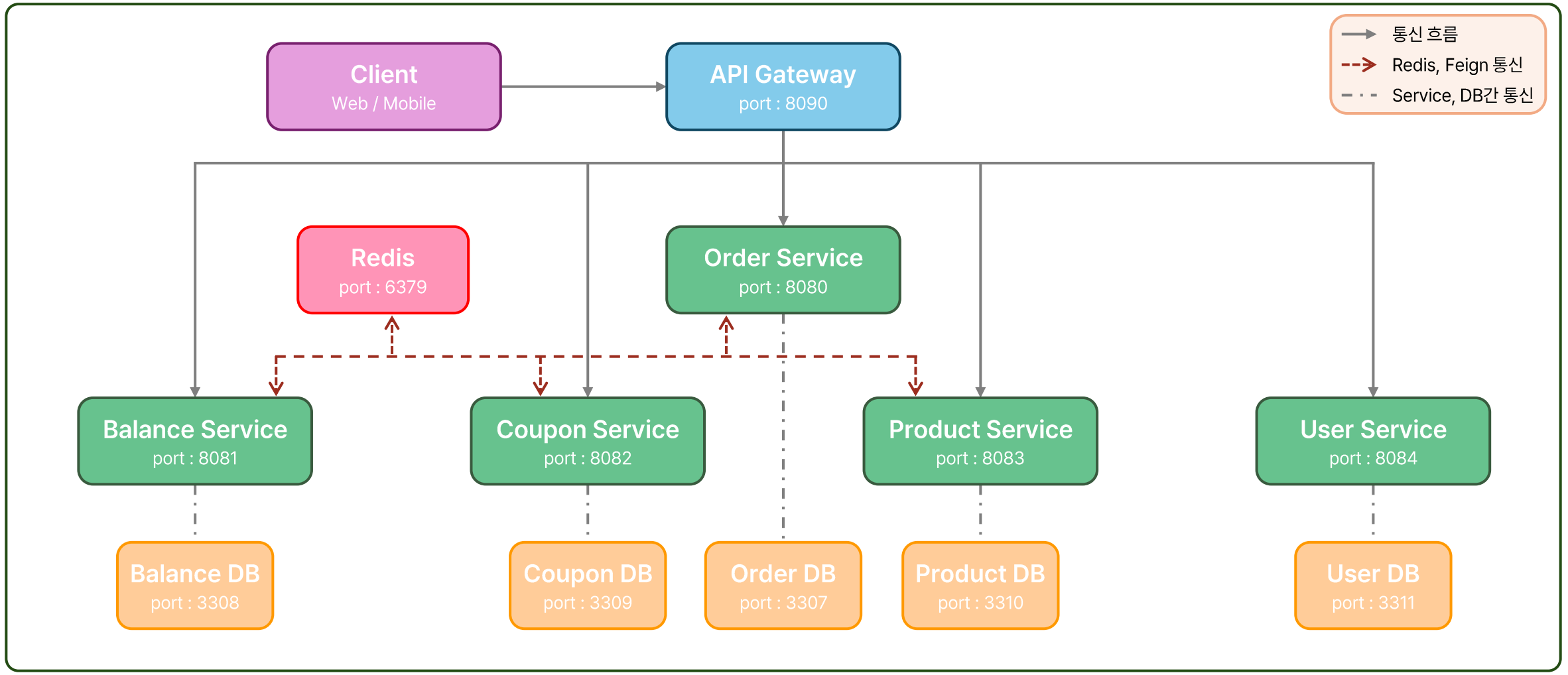
03

프로젝트 전체 아키텍처

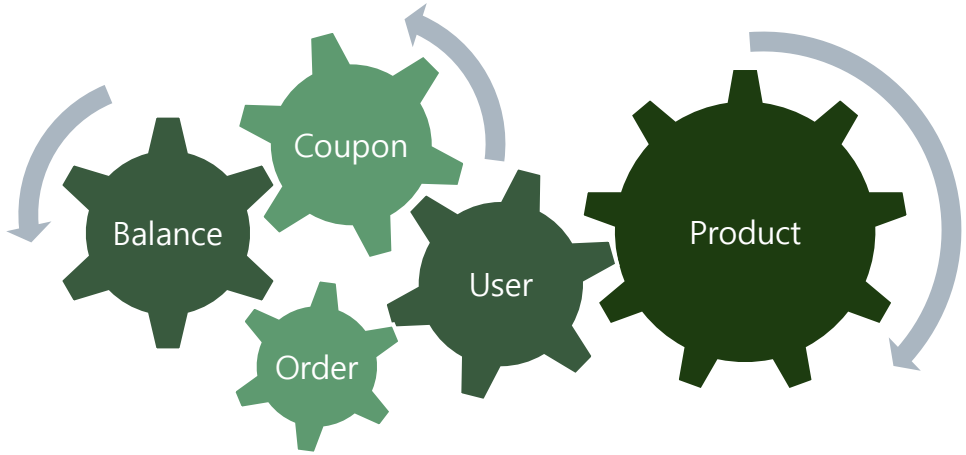
이커머스 서비스의 전체 아키텍처를 통해 서비스 구성 요소와 흐름에 대해 파악하고, 데이터 흐름을 명확히 정의함으로써 효율적인 개발 및 운영 체계를 구축하였습니다.



02

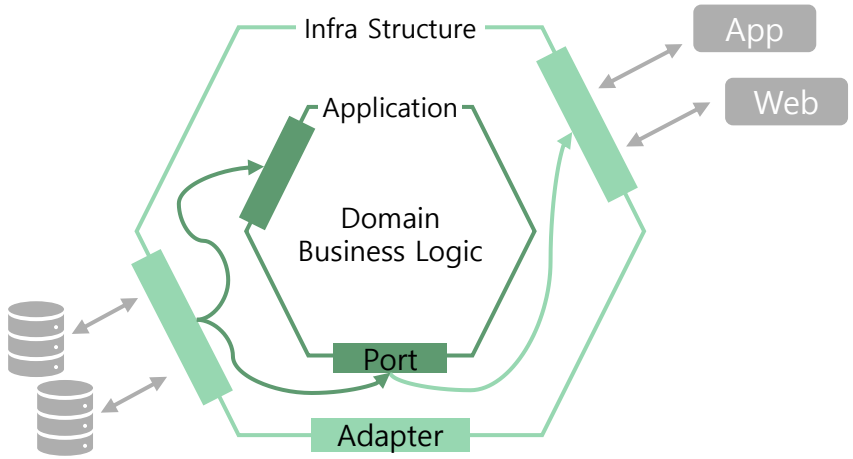
아키텍처 및 폴더 구조

MSA(Micro Service Architecture) 아키텍처를 통해 각 서비스별 기능을 분리하였으며, 헥사고날 아키텍처를 사용하여 폴더 구조를 체계적으로 관리하였습니다.



MSA

```
ecommerce-project/
├── balance-service/
├── coupon-service/
├── order-service/
├── product-service/
├── user-service/
├── common/
└── api-gateway/
```



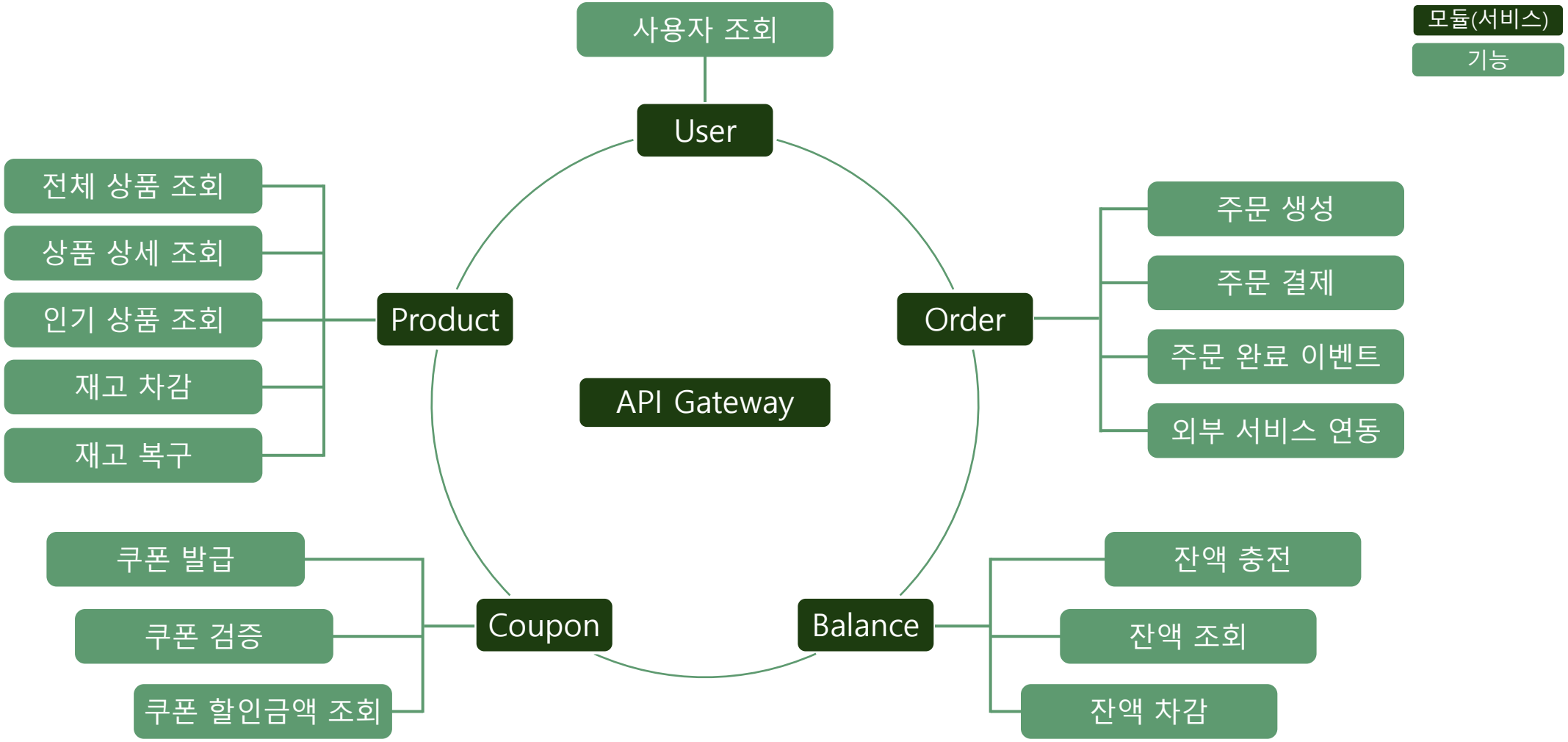
헥사고날

```
balance-service/
├── java/com/hhplus/ecommerce/balance/
│   ├── adapter/
│   │   ├── in/
│   │   │   └── web/
│   │   └── out/
│   │       └── persistence/
│   ├── application/
│   │   ├── port/
│   │   │   ├── in/
│   │   │   └── out/
│   │   └── service/
│   └── domain/
└── resources/
```

03

주요 기능 및 구현

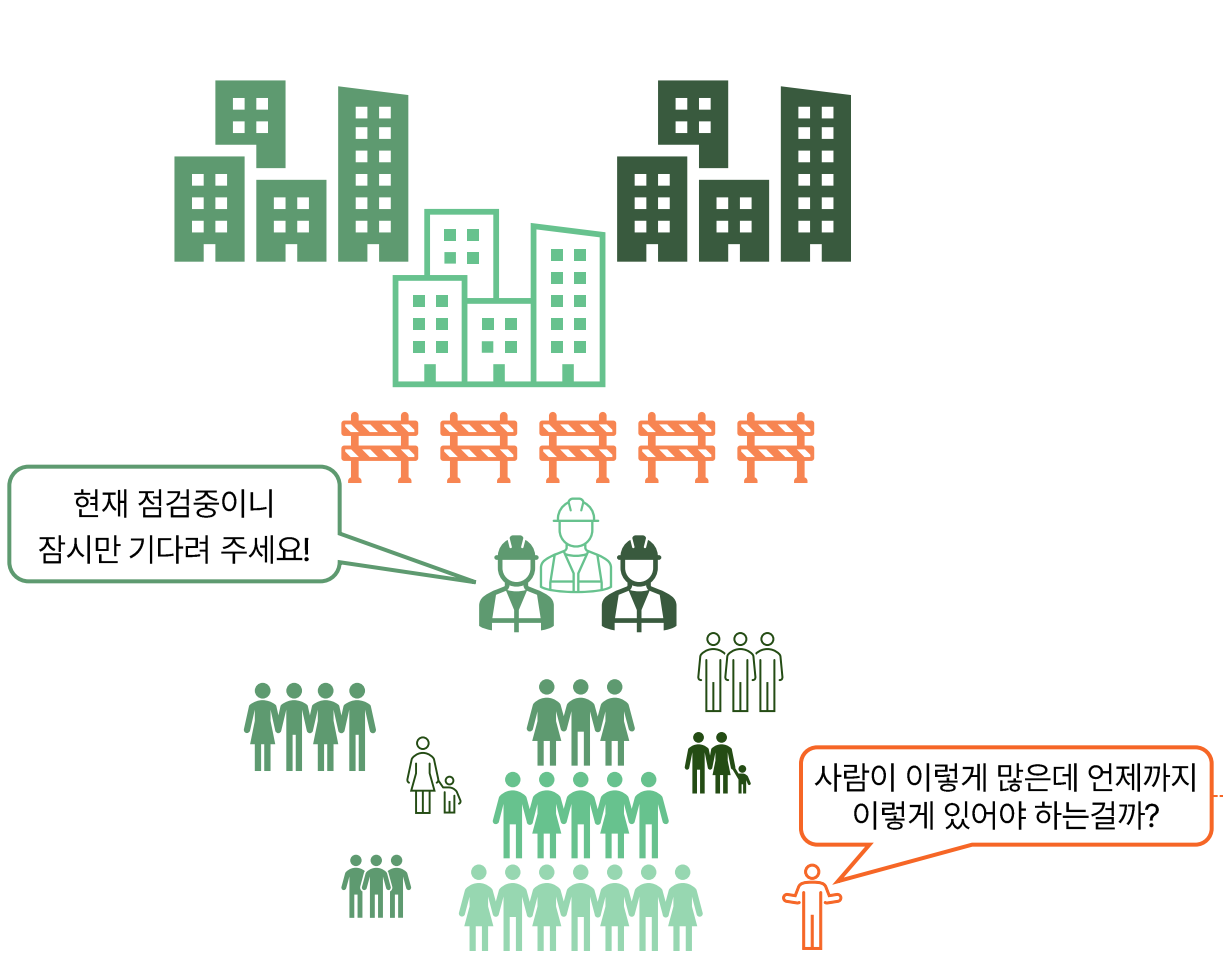
이커머스 서비스에서 필수적으로 요구되는 기능을 구현하여 서비스 이용에 문제 없도록 하였으며 서비스를 운영하며 고객의 요구에 맞게 서비스를 확장할 수 있도록 하였습니다.

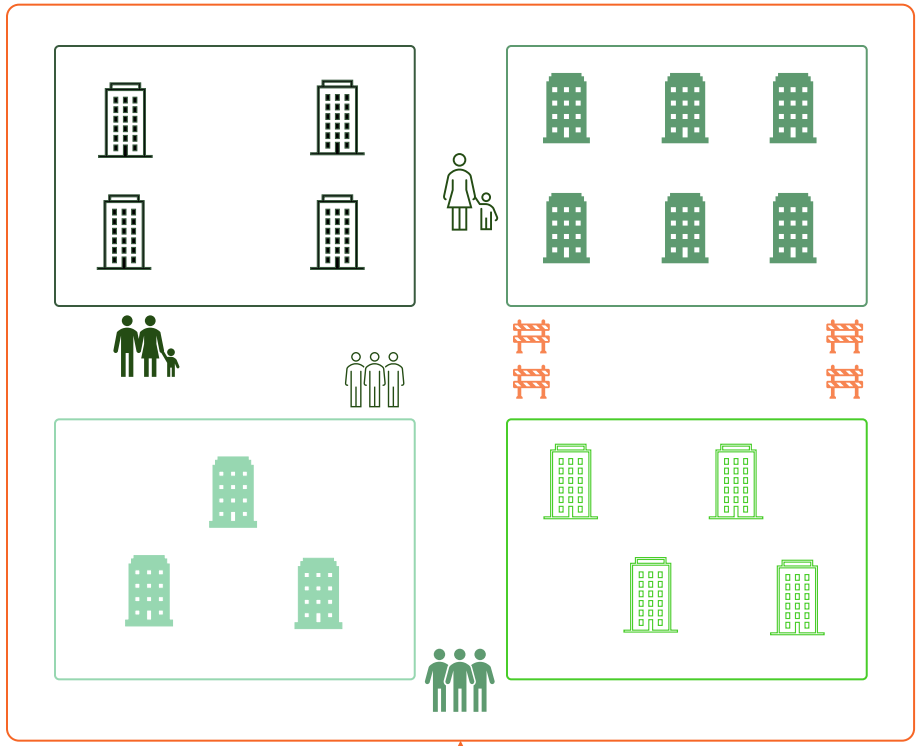


02

MSA 아키텍처로의 전환(1/2)

서비스 이용자 증가에 대비하여 기존 Monolithic Architecture(모놀리식 아키텍처)에서 Micro Service Architecture(마이크로서비스 아키텍처)로 전환하였습니다.



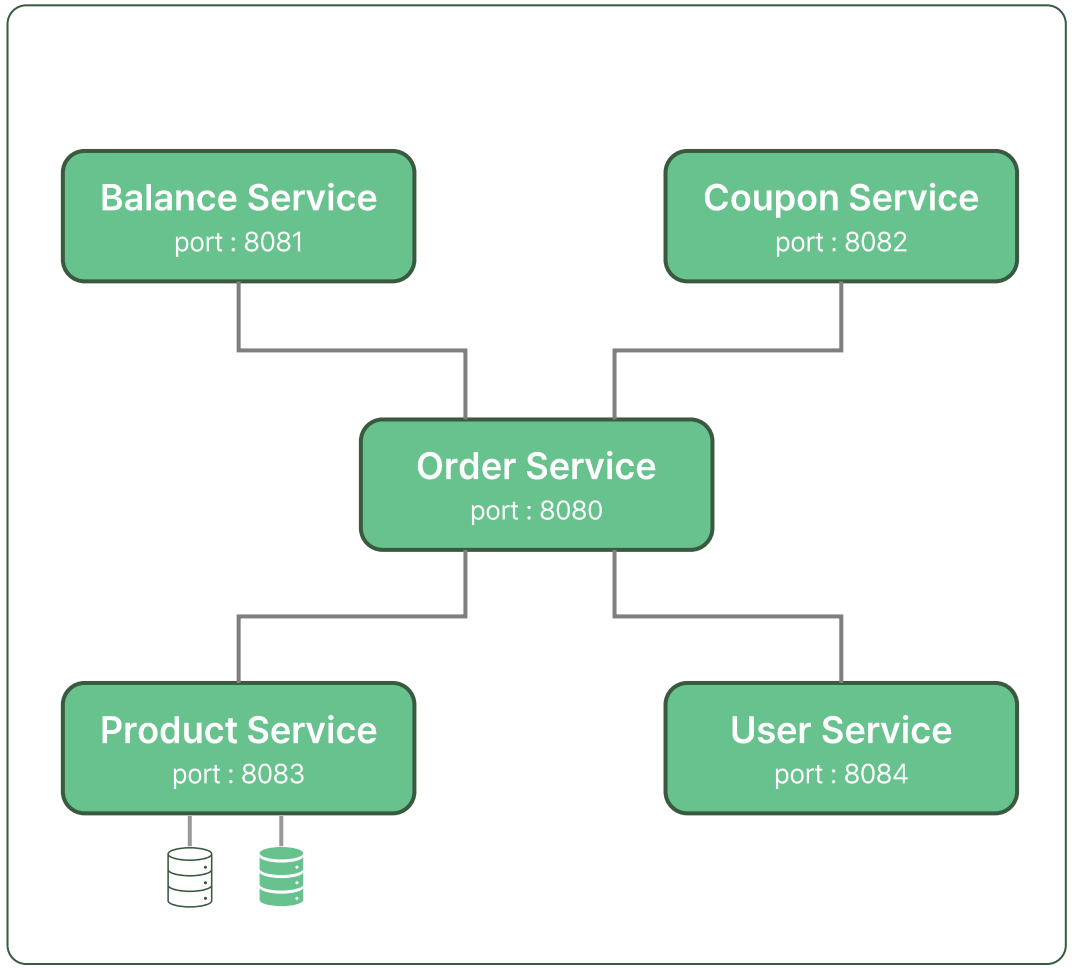


이용자가 증가하더라도 서비스 운영은 계속 되어야 하기에
Micro Service로 전환하여 이용자 불편을 최소화

02

MSA 아키텍처로의 전환(2/2)

서비스 단위가 작아짐에 따라 오케스트레이션 기반의 SAGA 패턴을 적용하여 분산 트랜잭션의 데이터 일관성을 확보하고 CQRS를 적용하여 각 서비스에 최적화된 성능을 제공할 수 있도록 하였습니다.



변경 내용

- ✓ 이커머스 서비스를 구성하는 5개의 영역으로 분리
- ✓ 오케스트레이션 방식의 SAGA 패턴이 적용된 Order Service를 중앙에 배치하여 데이터 일관성 확보
- ✓ 인기상품 조회의 경우 단순히 조회 기능만 있으면서 많이 사용되는 기능
- ✓ 상품에 대해서 쓰기 기능이 사용되는 빈도는 읽기 기능에 비해 적음
- ✓ Product Service의 경우 CQRS를 적용하여 대규모 트래픽에 효율적으로 대용

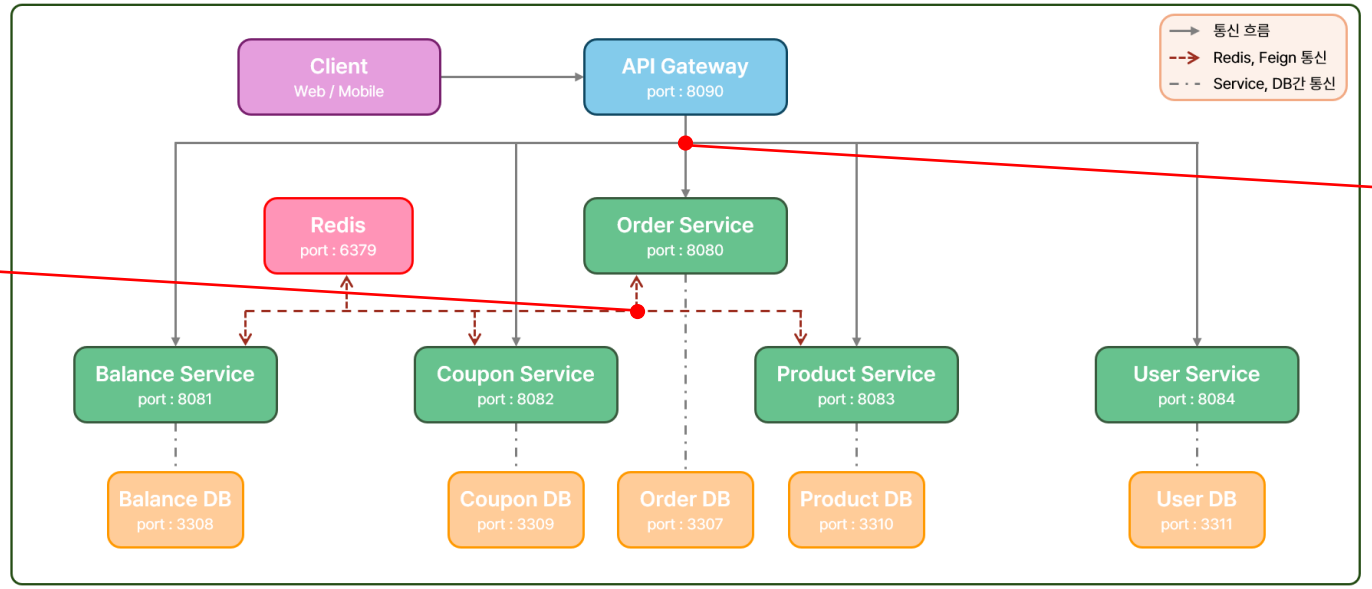
04

장애 대응 및 개선 방안

이커머스 서비스 아키텍처를 이용하여 트랜잭션 처리의 한계를 확인하고 대응 방안을 설계 하였습니다.

(5주차때 구현한 SAGA와 CQRS의 경우 1주마다 새로운 내용을 적용하며 진행하기 어려울 것 같아 제외하였습니다.)

서비스가 분리되면서 데이터 일관성 문제가 발생할 수 있습니다.



서비스간 통신 중 네트워크 장애에 따른 불확실성에 의해 중복 처리나 누락 처리의 위험성이 있습니다.

대응 방안 1

SAGA Pattern 적용

- 중앙 관리(Orchestration)방식으로 전체 진행과정을 담당하고 문제가 발생하게 될 경우 이전단계까지 진행했던 내용을 취소

대응 방안 2

보상 트랜잭션 로직 추가로 데이터 정합성 강화

- 이벤트 발행으로 관심사가 분리되어 있으므로 실패하게 될 경우 보상 트랜잭션 이벤트를 발행