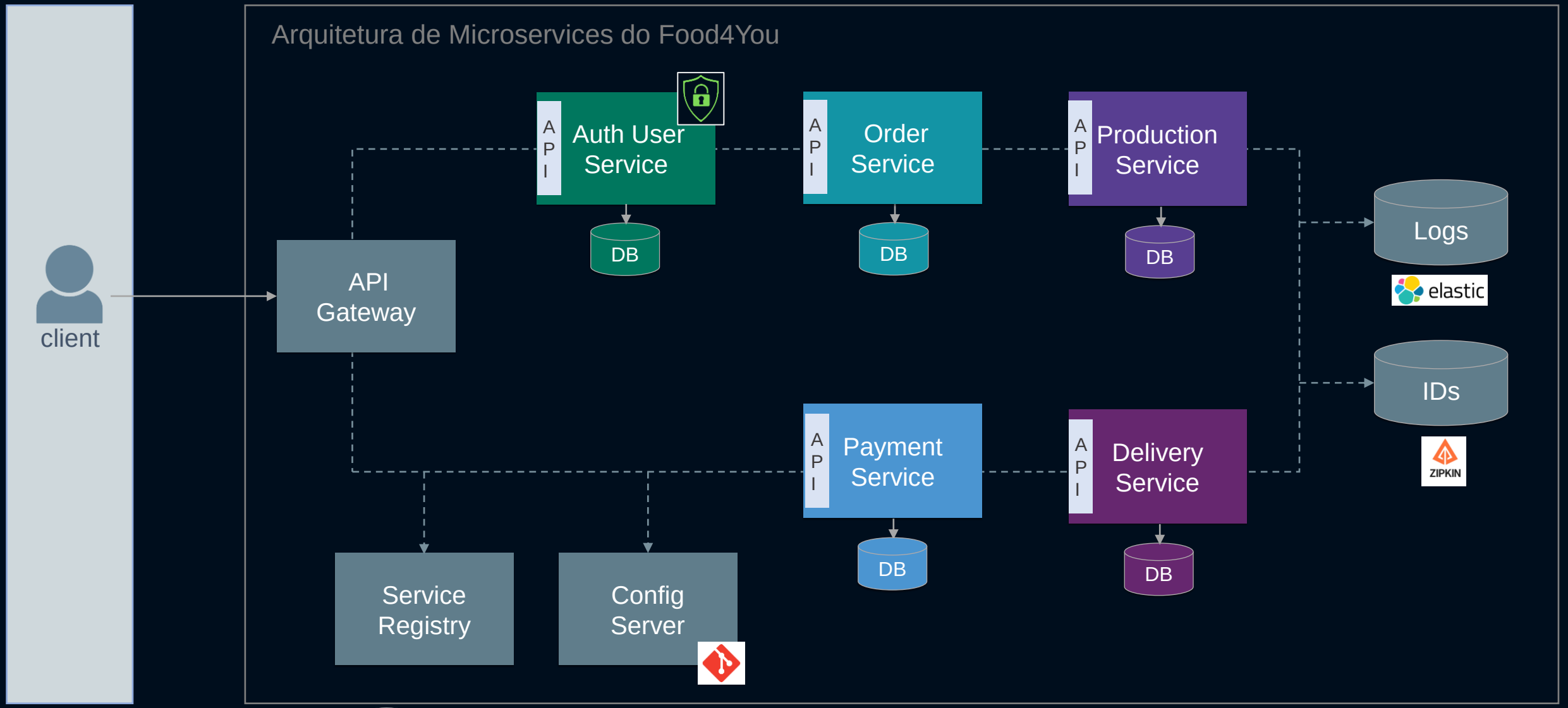
DECODER WEEK

Patterns para Arquiteturas de Microservices

DIA 3

Arquitetura completa do **Food4You**



DESCOMPLICANDO

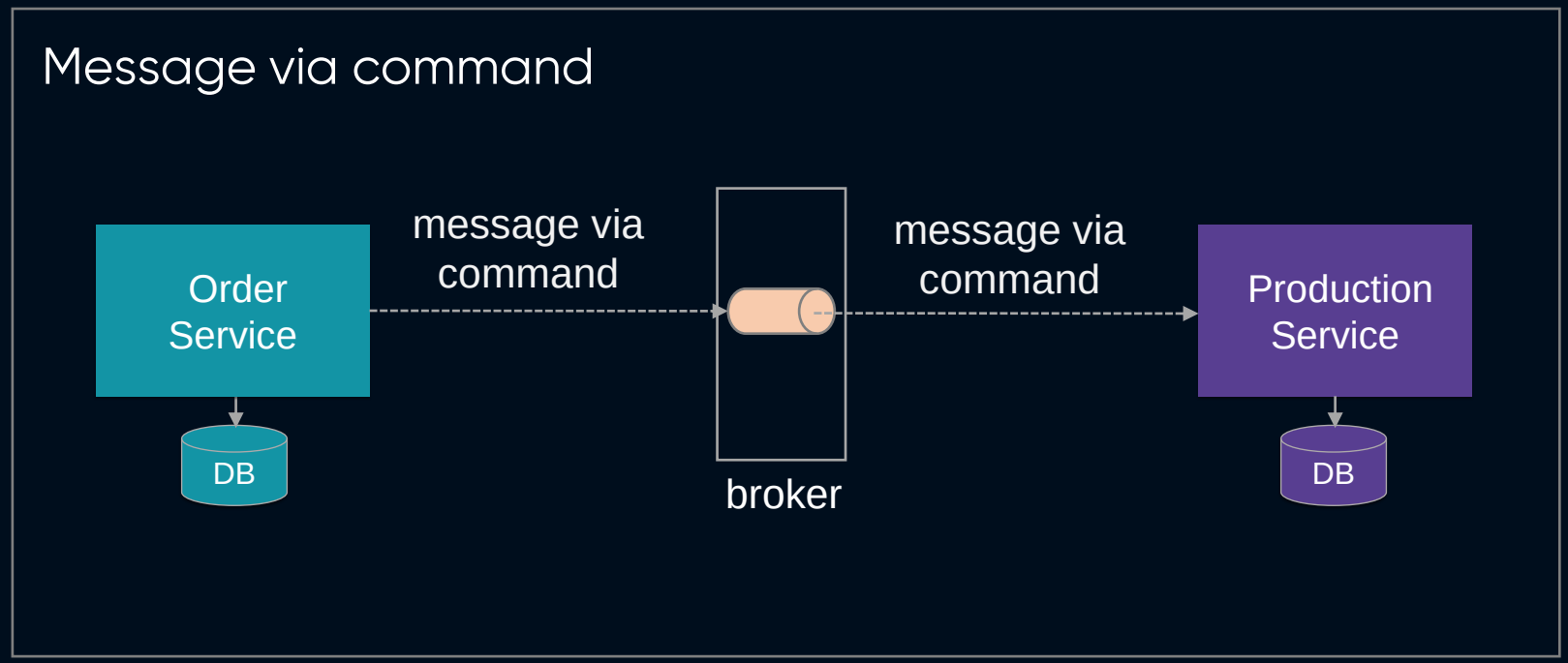
#decoderweek

Microservices Patterns são um conjunto de ferramentas para não reinventar a roda em soluções já conhecidas.

Boas práticas e padrões em Microservices

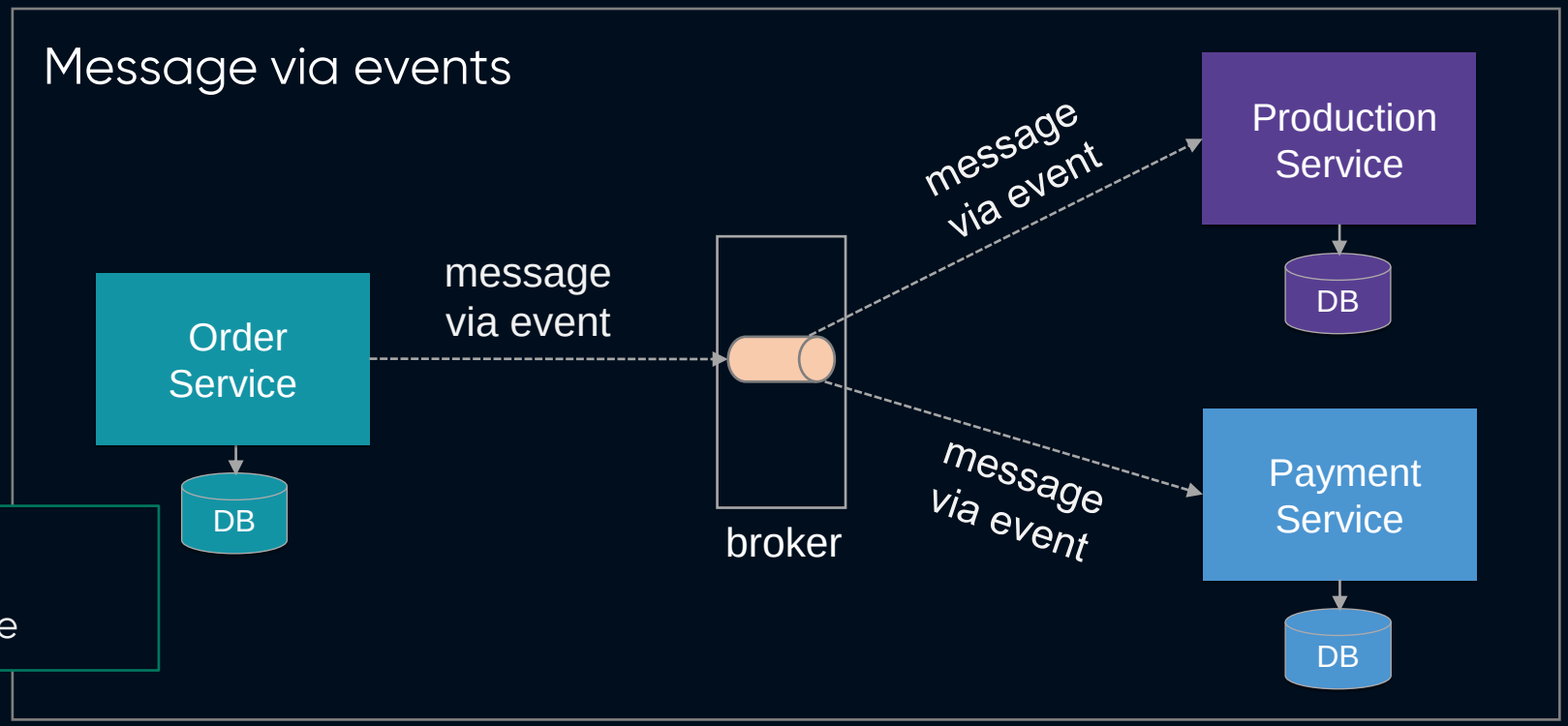
- Broker Pattern e Mediator Pattern
- API Composition Pattern
- Event Notification Pattern
- Registry Discovery Pattern
- Event-Carried State Transfer Pattern
- Distributed Tracing Pattern
- Saga Pattern com Coreografia
- Log Aggregation Pattern
- Saga Pattern com Orquestração
- Ports and Adapters Pattern

Broker Pattern



Utilizado nas comunicações assíncronas via mensageria.

Broker Pattern



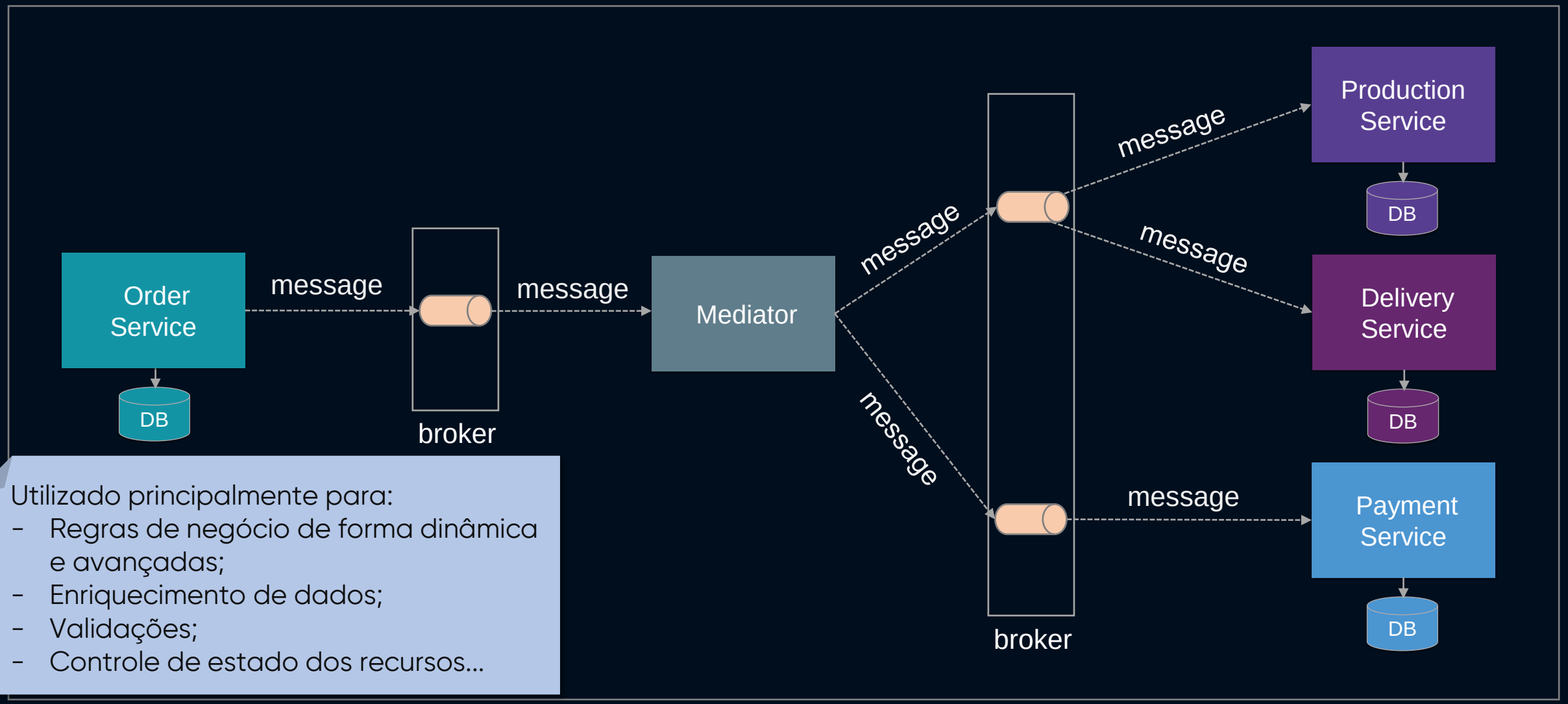
Vantagens

- Menor acoplamento
- Maior disponibilidade

Pontos de Atenção

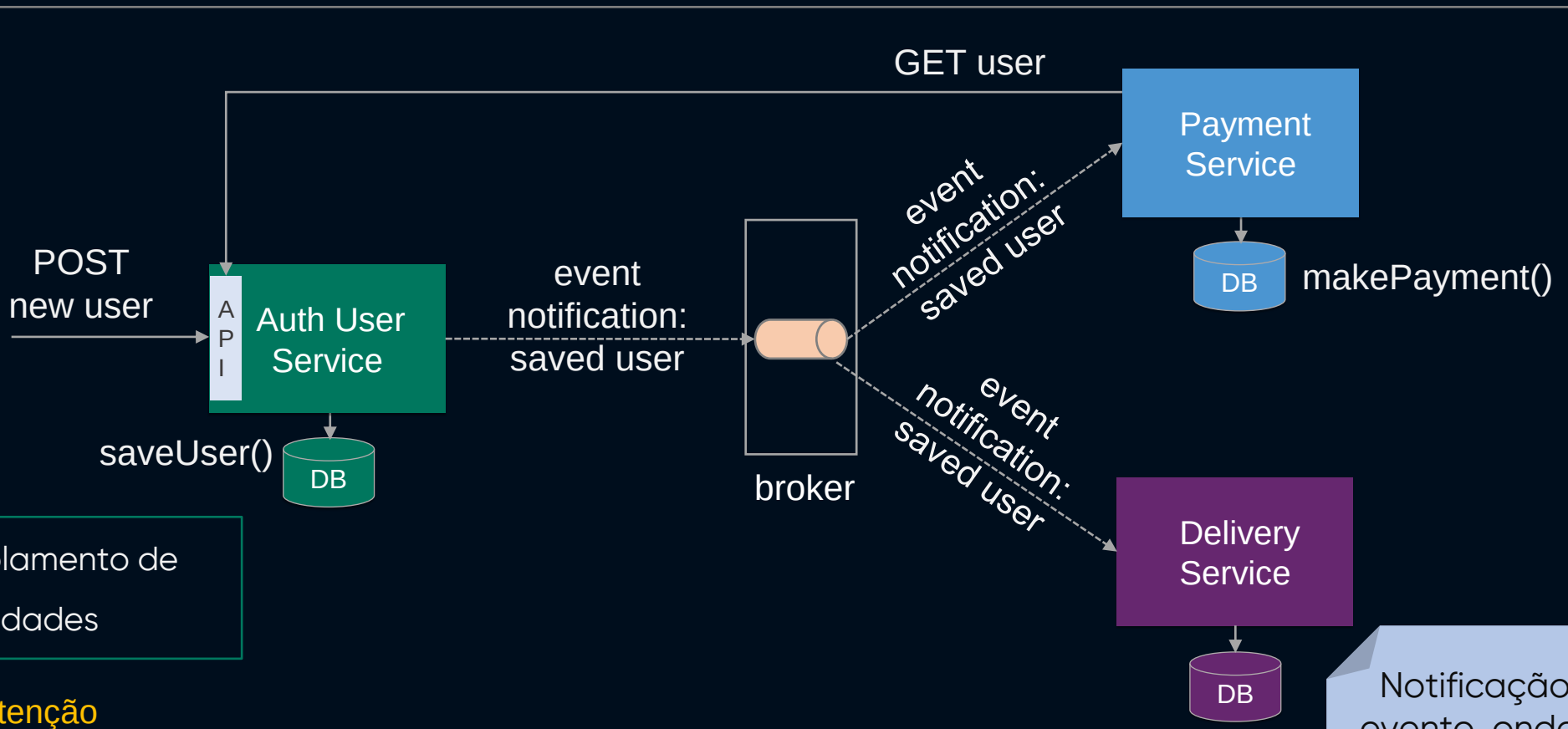
Cuidado com os desdobramentos de eventos

Mediator Pattern



Utilizado principalmente para:
- Regras de negócio de forma dinâmica e avançadas;
- Enriquecimento de dados;
- Validações;
- Controle de estado dos recursos...

Event Driven – Event Notification Pattern



Vantagens

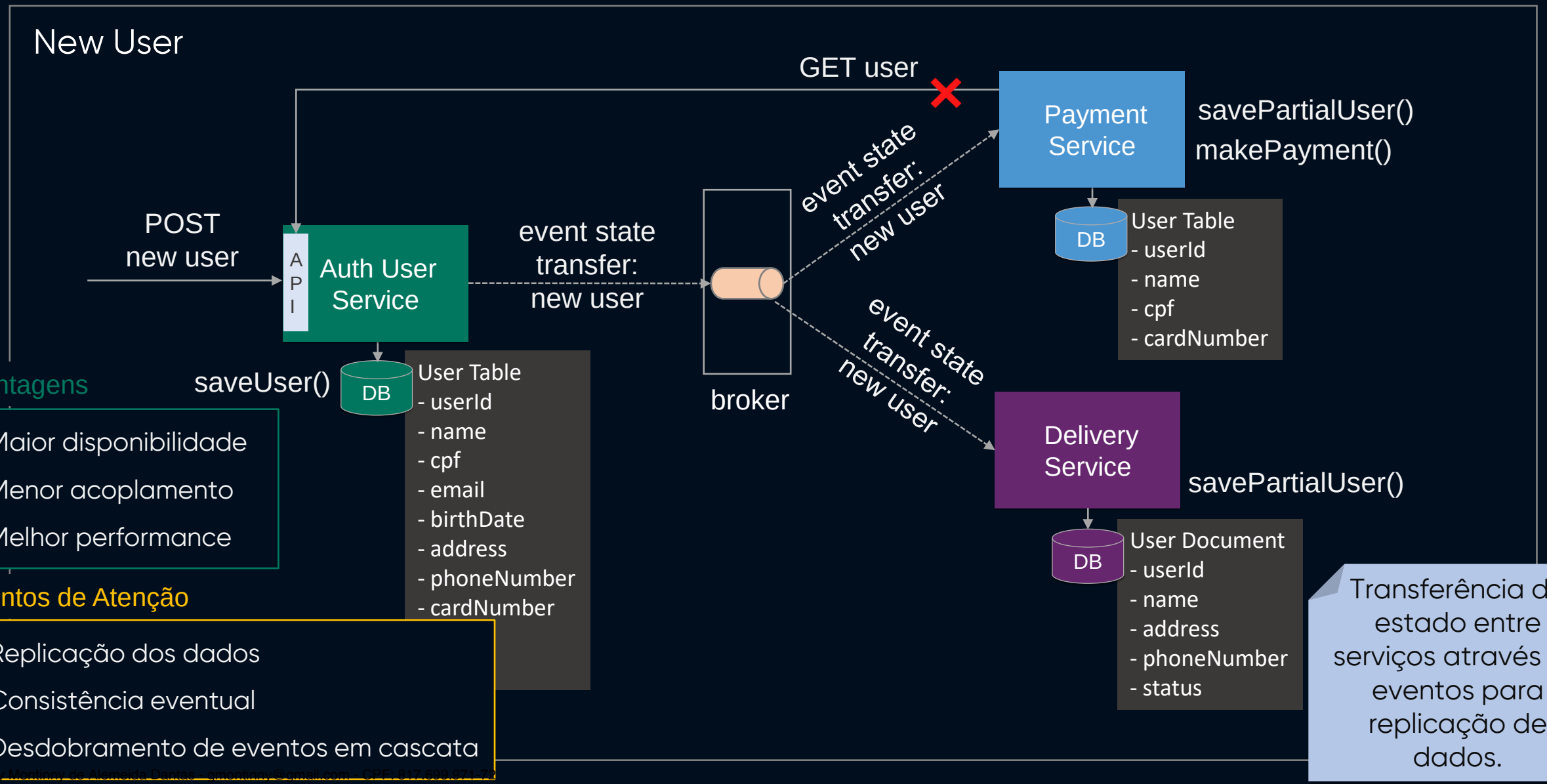
Menor acoplamento de responsabilidades

Pontos de Atenção

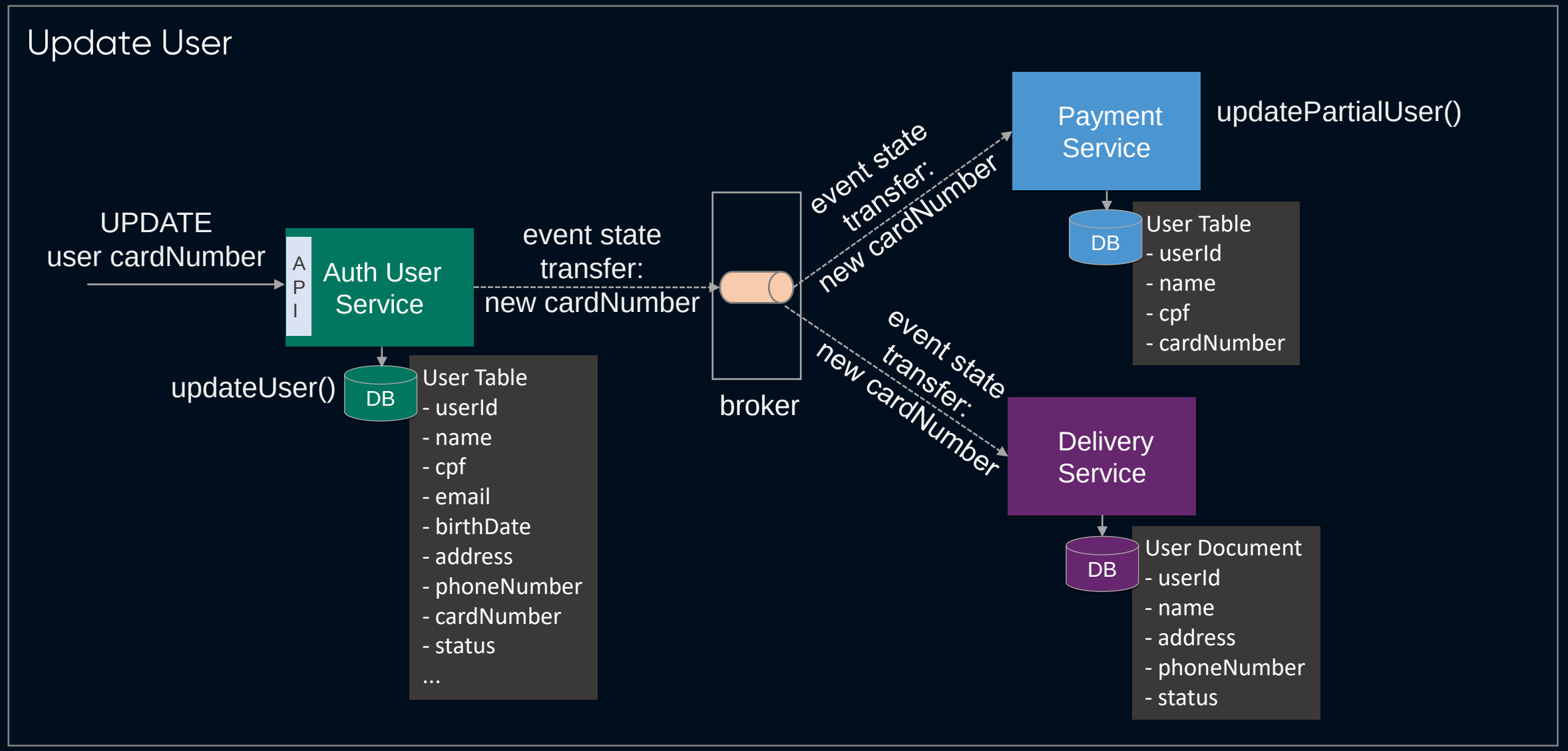
Pode gerar uma sobrecarga de sub-consultas na arquitetura

Notificação através de um evento, onde os serviços que precisam processar esse evento devem recorrer ao serviço primário para obter o estado atual do recurso.

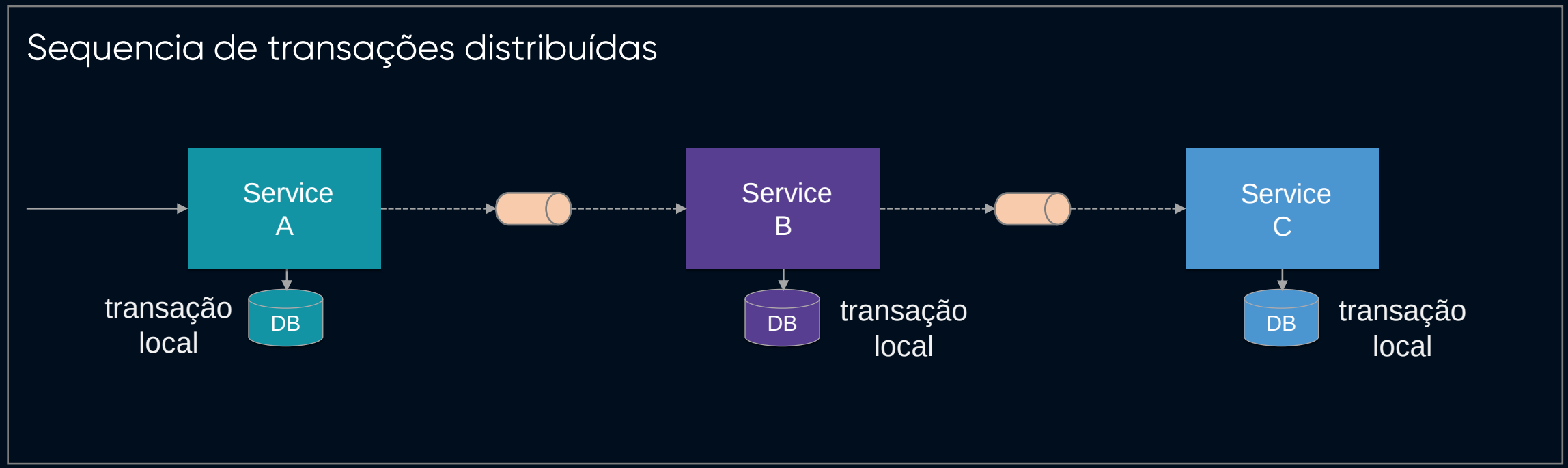
Event Driven – Event Carried State Transfer Pattern



Event Driven – Event Carried State Transfer Pattern

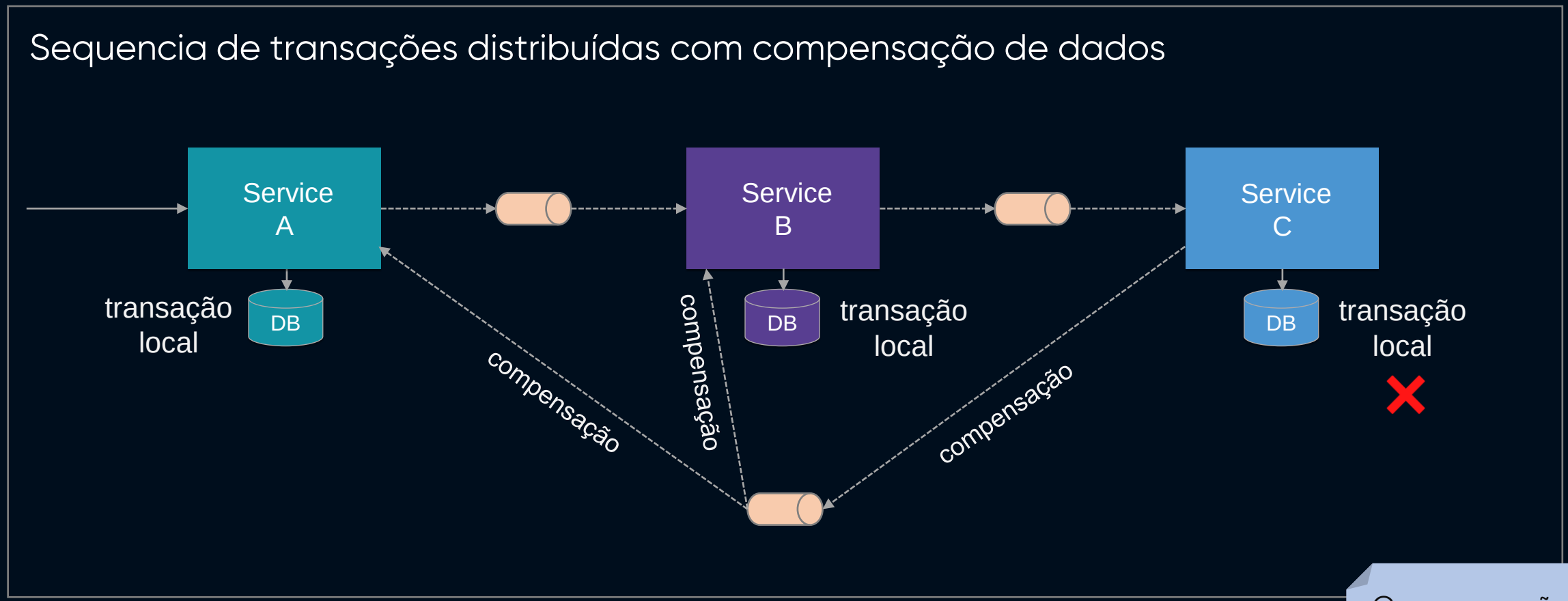


Saga Pattern



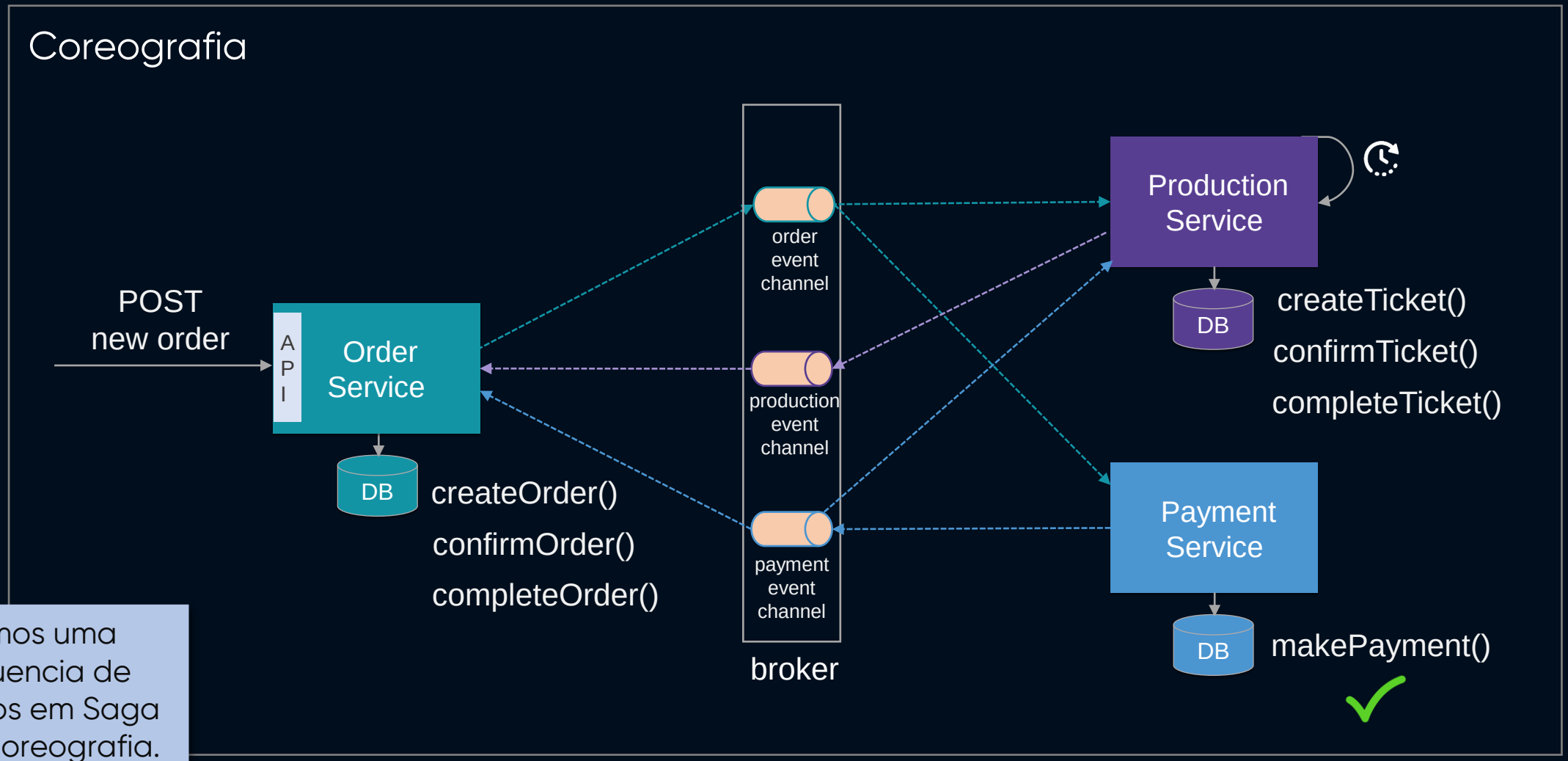
Sequencia de transações locais que envolvem múltiplos Microservices quando temos dados distribuídos na arquitetura.

Saga Pattern

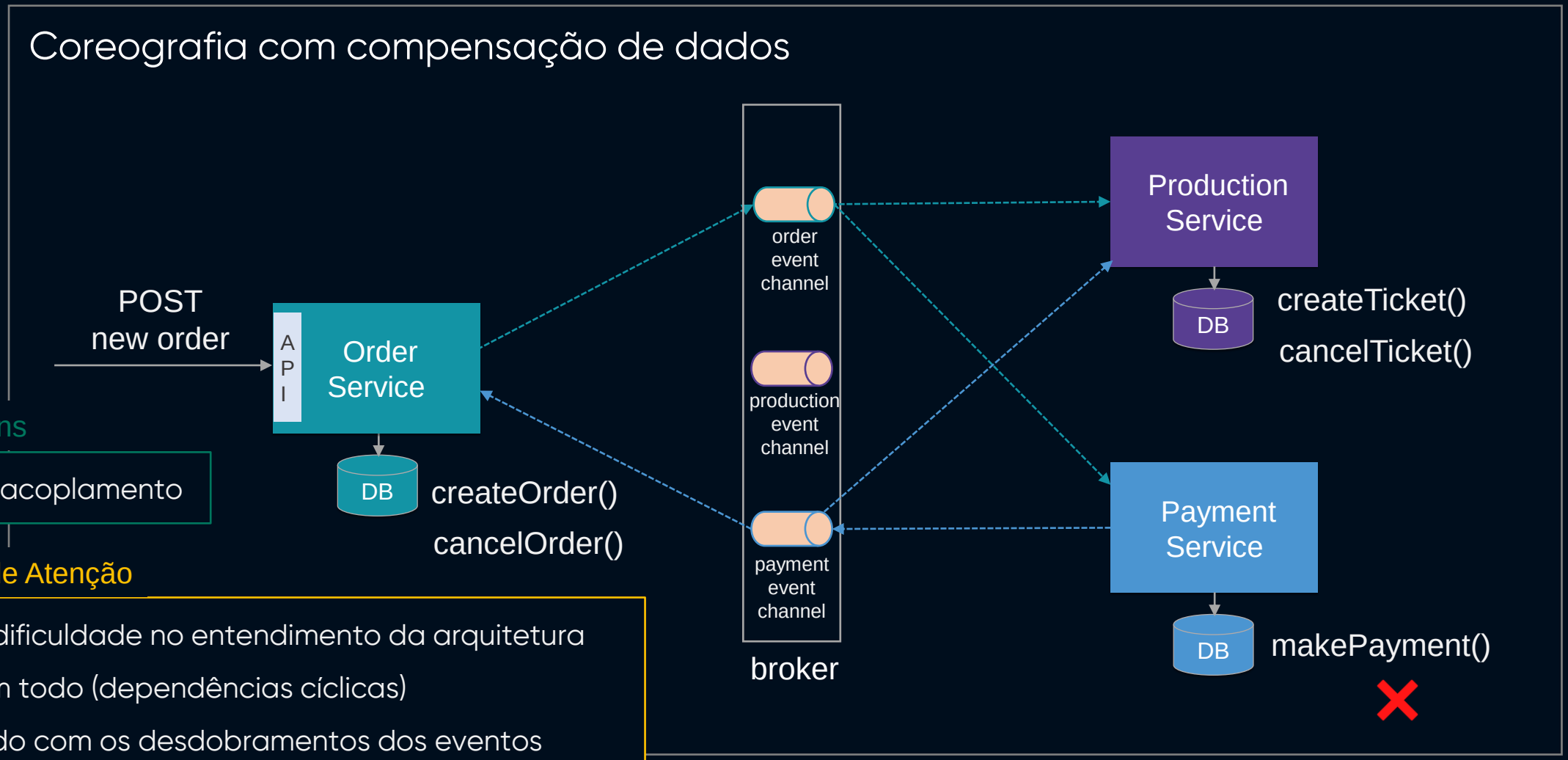


Compensação de dados do estado atual dos recursos.

Saga Pattern com Coreografia



Saga Pattern com Coreografia



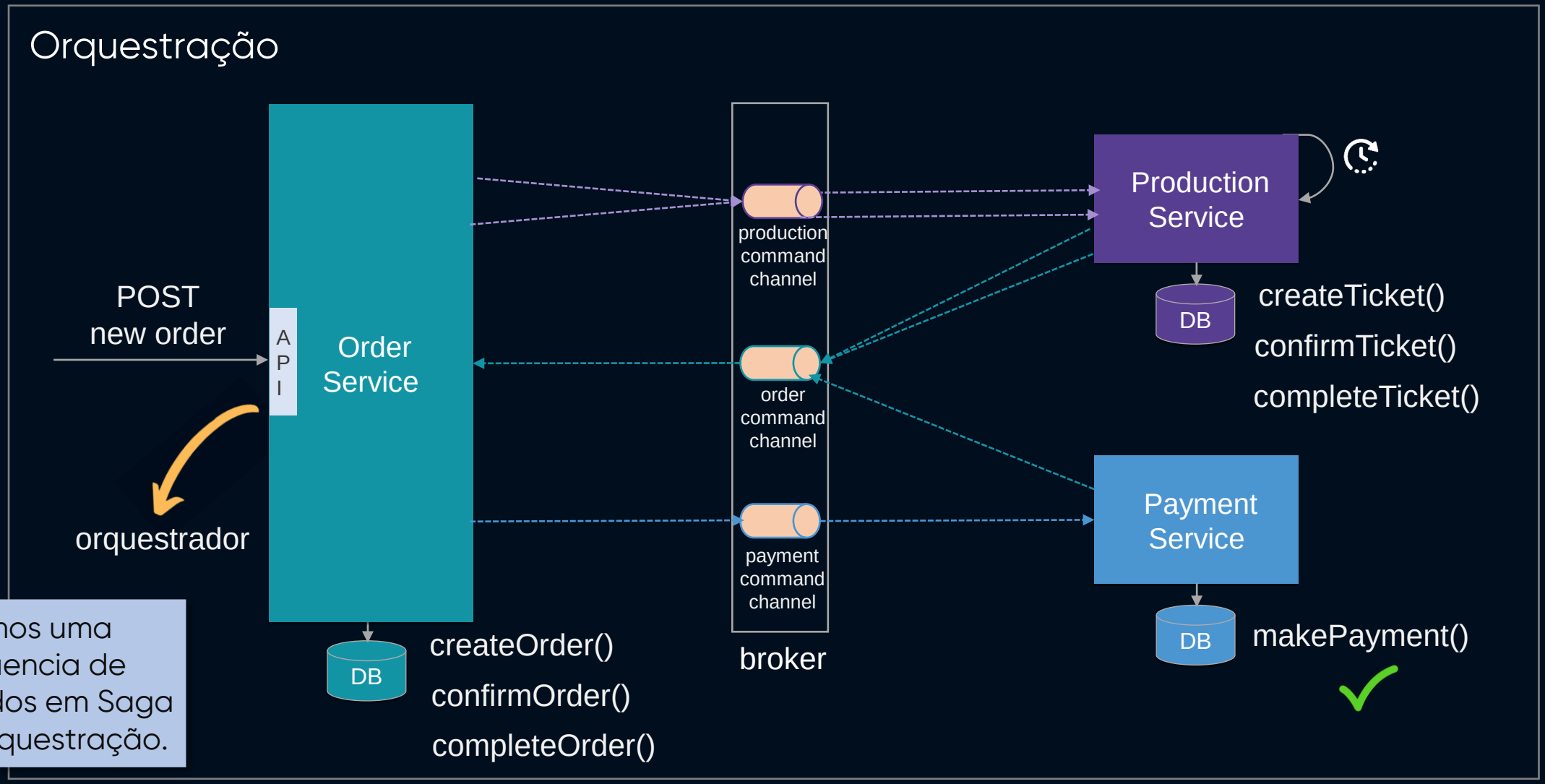
Vantagens

- Menor acoplamento

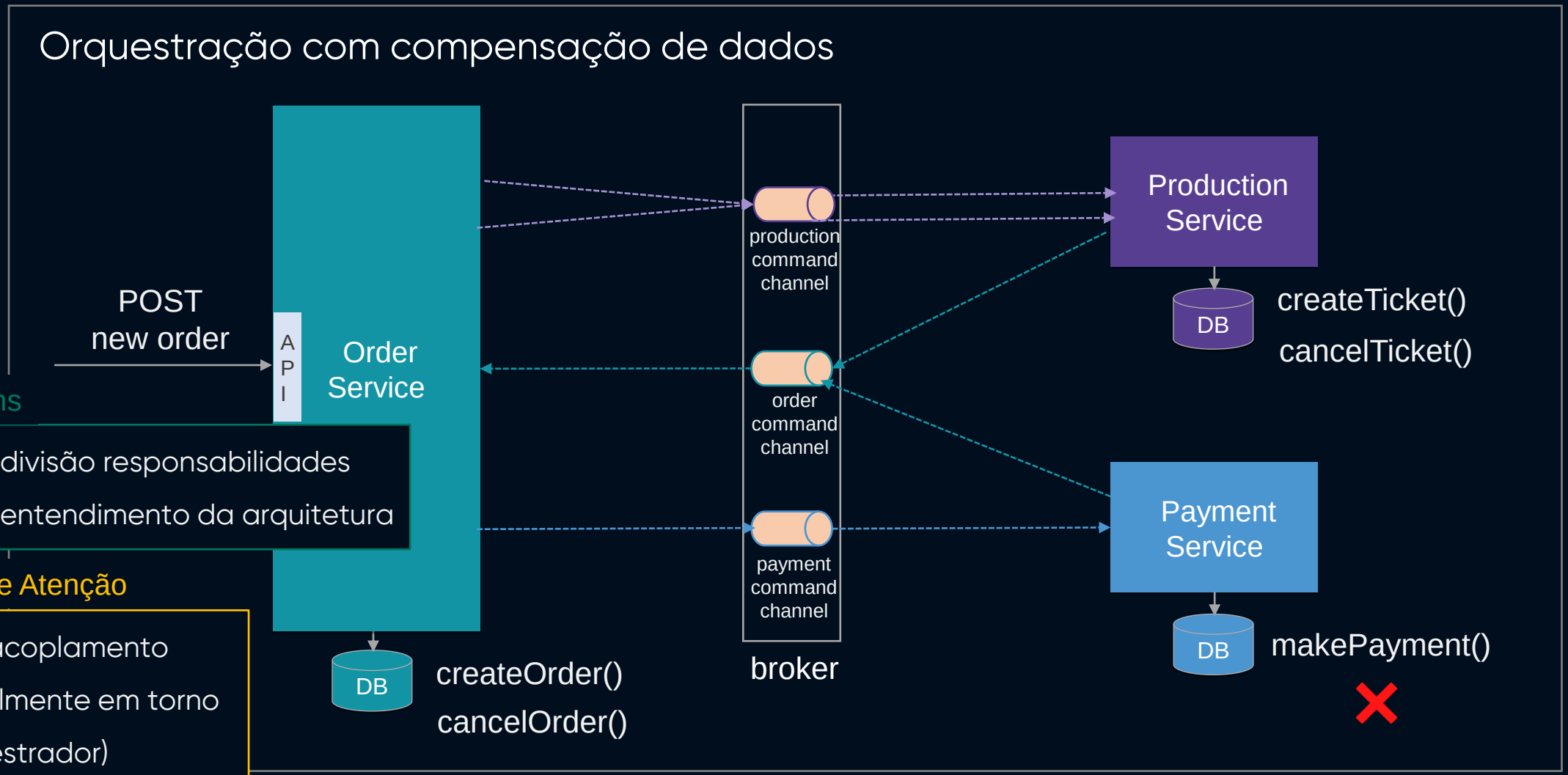
Pontos de Atenção

- Maior dificuldade no entendimento da arquitetura como um todo (dependências cíclicas)
- Cuidado com os desdobramentos dos eventos

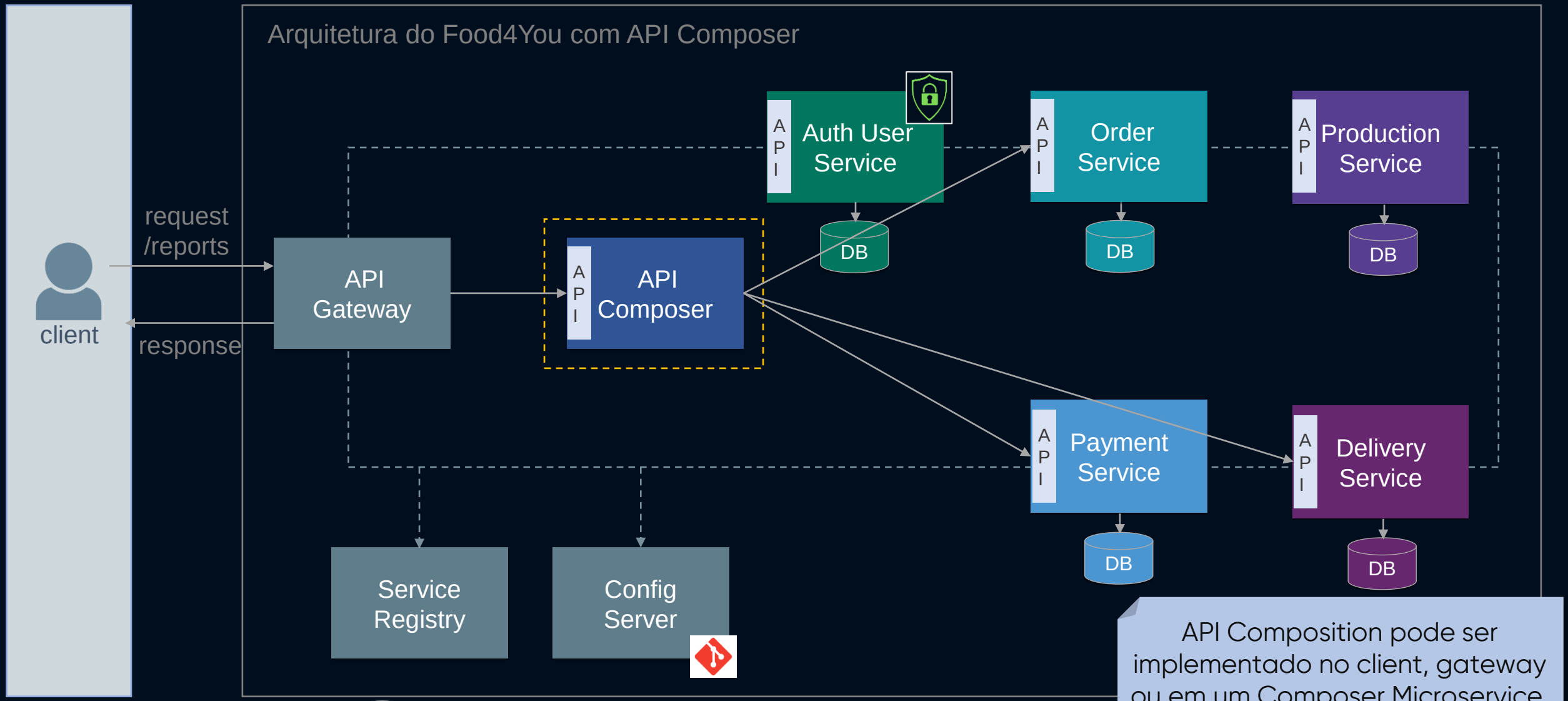
Saga Pattern com Orquestração



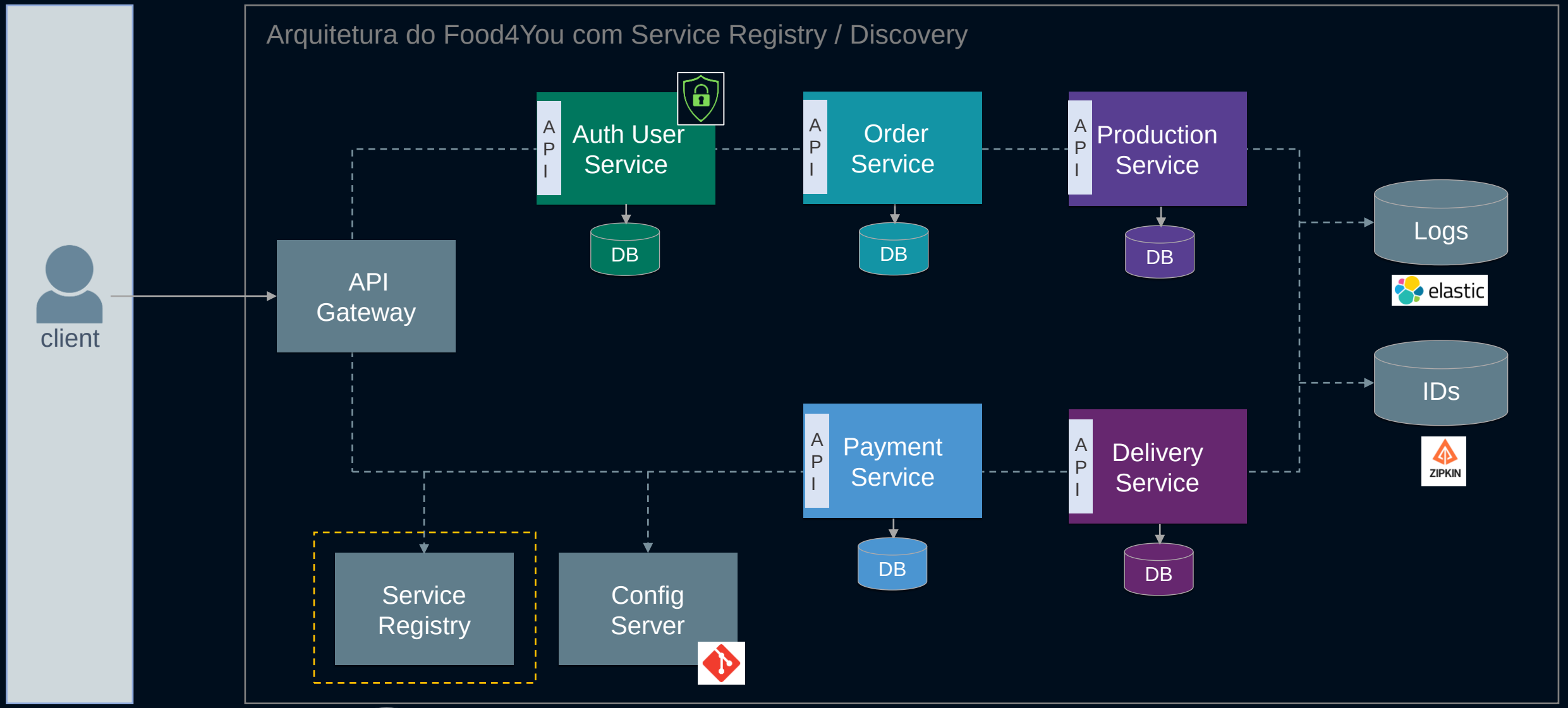
Saga Pattern com Orquestração



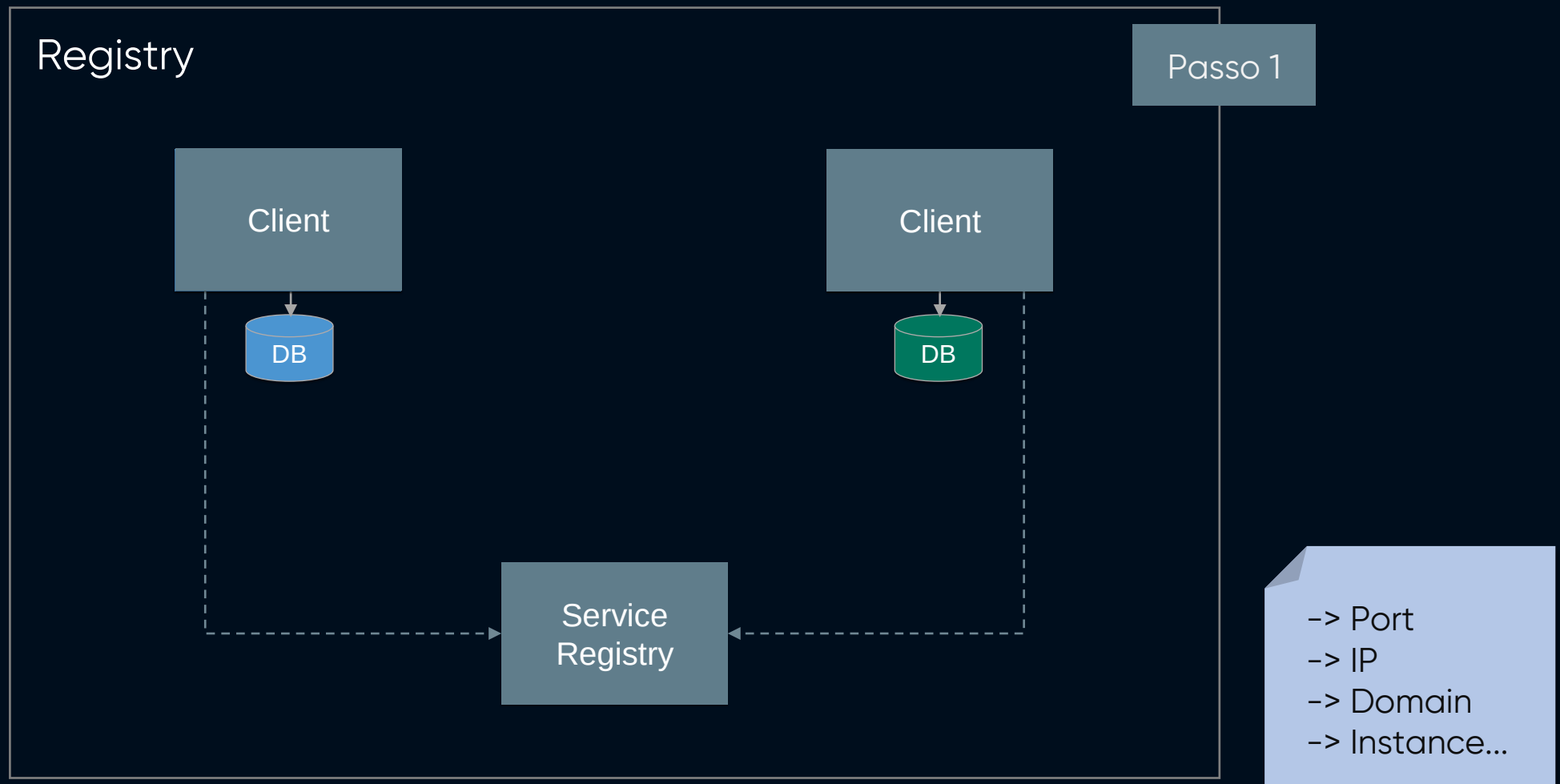
API Composition Pattern



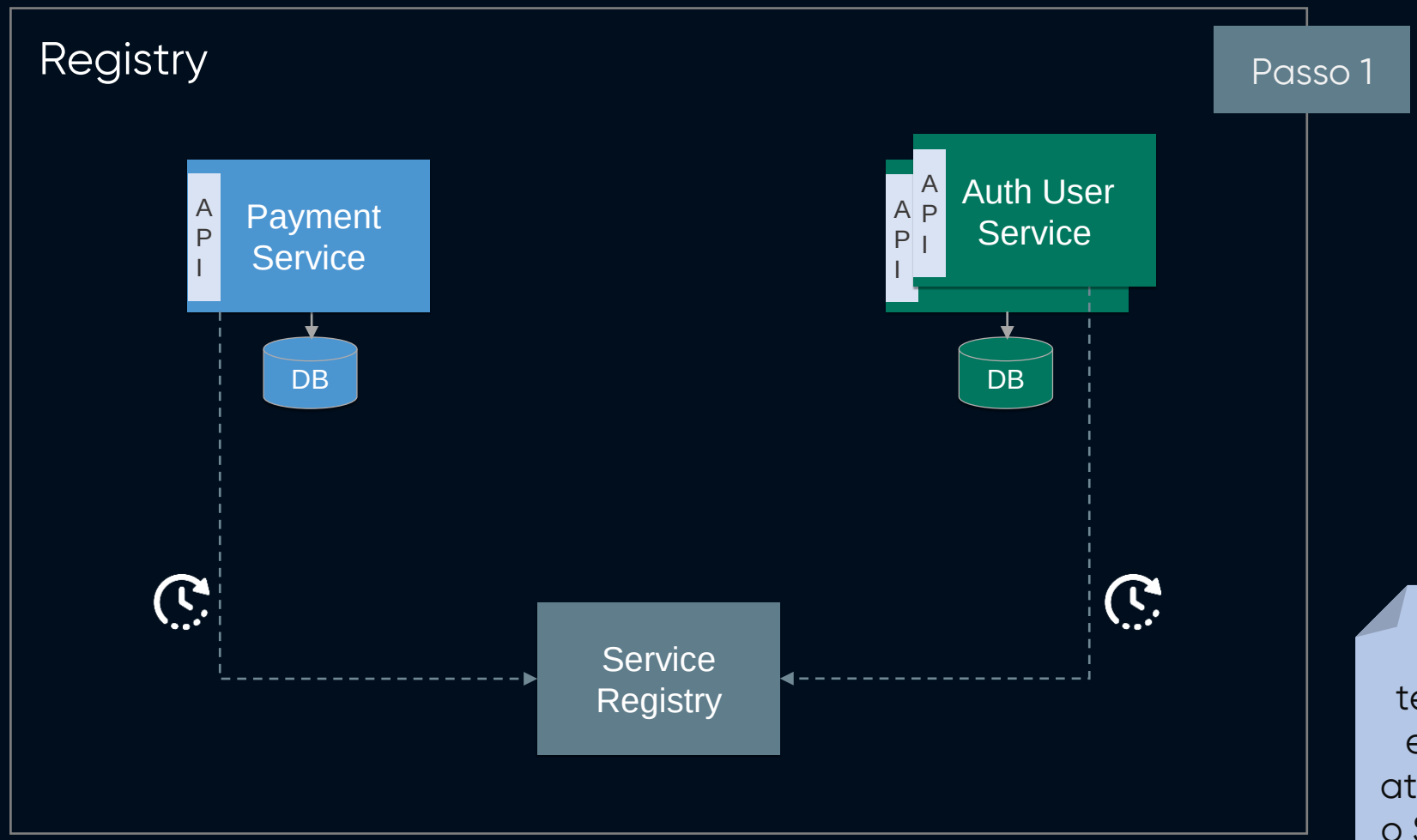
Registry Discovery Pattern



Registry Discovery Pattern

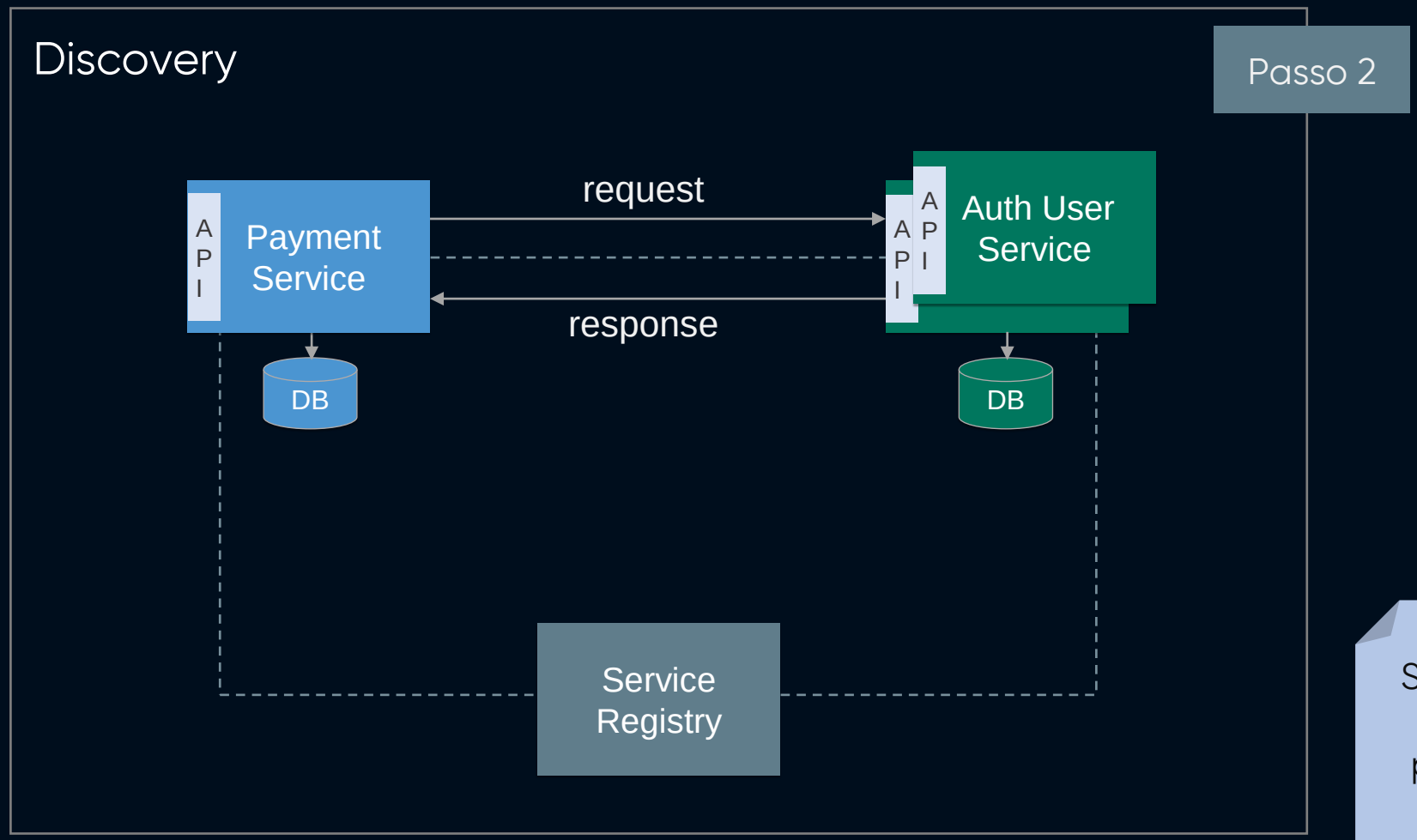


Registry Discovery Pattern



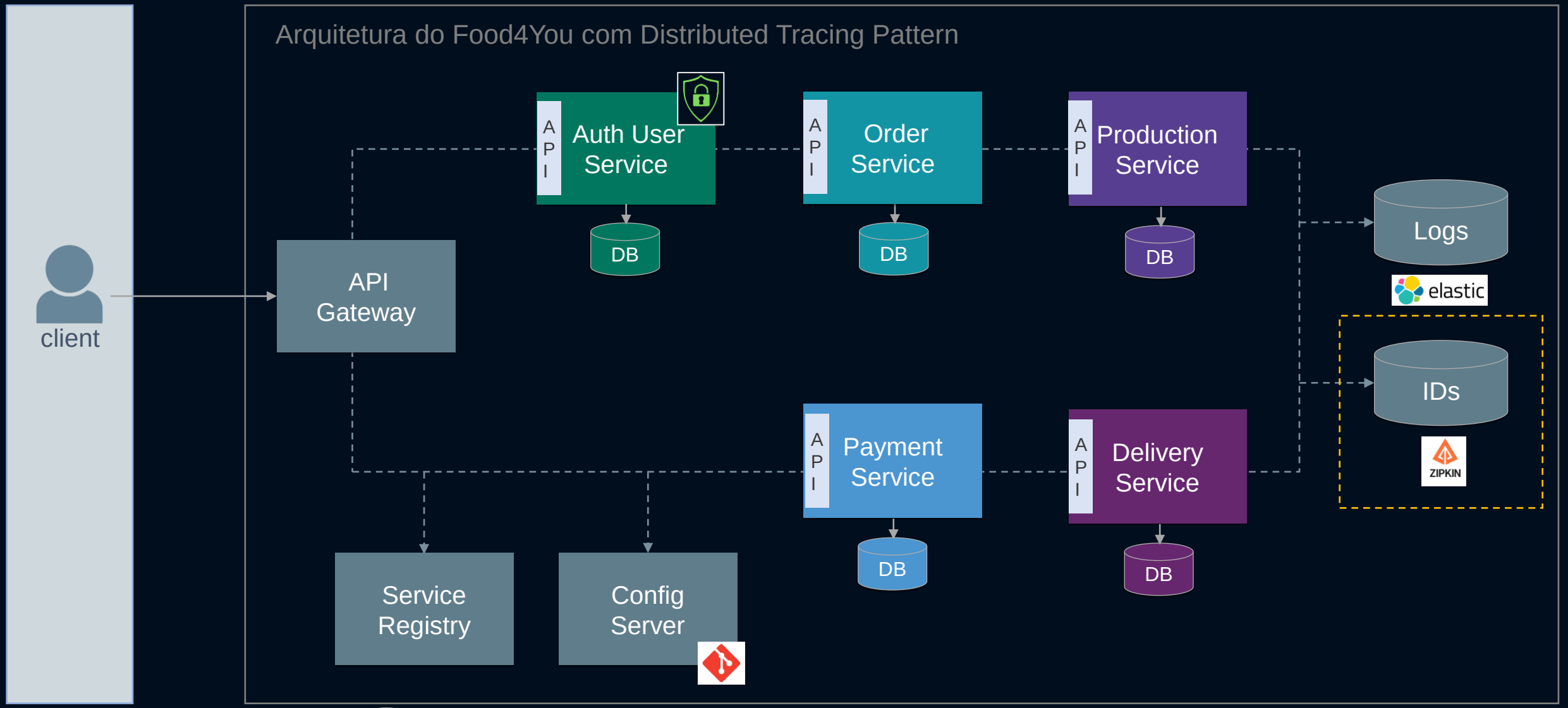
De tempo em tempo os clients enviam as suas atualizações para o Service Registry.

Registry Discovery Pattern

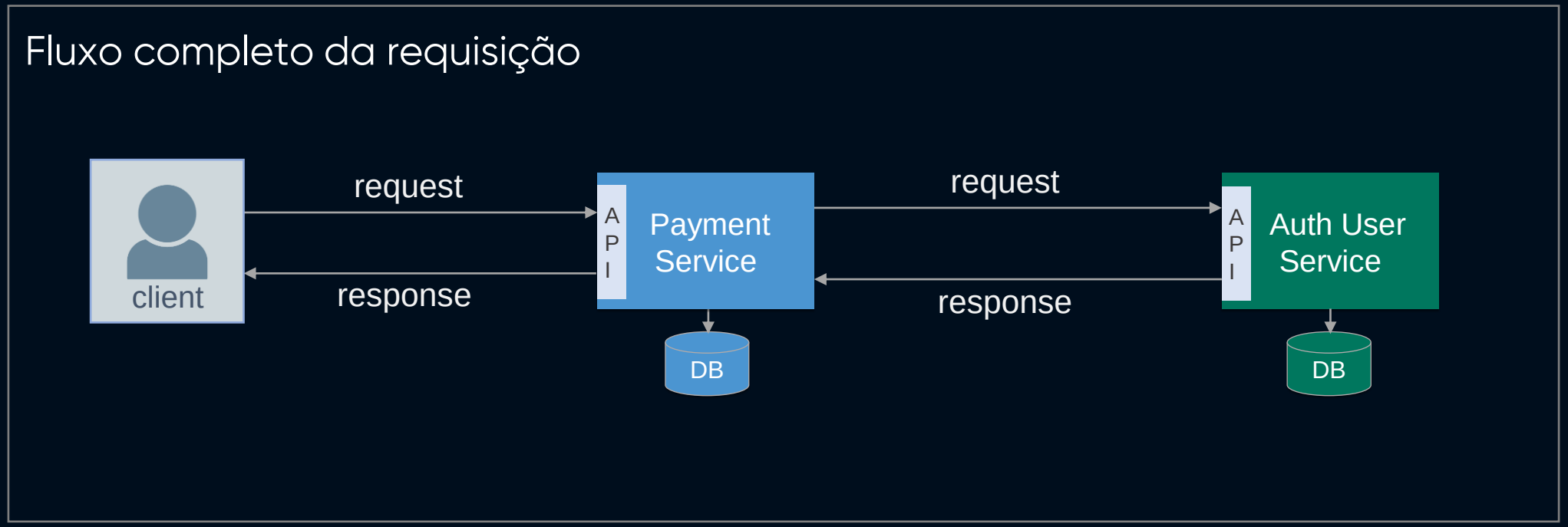


Service Registry envolvem as preocupações transversais

Distributed Tracing Pattern

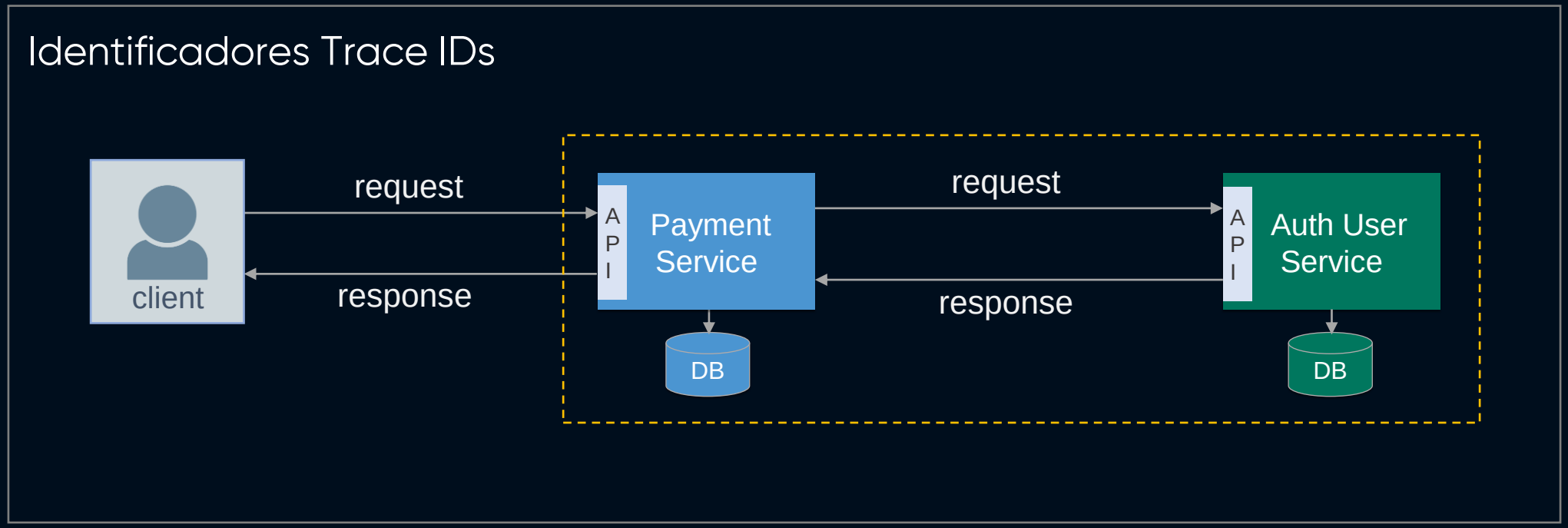


Distributed Tracing Pattern



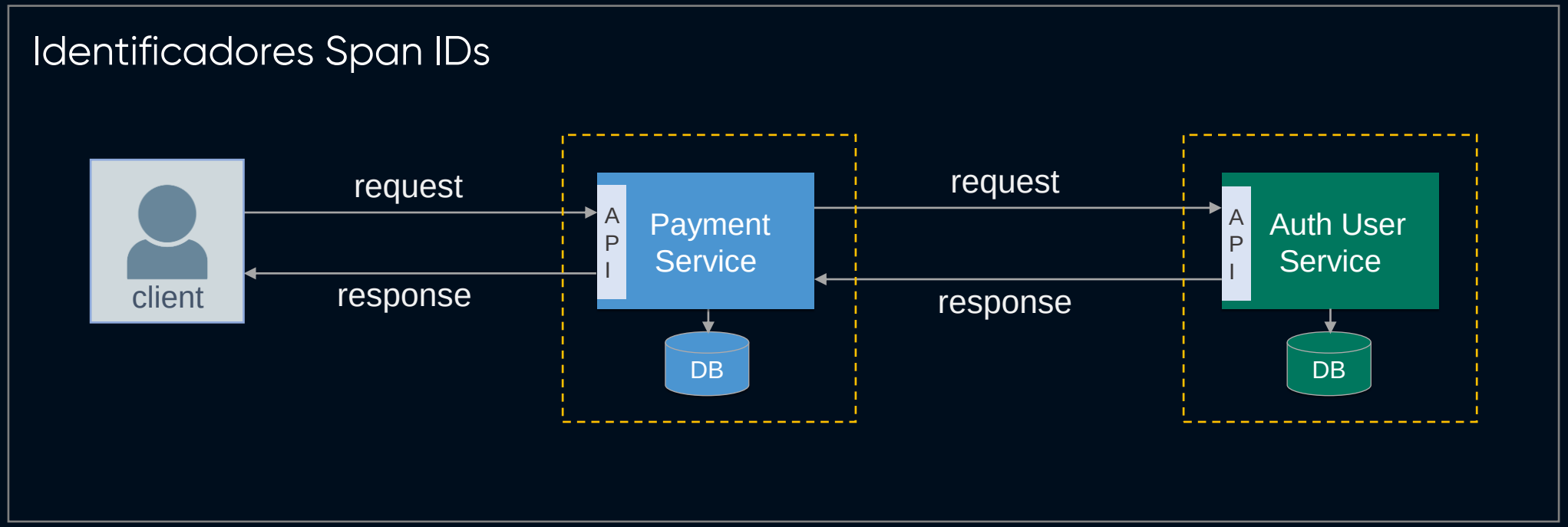
Tempo total da request = Payment Service + Auth User Service

Distributed Tracing Pattern



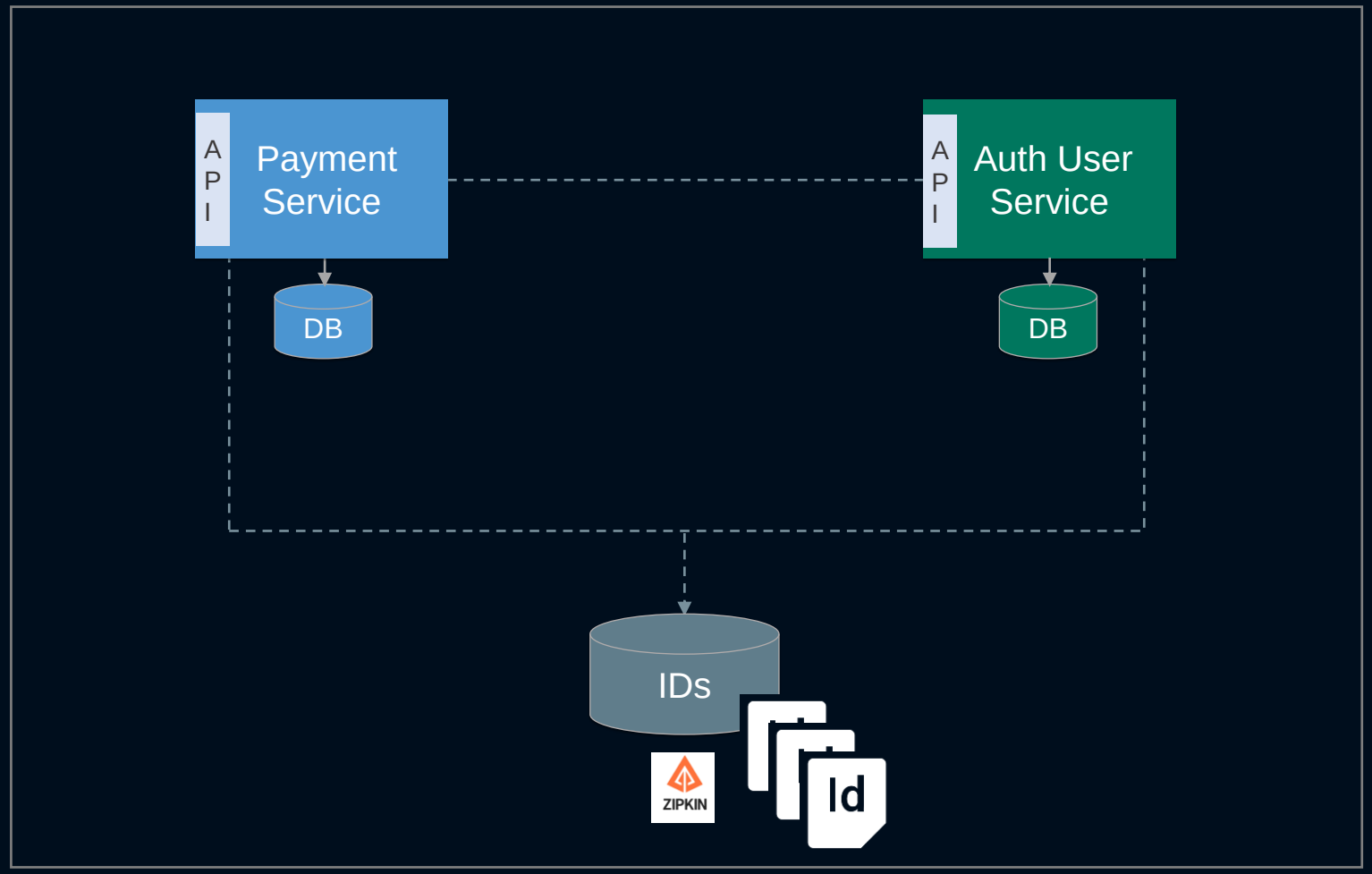
Tempo total da request = Payment Service + Auth User Service

Distributed Tracing Pattern

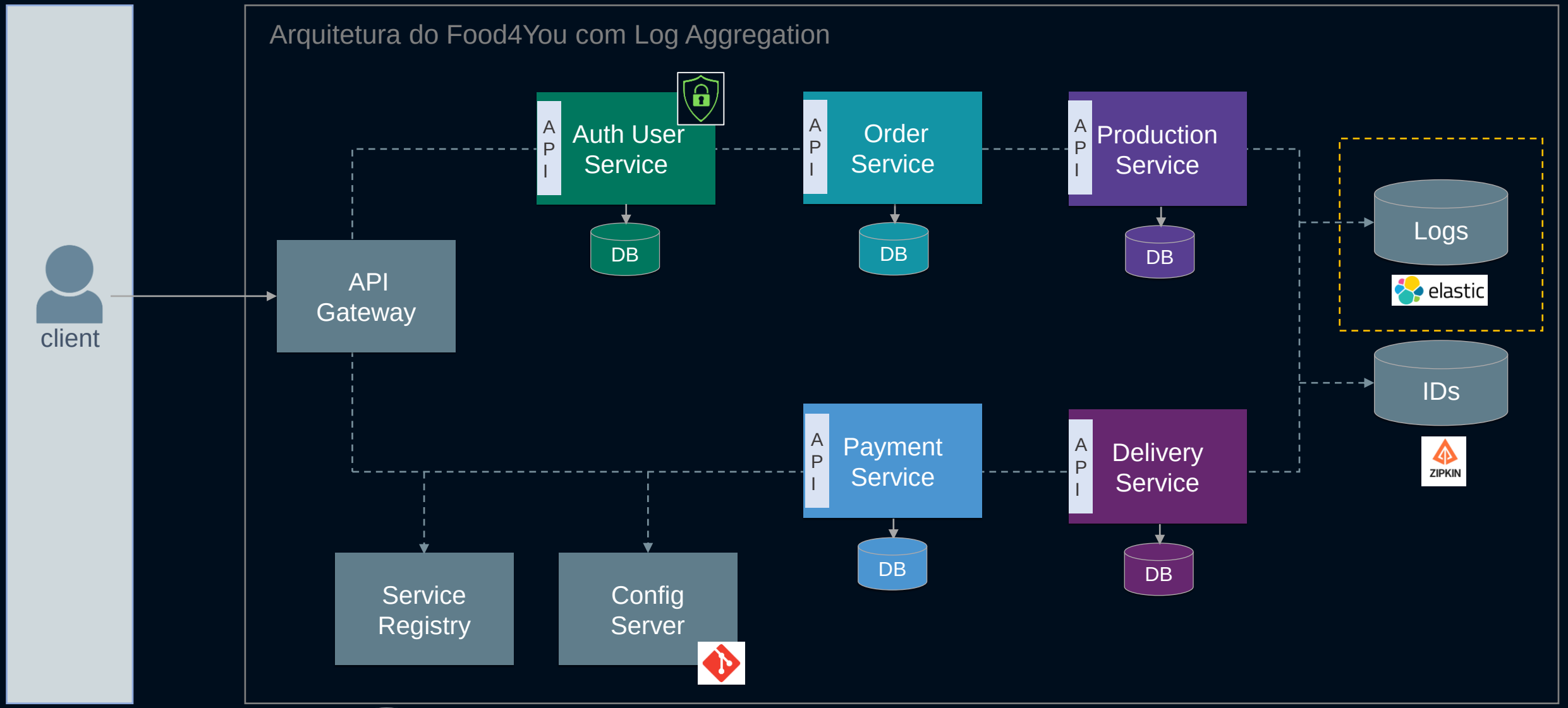


Tempo total da request = Payment Service + Auth User Service

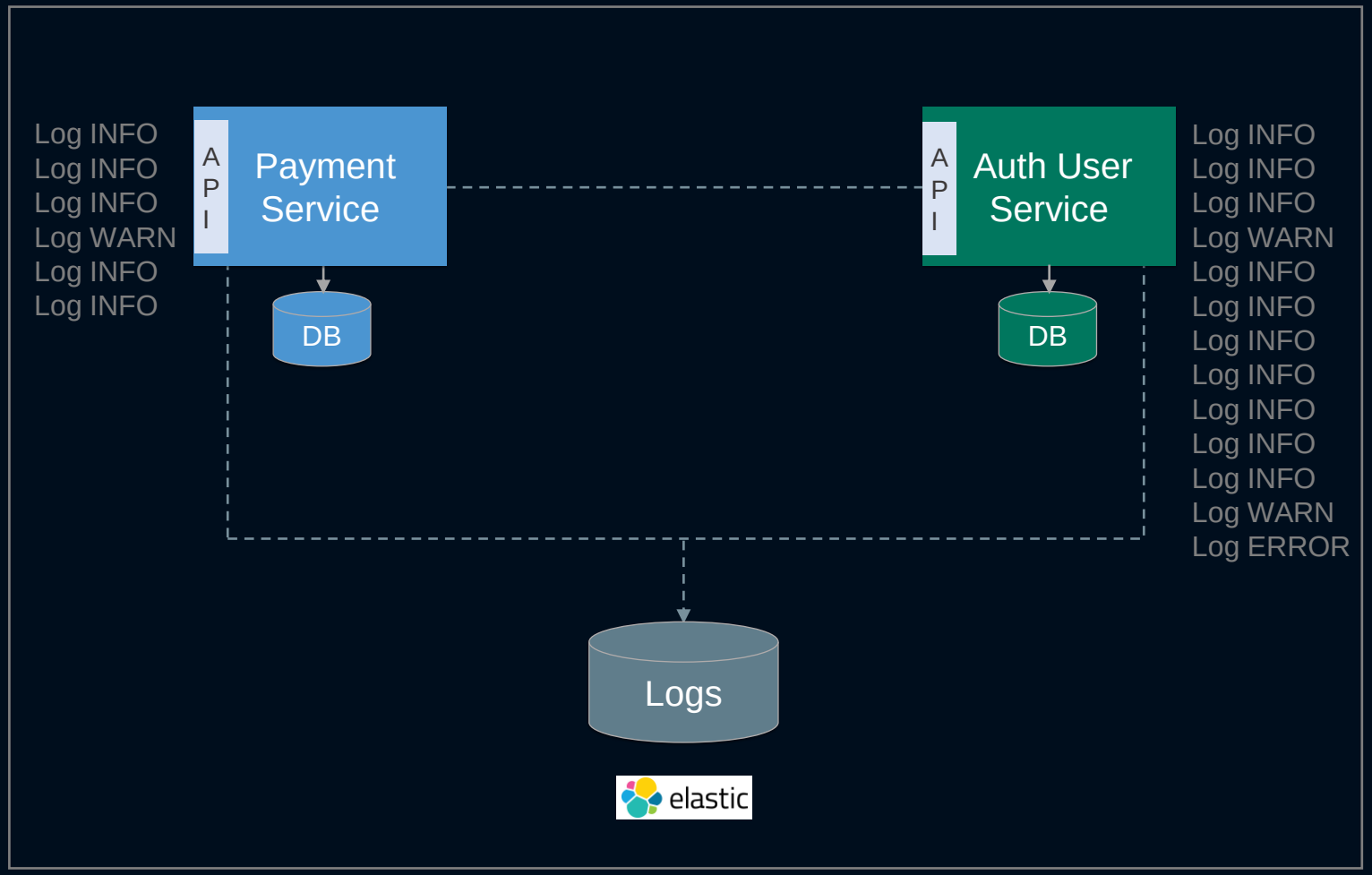
Distributed Tracing Pattern



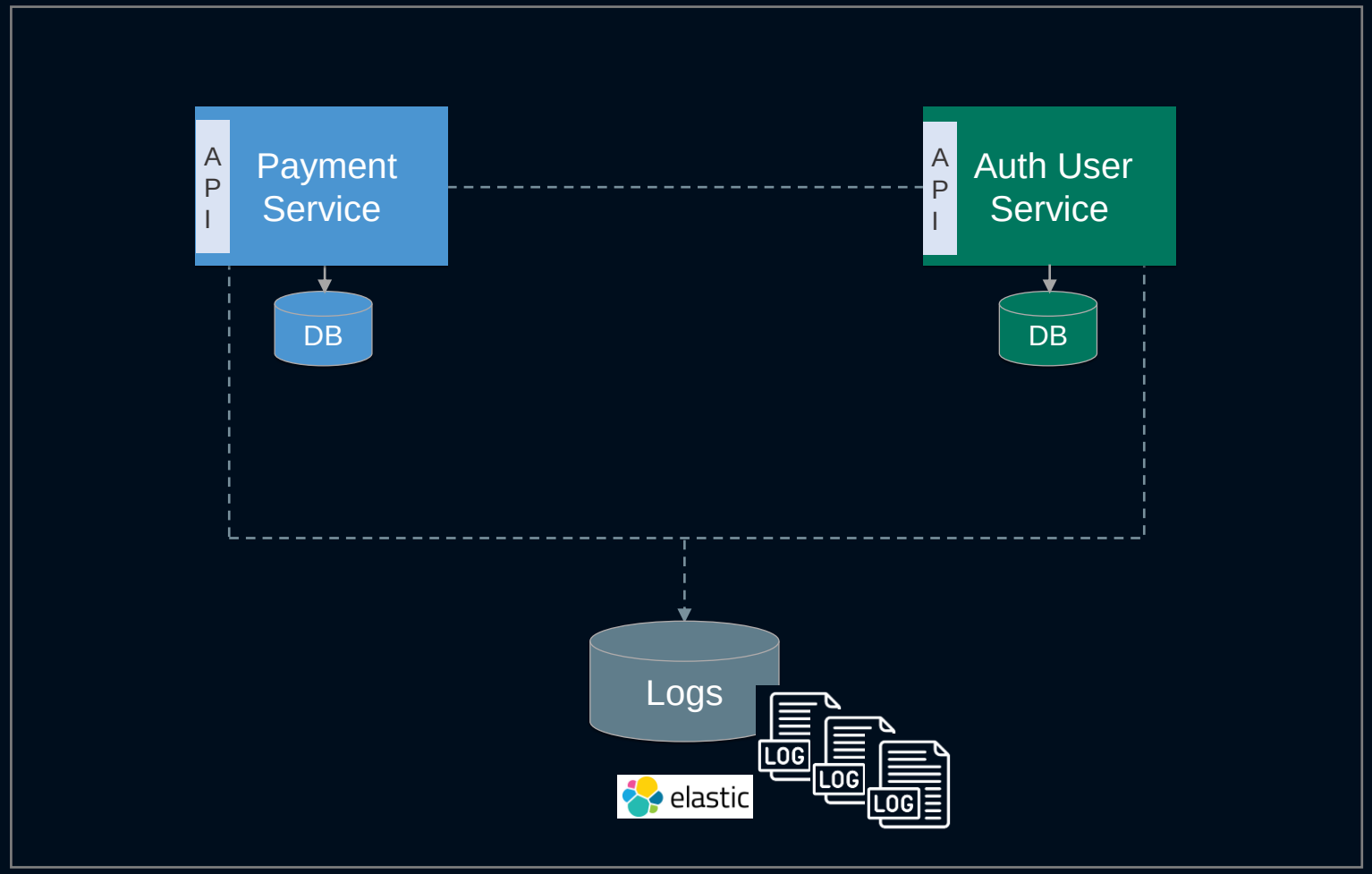
Log Aggregation Pattern



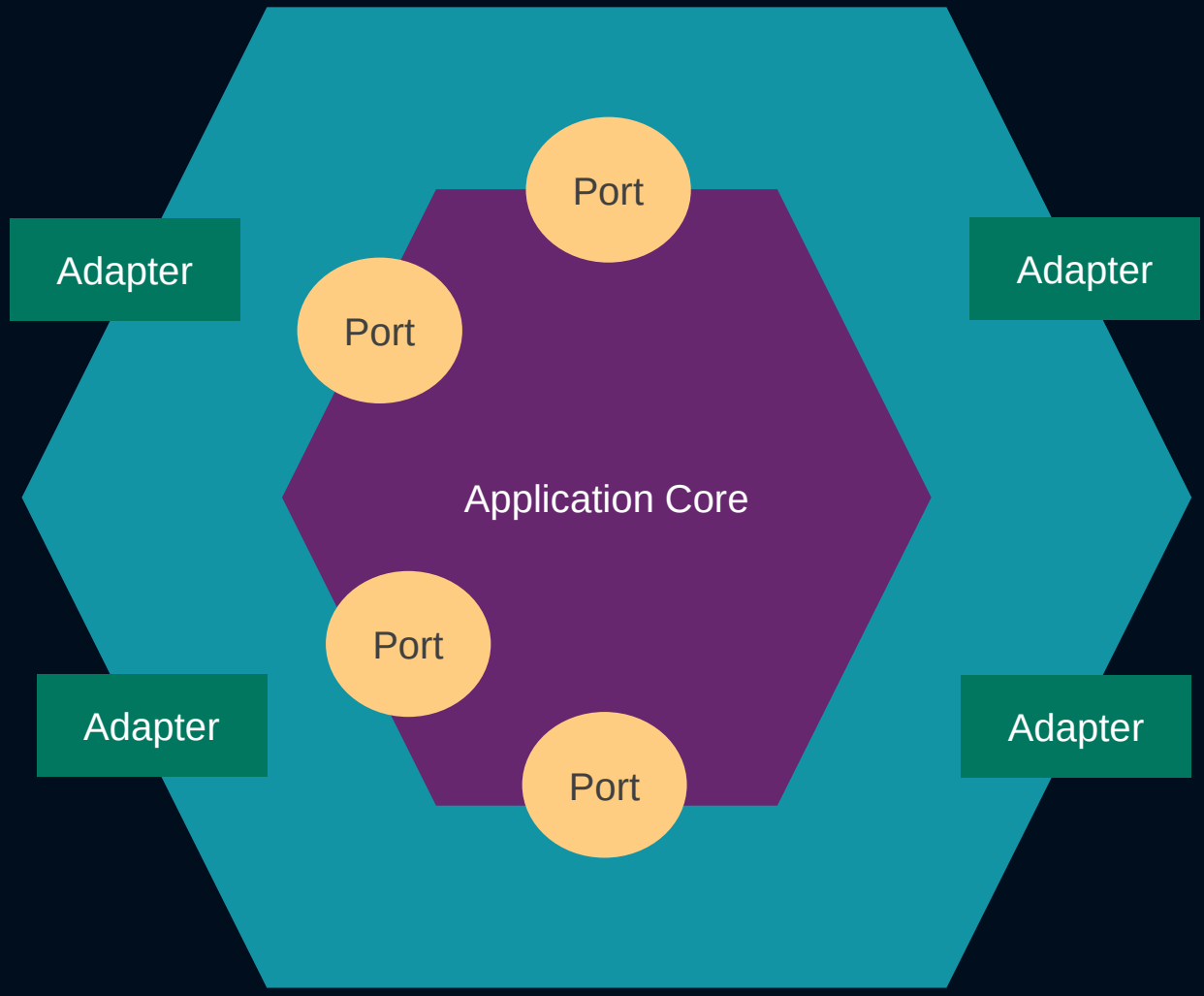
Log Aggregation Pattern



Log Aggregation Pattern

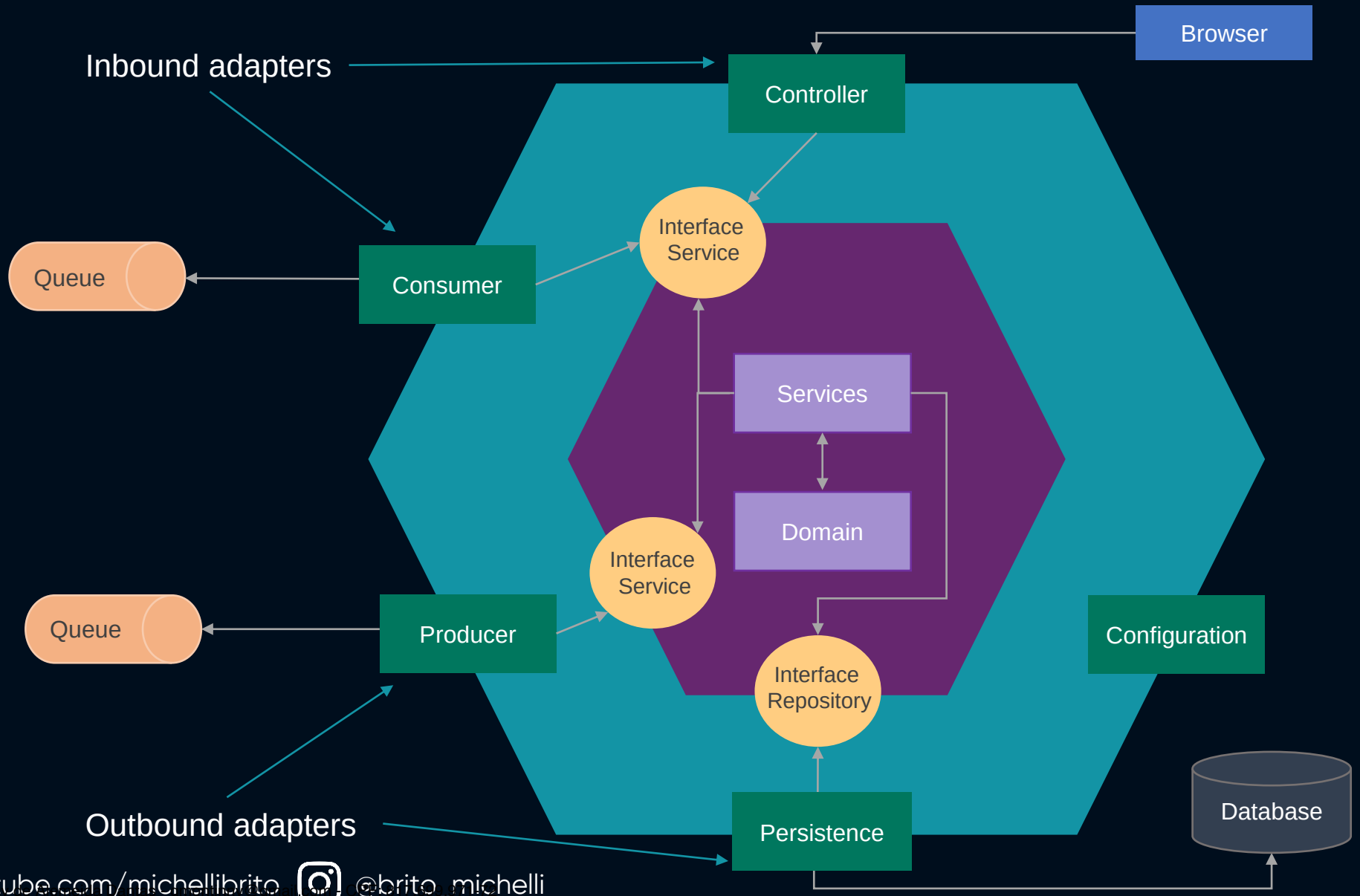


Ports and Adapters Pattern



Design de Negócio
Arquitetura
Hexagonal

Ports and Adapters Pattern



DESCOMPLICANDO

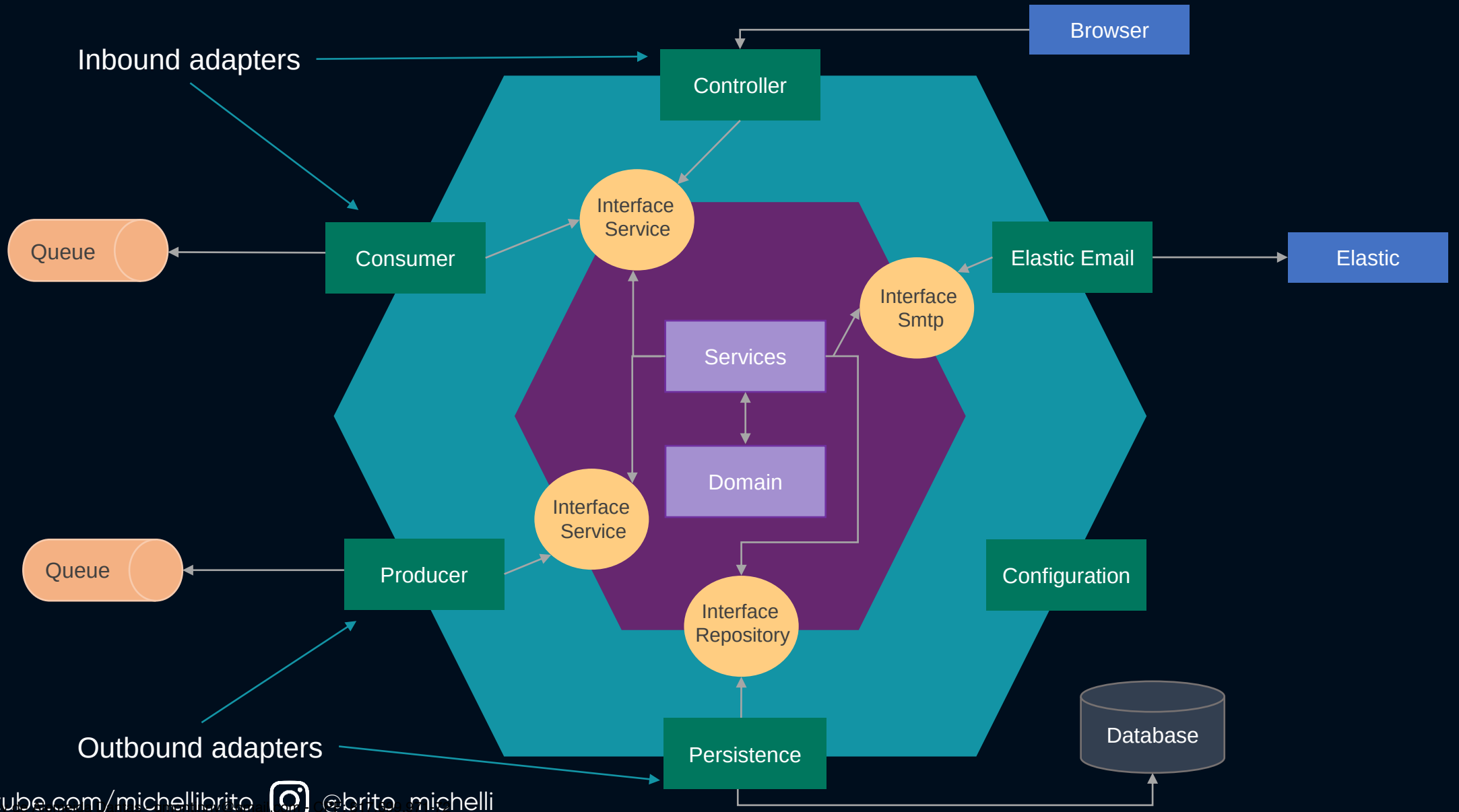
#decoderweek

Arquitetura Hexagonal trata-se da dissociação da lógica de negócio das tecnologias e frameworks, preocupando-se principalmente com a SUSTENTABILIDADE.

Ou seja, a capacidade e a facilidade de modificar, testar e modernizar softwares ao longo do tempo.

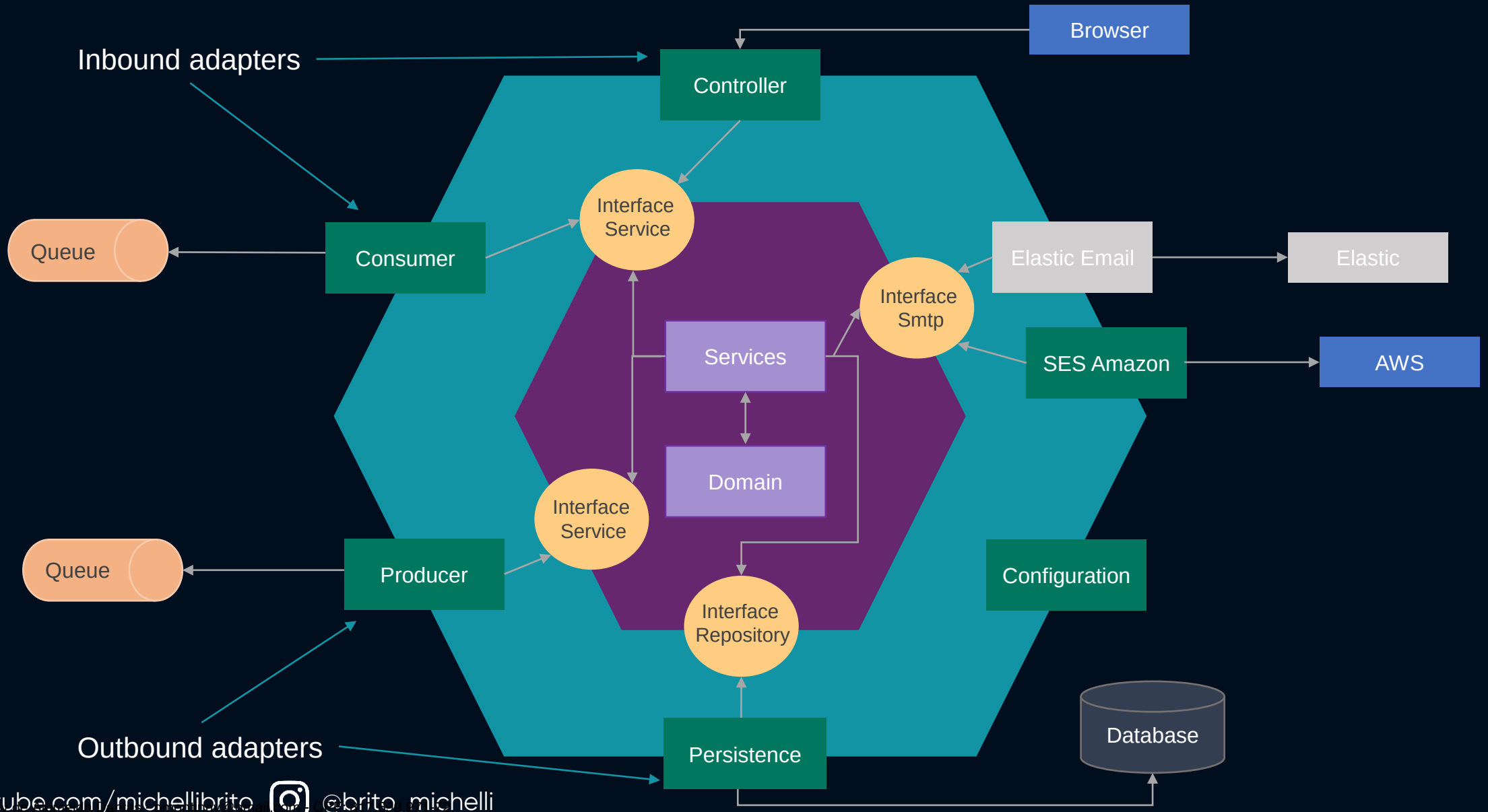
#decoderweek

Ports and Adapters Pattern



#decoderweek

Ports and Adapters Pattern



youtube.com/michellibrito @brito_michelli

Boas práticas e padrões em Microservices

- Broker Pattern e Mediator Pattern
- API Composition Pattern
- Event Notification Pattern
- Registry Discovery Pattern
- Event-Carried State Transfer Pattern
- Distributed Tracing Pattern
- Saga Pattern com Coreografia
- Log Aggregation Pattern
- Saga Pattern com Orquestração
- Ports and Adapters Pattern