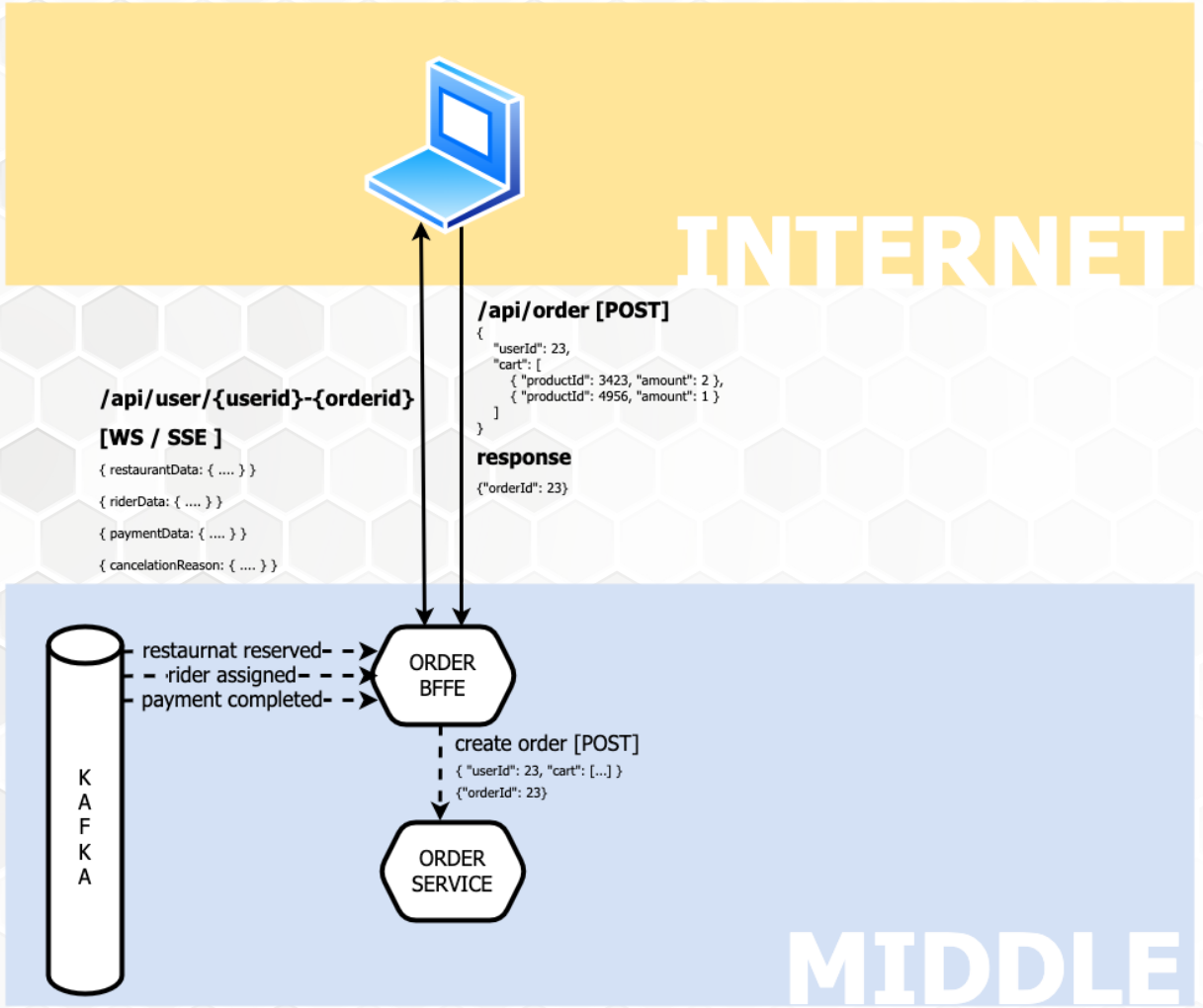


middleware

- modular
- servicios independientes
- basada en eventos
- coreografia
- sin maquina de estados
- servicios idempotentes
- resiliencia
- escalabilidad
- independiente del consumidor

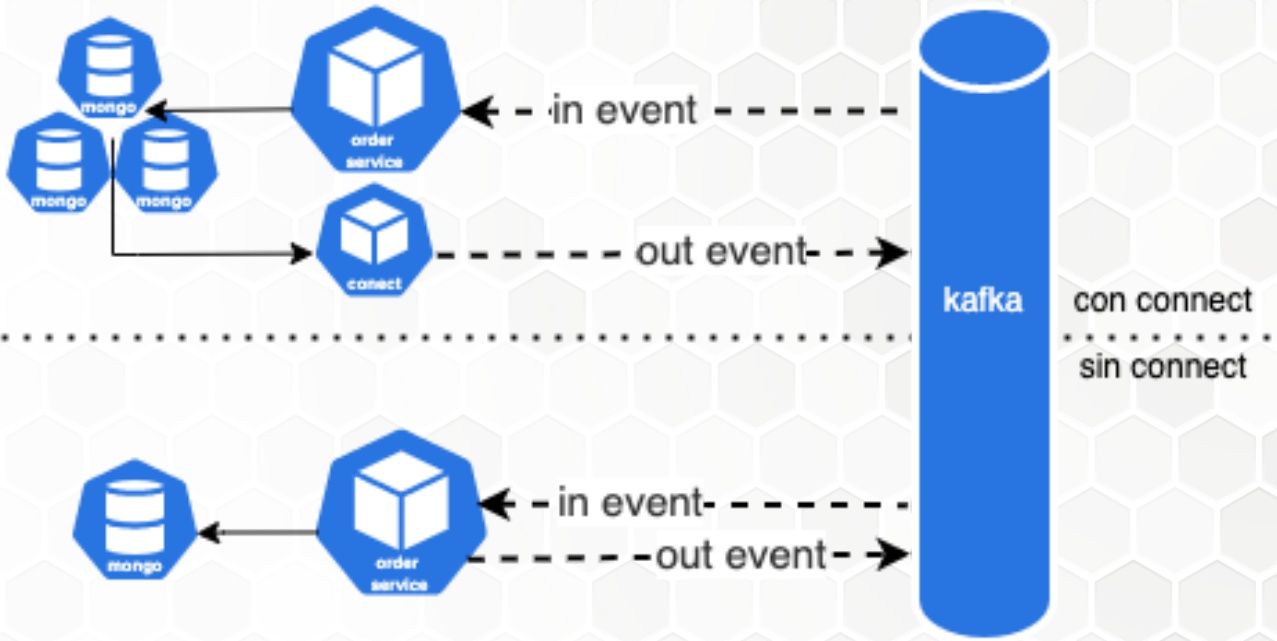
Frontend

- backend for frontend
- reactivo
- comunicaciones middle->front
- resiliencia
- escalabilidad
- independiente del consumidor



Stack middleware

- Kubernetes
- Kafka
- BBDD mongo
- Nodejs, kafkajs, express, mongoose
- Todas las comunicaciones por eventos
- mongo connect



Kafka Mongo connect

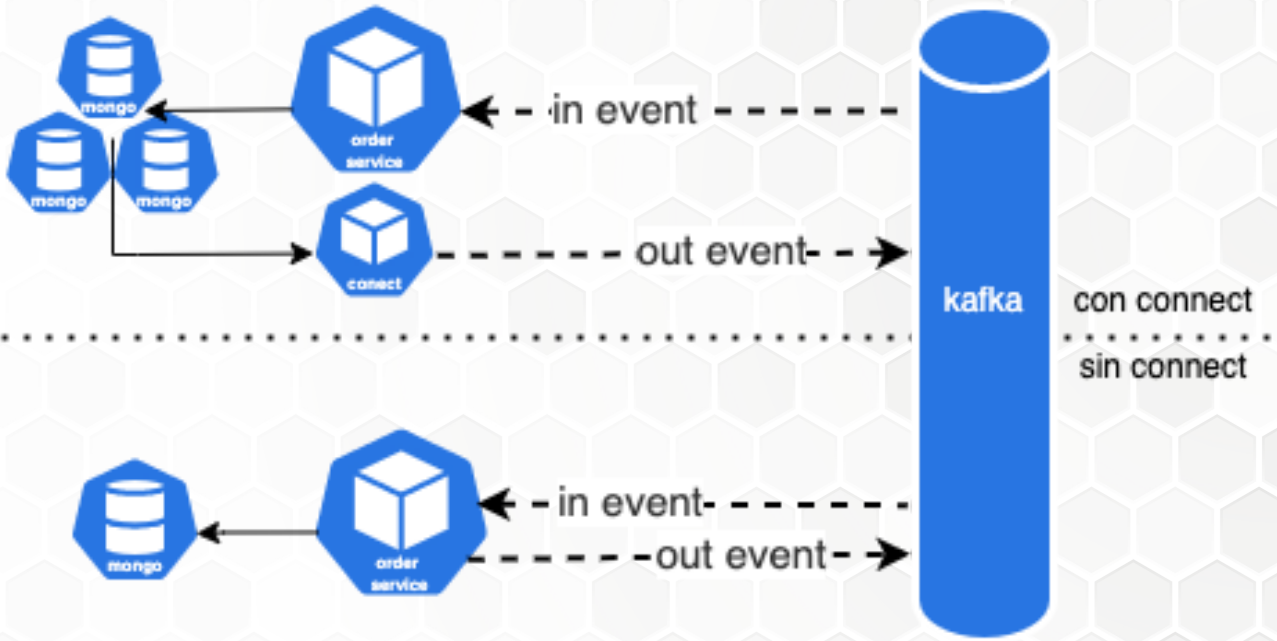
Pros

- envia eventos al persistir en bbdd
- simplifica idempotencia

Cons

- un servico más
- obliga a tener mongo en replica set de al menos 3 instancias

> existen ramas con ambas opciones implementadas



Idempotencia

Cuando un evento es leido se guarda el offset y no se marca como leido hasta que no se ha terminado de enviar. Da igual en el punto en el que se caiga el servicio, si se restaura volveria a procesarse de nuevo