

# API Composition Pattern



# API Composition Pattern

Implementação de consultas que recuperam dados de vários Microservices, consultando cada um deles através de suas respectivas APIs, combinando esses resultados.

## Prós

Ideal para consultas em uma Arquitetura Microservices

Maneira simples de consultar e combinar dados de vários serviços

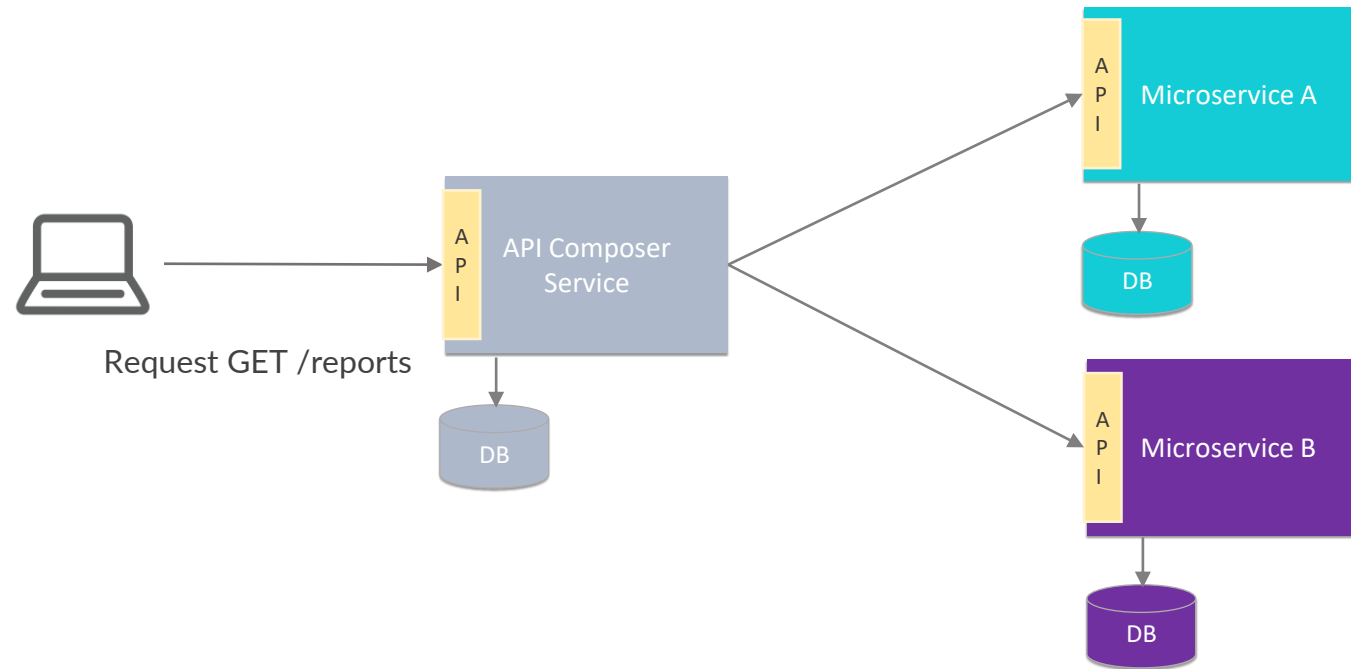
Diferentes maneiras de implementar esse Pattern

## Contras

Maior acoplamento entre os Microservices

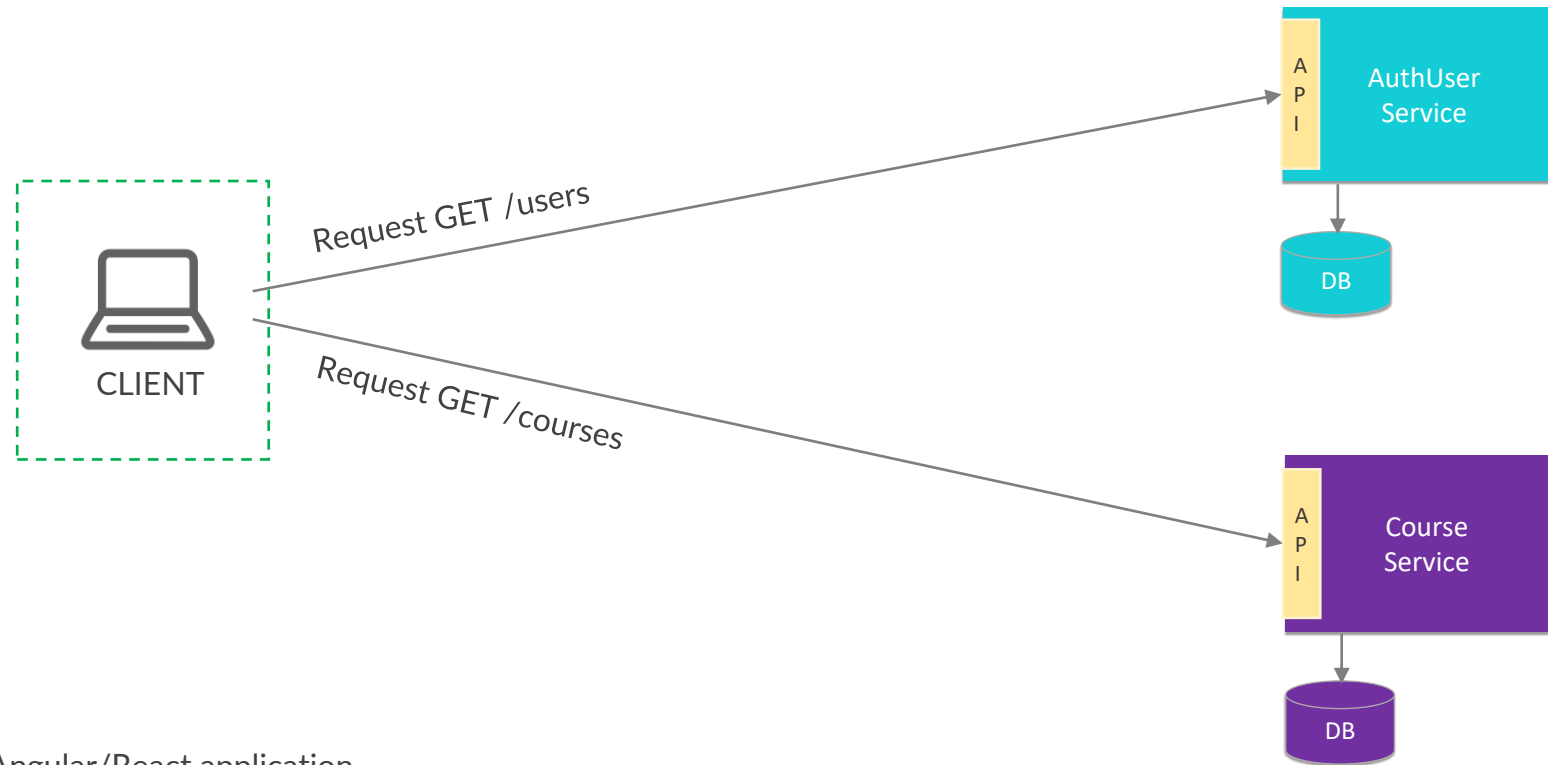
Disponibilidade reduzida

Consistência dos dados pode ser afetada



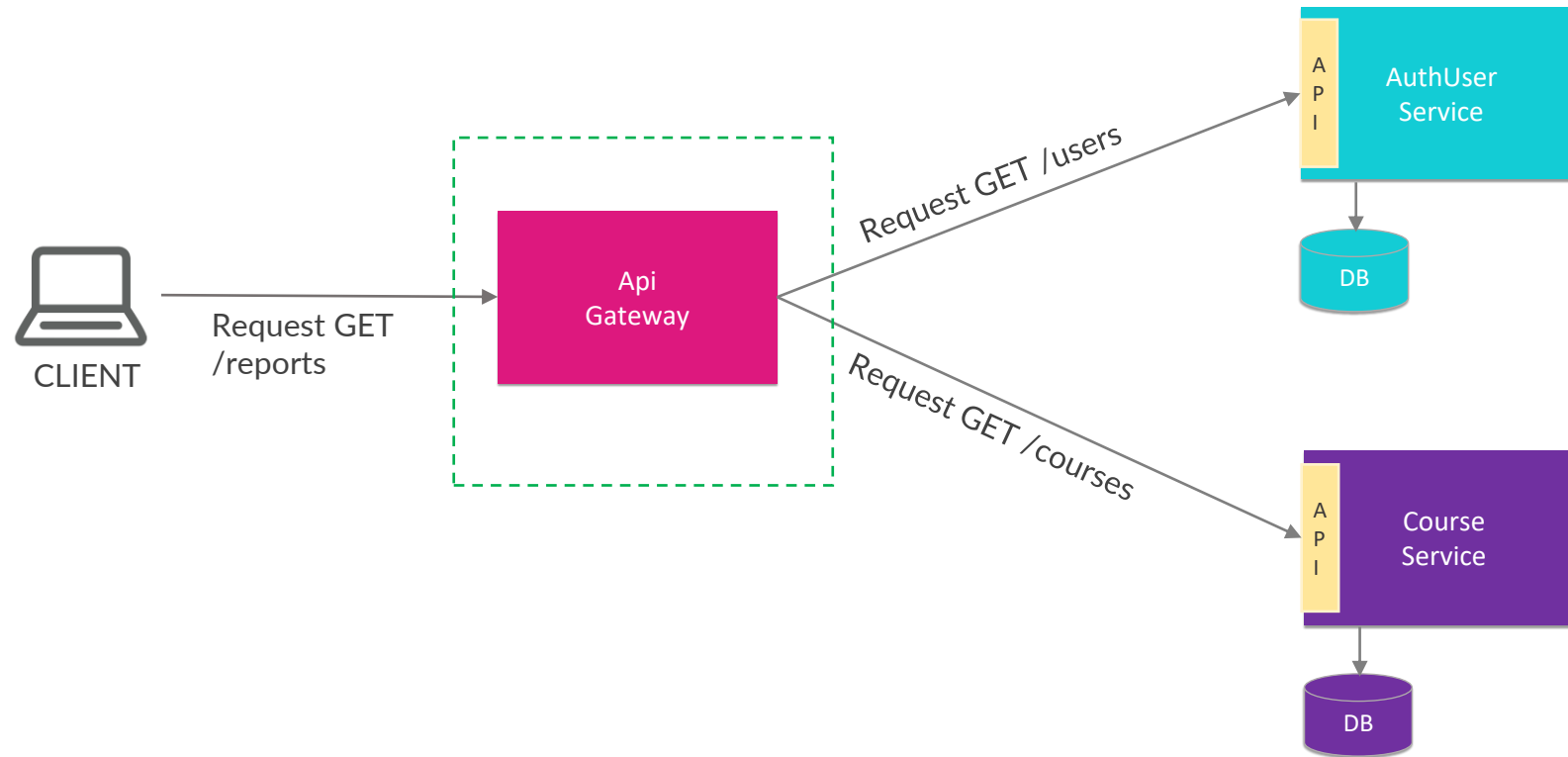
# How to implement API Composition Pattern?

# API Composition in a Client

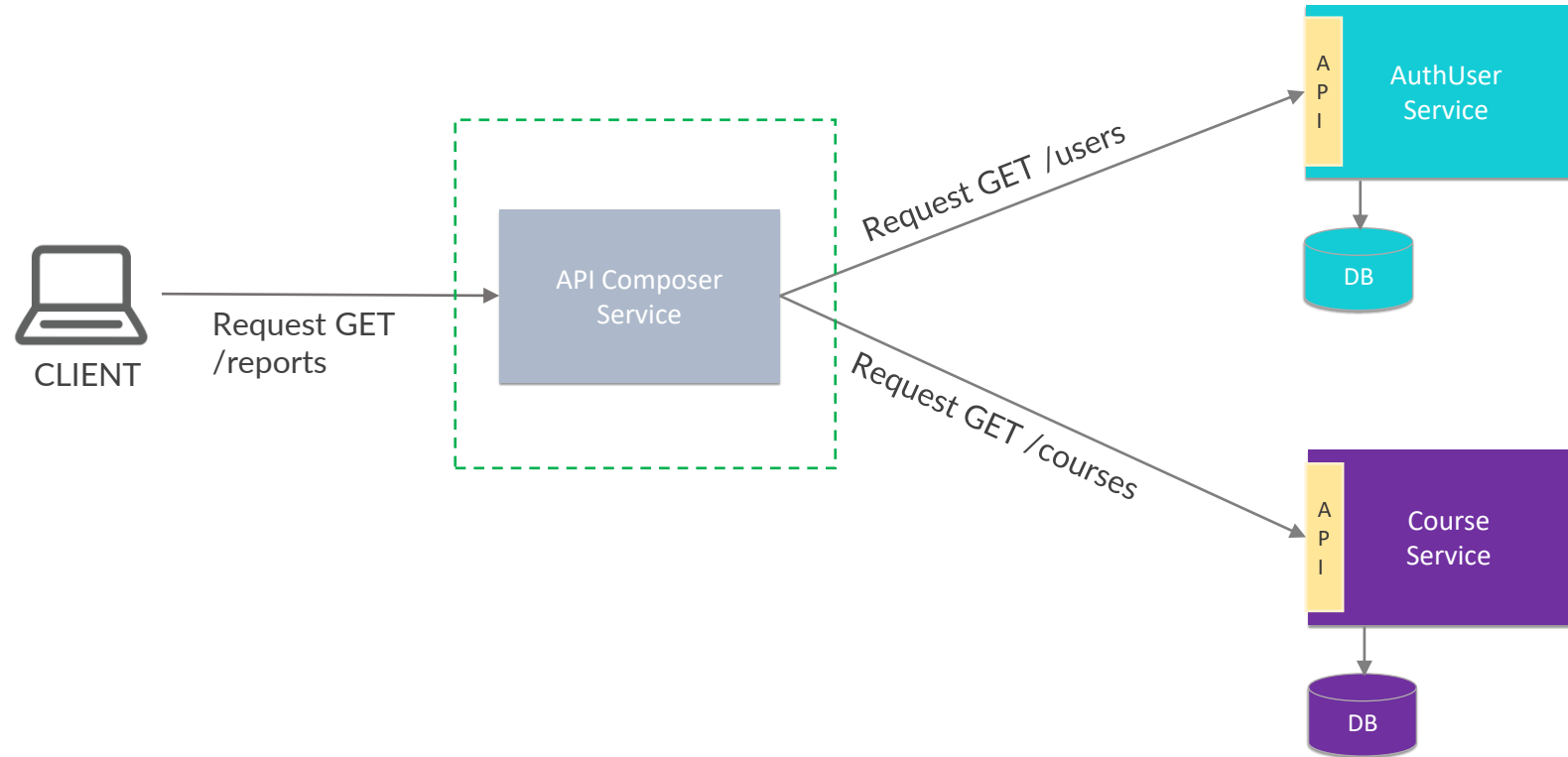


Client -> Angular/React application

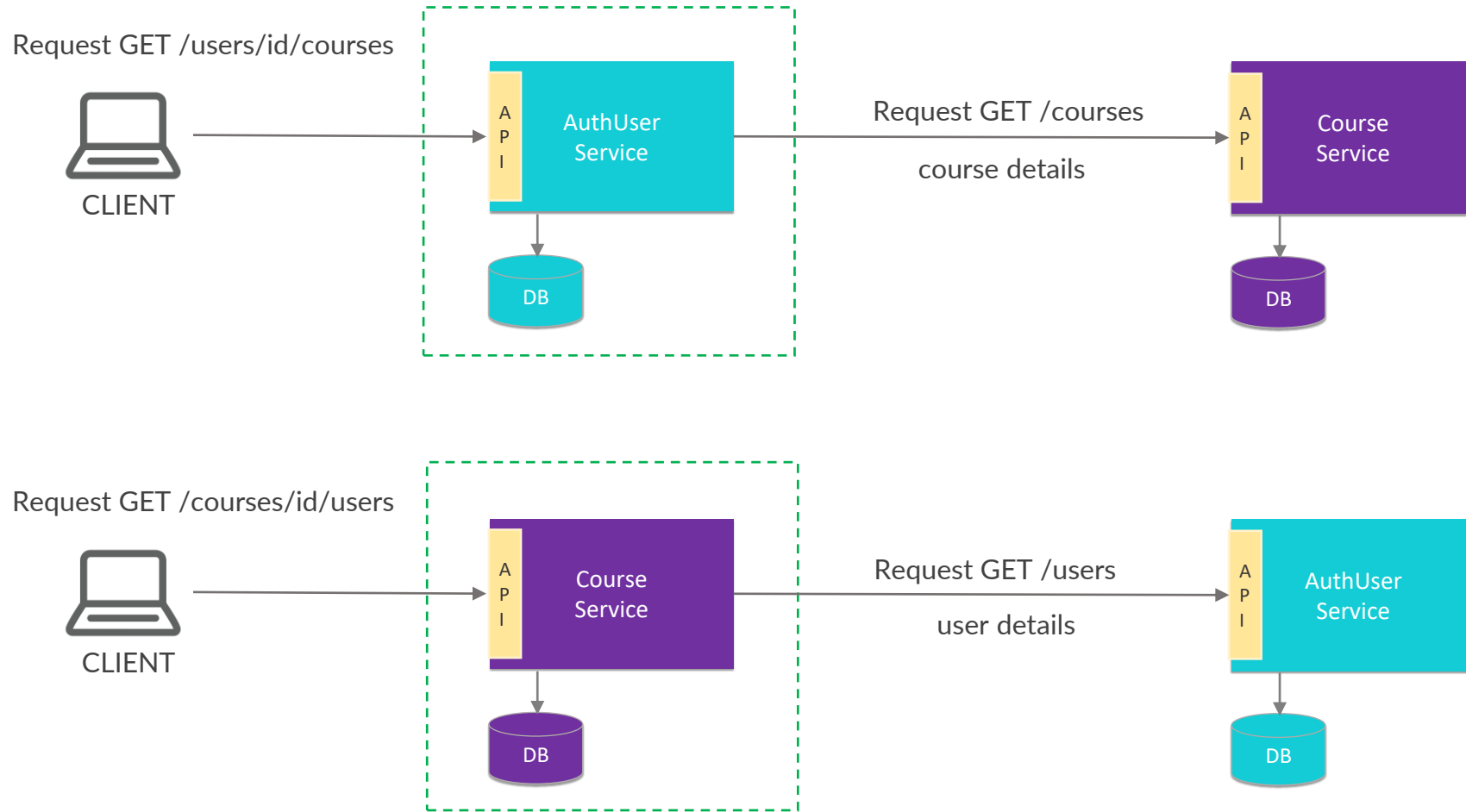
# API Composition in the API Gateway



# API Composition in Isolated Service Composer



# API Composition in a Service



# GET All Courses By User – API Composition

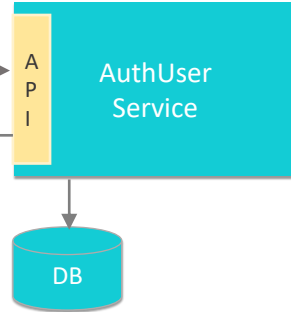
1

Request GET /users/userId/courses



CLIENT

Response



2

Request GET /courses?userId=id



Coleção de Courses em  
que um determinado User  
está inscrito

Relação courseId com  
userId

1

GET



http://localhost:8087/users/ed235441-89e3-4de5-87af-0cfb2ac8a9d8/courses

2

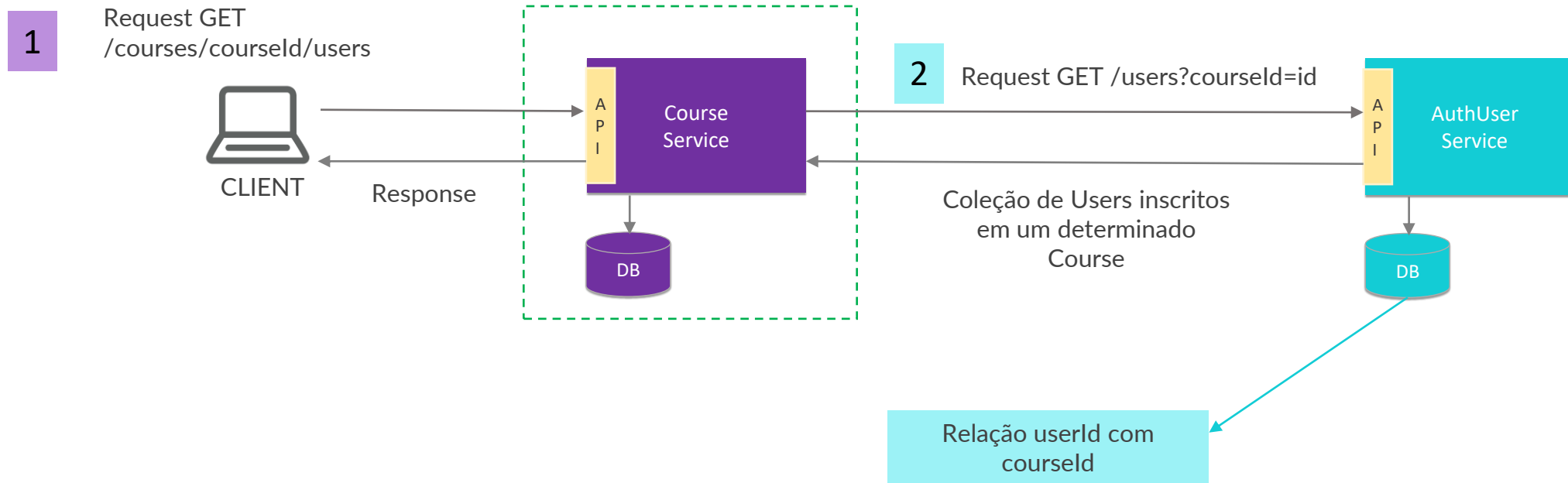
GET



http://localhost:8082/courses?userId=287ba167-44c9-4670-8dda-0404e0ff60db



# GET All Users By Course – API Composition



**1** GET <http://localhost:8082/courses/32e36682-8b0e-4816-b566-cb03b9e7ceb9/users>

**2** GET <http://localhost:8087/users?courseId=06c2985c-376d-43fe-bfe5-c586e0d6f233>

# Next Steps

- Criar mapeamento entre userId e courseId em AuthUser Microservice;
- Criar mapeamento entre courseId e userId em Course Microservice;
- Preparar métodos GET ALL (getAllUsers e getAllCourses) para receber os parâmetros courseId e userId respectivamente;
- Implementar API Composition através da comunicação síncrona via HTTP REST entre os Microservices para obter os detalhes necessários dos recursos e retornar para o cliente.