

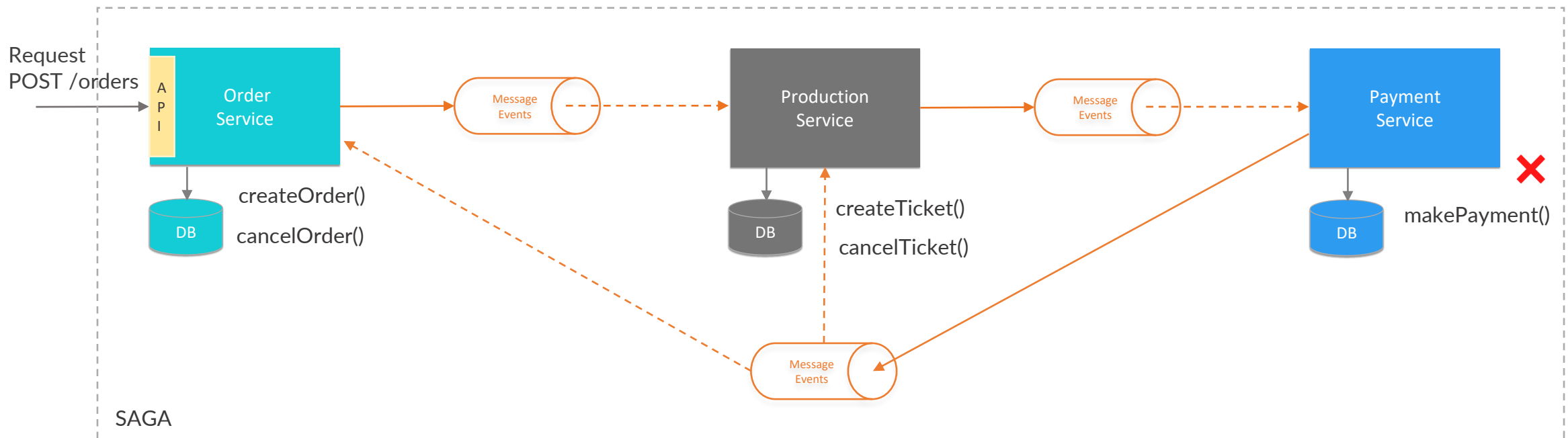
# Saga Pattern with Choreography vs Orchestration



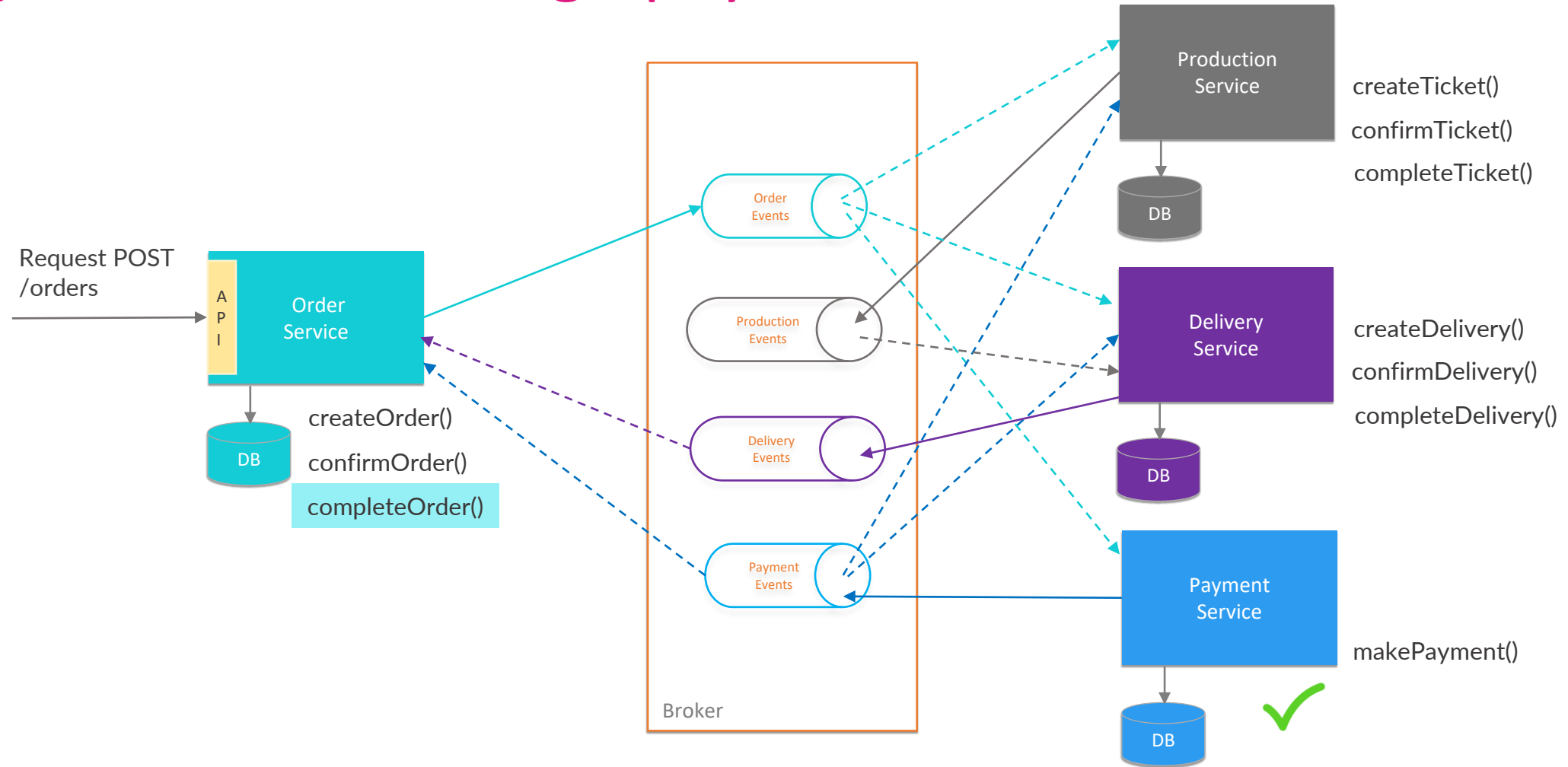
# Saga Pattern

Saga é uma sequencia de transações locais que abrangem vários Microservices quando trabalhamos com base de dados distribuídas na arquitetura, ou seja, ocorre uma transação local em um Microservice e em seguida é enviada uma mensagem ou evento para acionar as próximas transações nos demais Microservices.

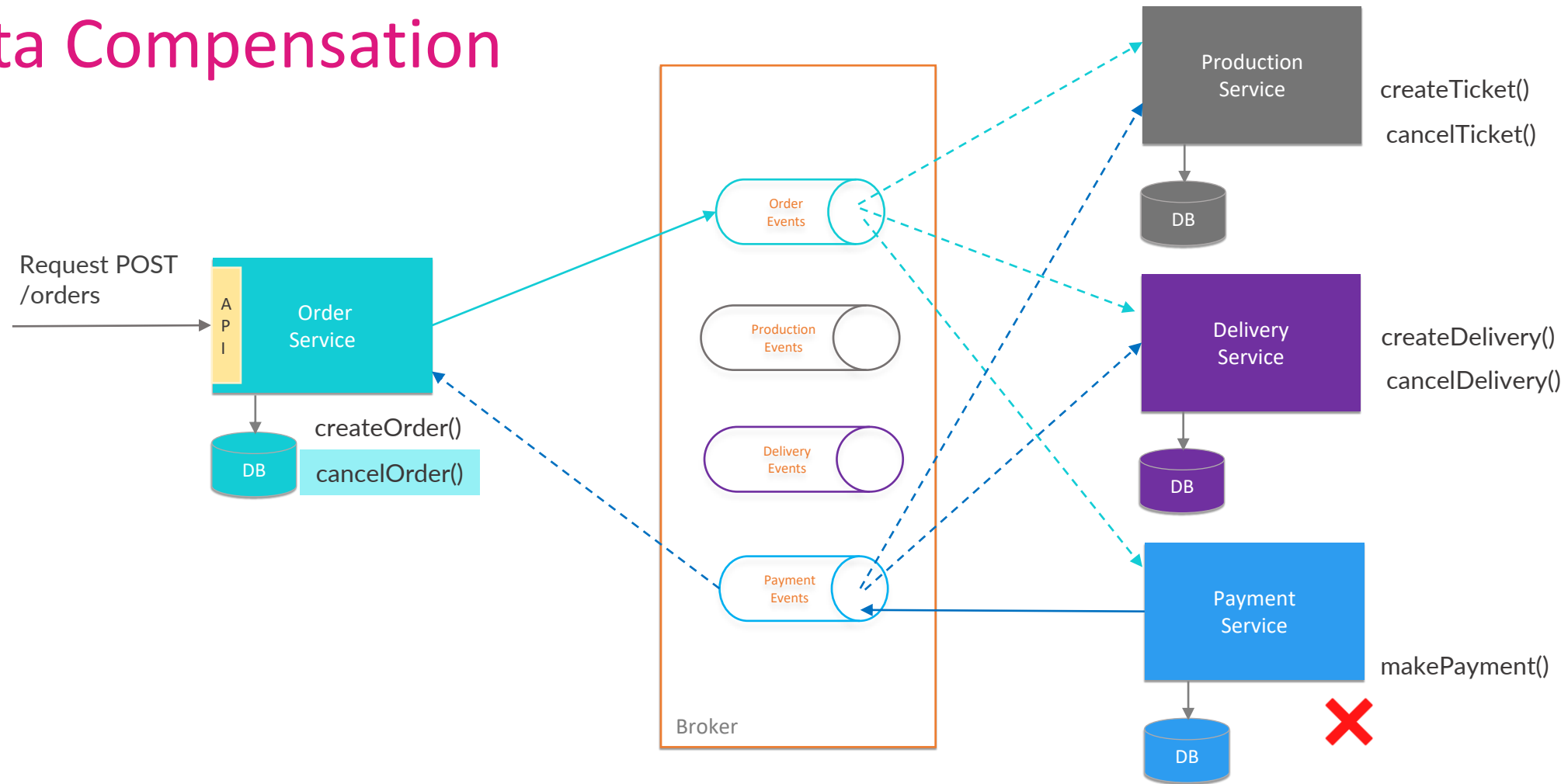
Se acaso uma dessas transações falham por algum motivo é necessário compensar as transações já realizadas anteriormente.



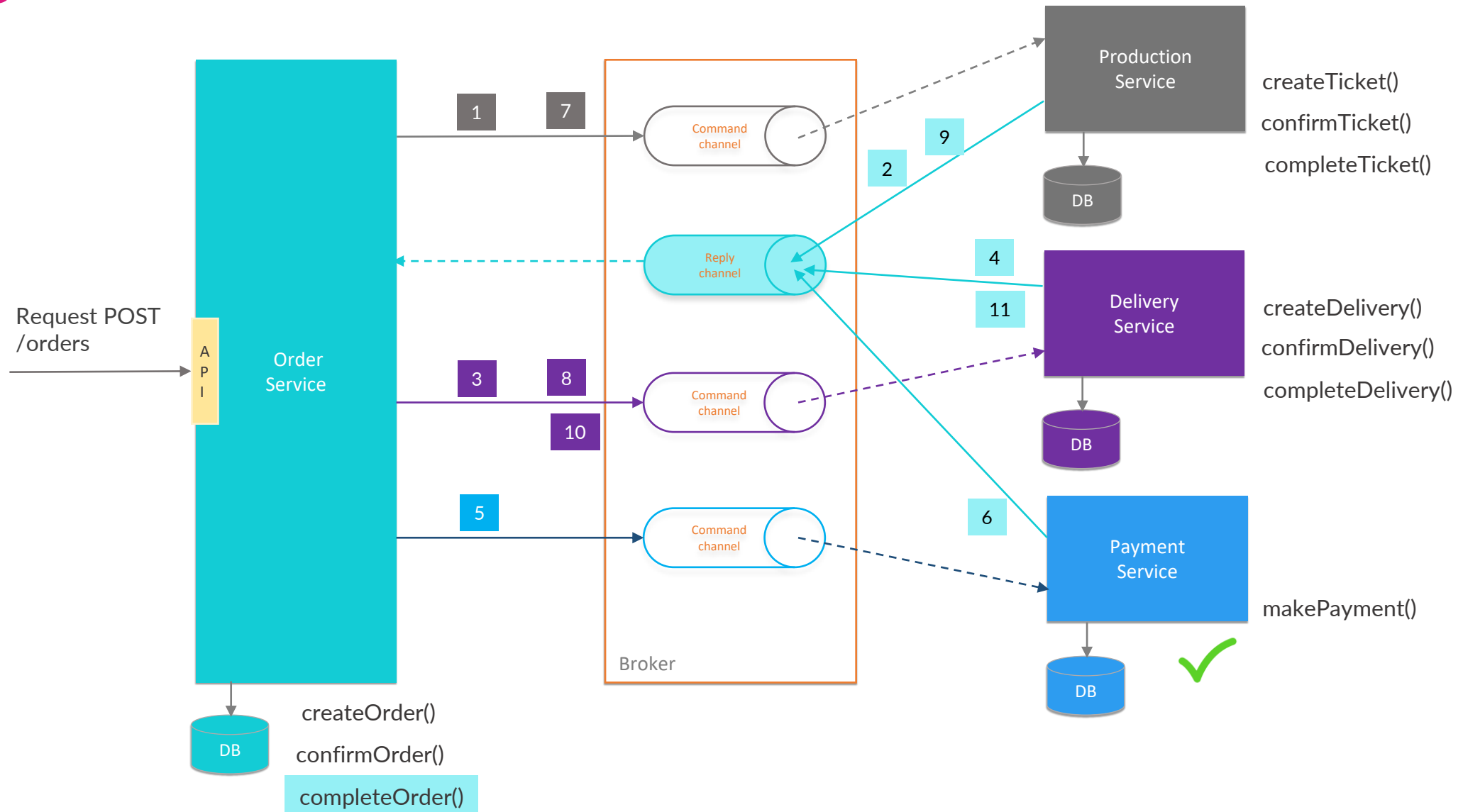
# Saga Pattern - Choreography



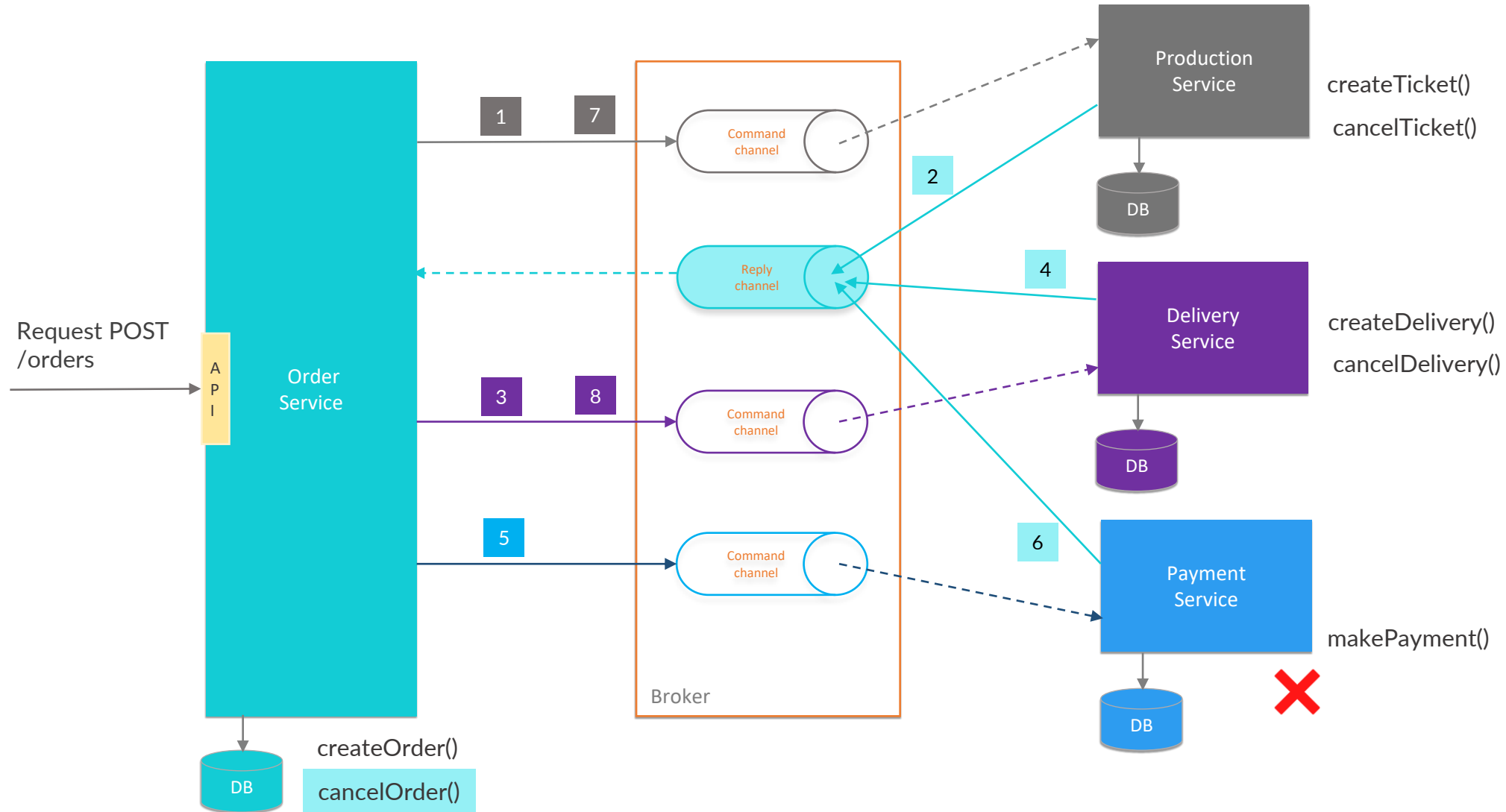
# Saga Pattern – Choreography and Data Compensation



# Saga Pattern - Orchestration



# Saga Pattern – Orchestration and Data Compensation



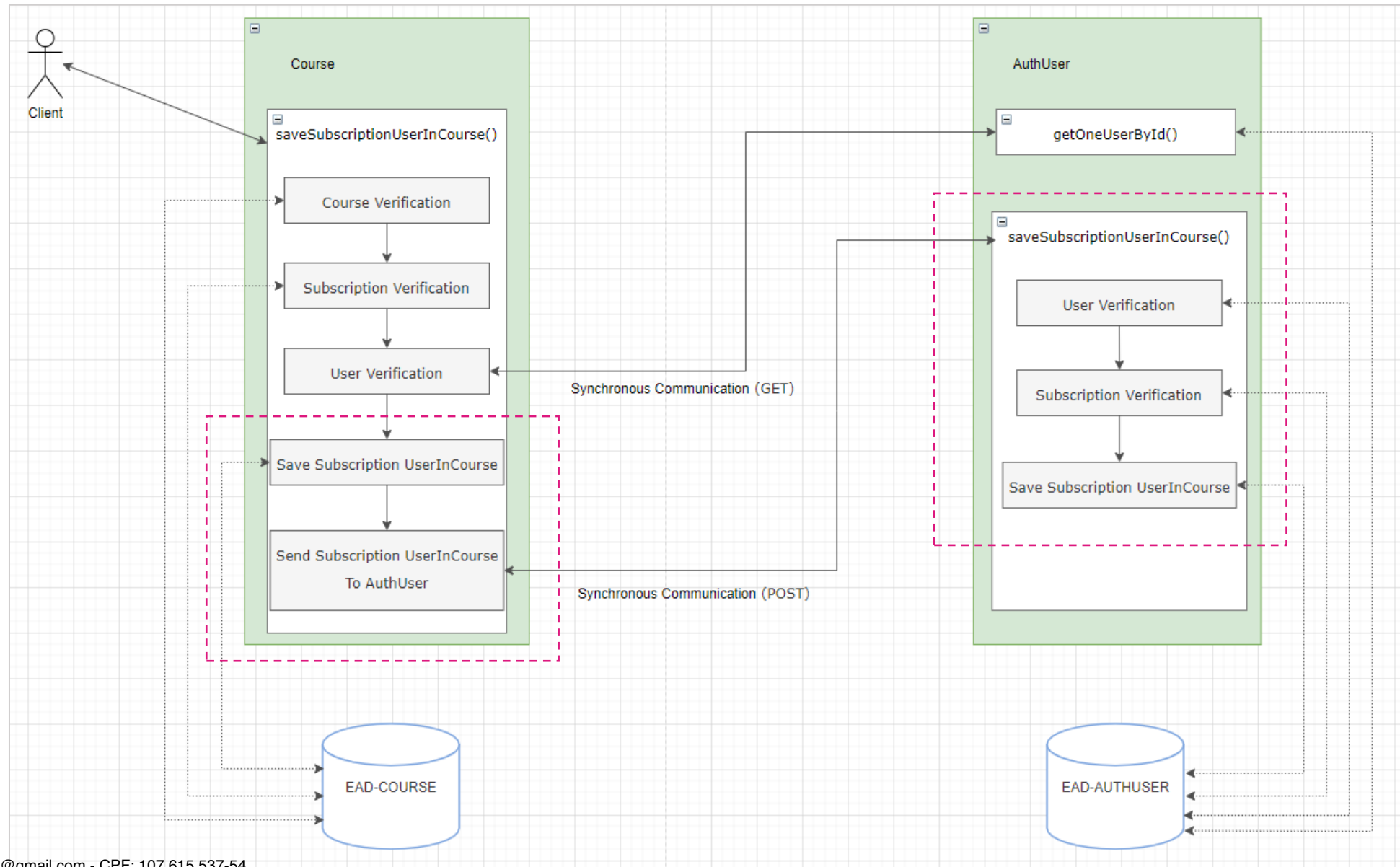
# Orchestration

- Benefícios:
  - Dependências mais simples (os Microservices apenas respondem, não invocam);
  - Melhora a divisão de responsabilidades;
  - Ideal para regras e lógicas dinâmicas que necessitam de controle e orquestração;
- Desafios:
  - Maior acoplamento em torno do orquestrador;
  - Gera um ponto de falha;
  - Regra de negócio muito centralizada no orquestrador.

# Choreography

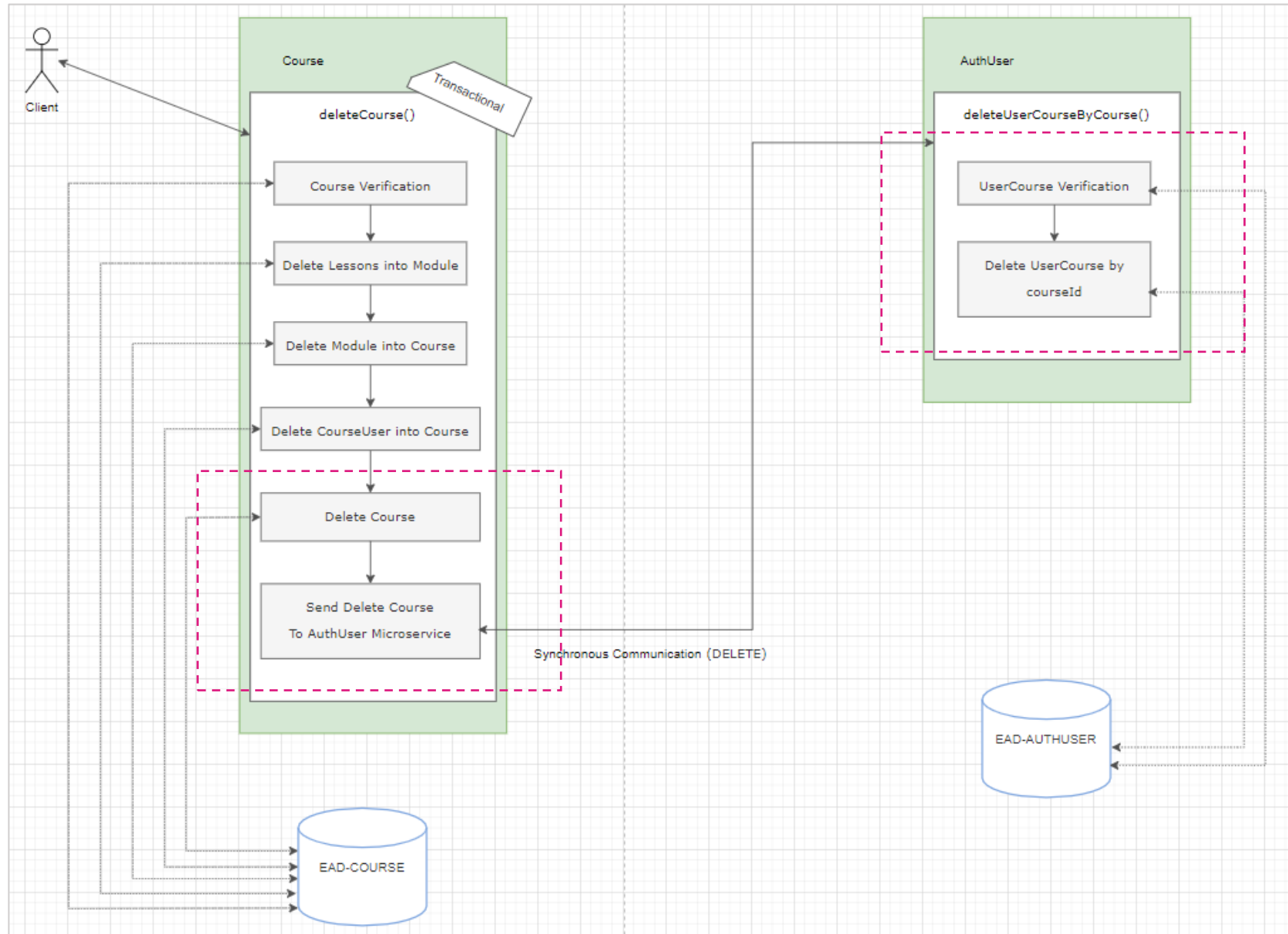
- Benefícios:
  - Simplicidade na implementação;
  - Baixo acoplamento;
- Desafios:
  - Maior dificuldade de entendimento da arquitetura;
  - Dependências cíclicas entre os Microservices;
  - Desdobramentos de eventos.

# Saga Pattern Example





# Saga Pattern Example



# Saga Pattern Example

