대규모 엔터프라이즈 시스템 개선 경험기

2부. 새 술 담을 새 부대 마련하기

김선철

김선철

- 18년도 1월 네이버 쇼핑 신입 입사
- 18년도 입사 후 스스로 만든 많은 장애 경험
- 19년도 팀의 격변기 극복
- 19년도 새로운 시스템 도전
- 이후로 여태껏 삽질 중
- 네이버 쇼핑 6년차

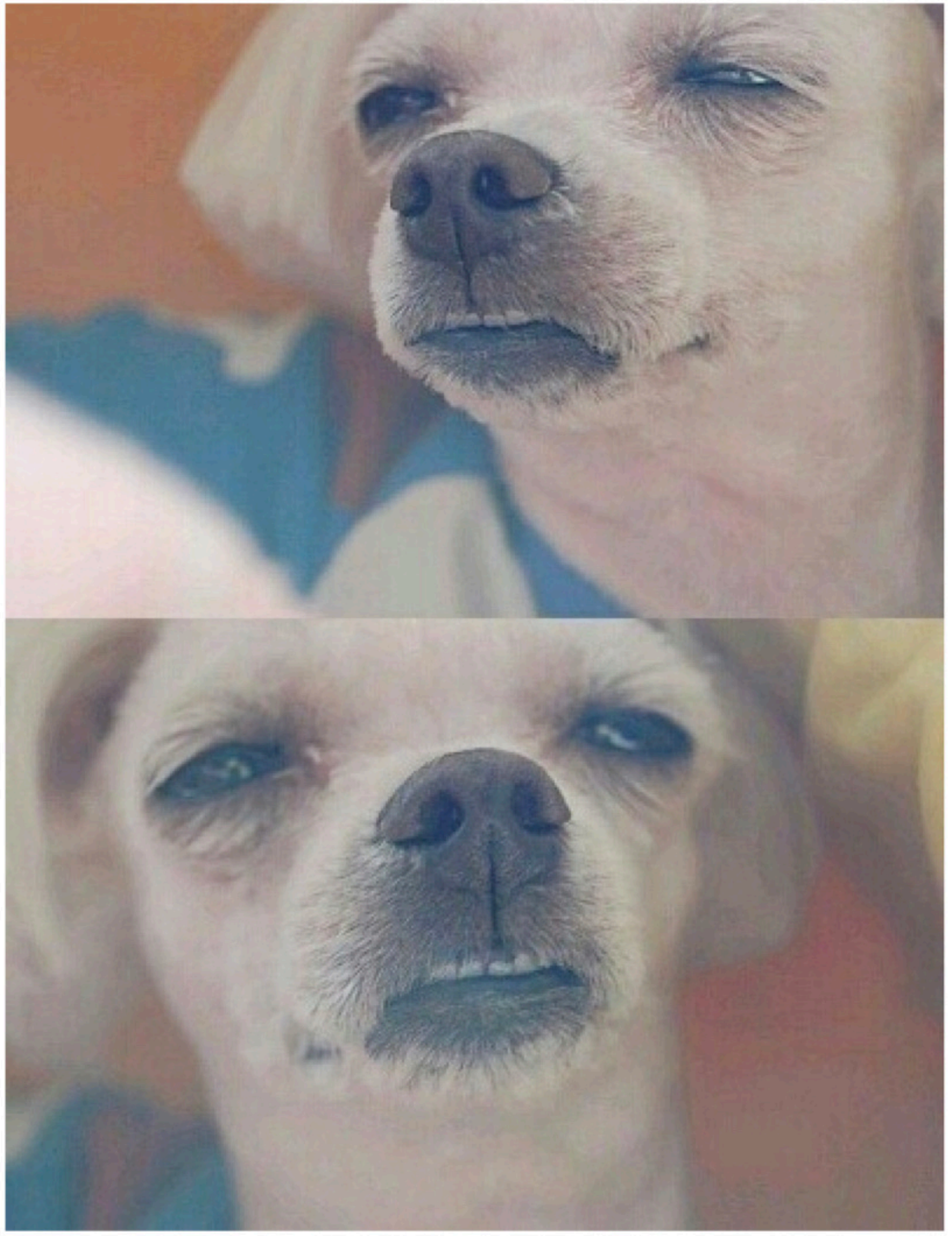
순서

- **주니어 시절의 경험**
  - 공통 모듈
  - 모호한 역할
  - 공유 데이터베이스
- **이론 학습**
  - Port & Adapter
  - Event Driven
  - Domain Driven
- **새로운 시스템 구성**
  - 패키지 구성
  - 이벤트 구성
  - 도메인 구성
  - 최종 구성
  - 소감
- **어려웠던 점들**
  - 책임, 역할 할당의 어려움
  - 의존성 해소의 어려움
  - Event의 흐름 정리


귀여운 햇병아리

새로운 것들 학습에 대한 재미

쉽지 않았던 주니어 시절..

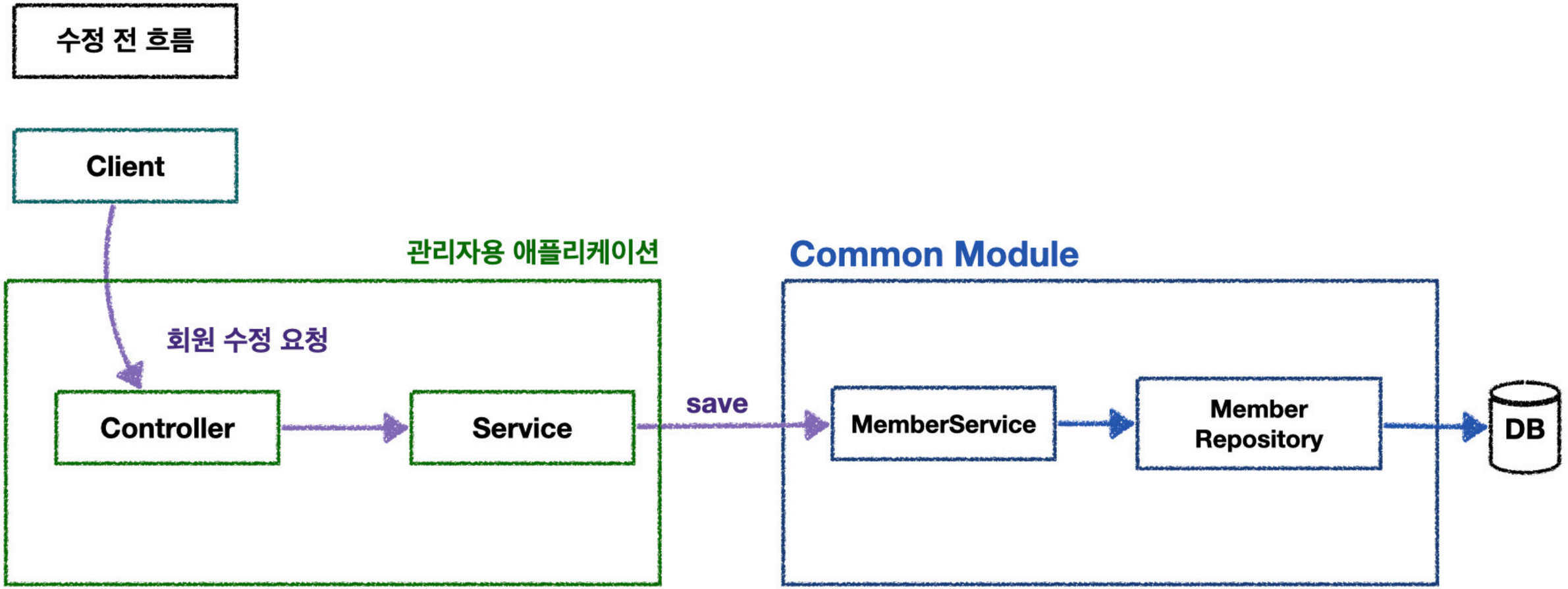


공통 모듈

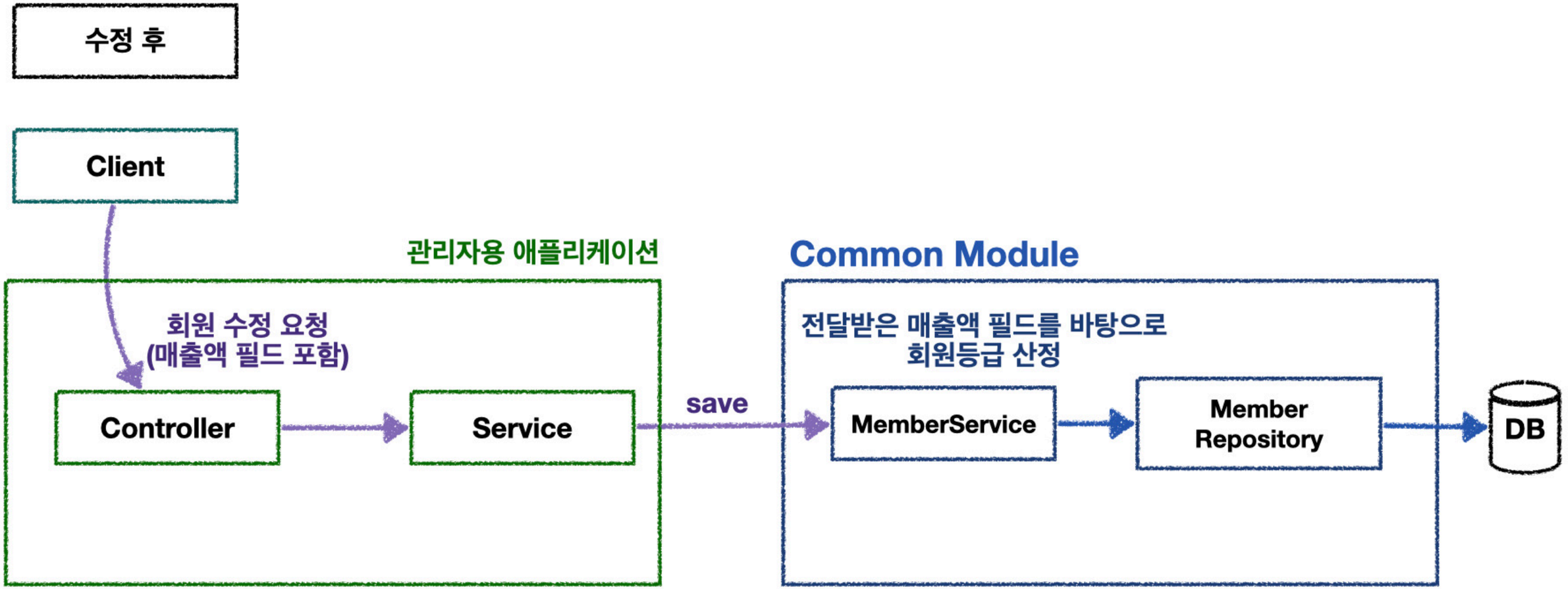
요구사항

- 관리자 화면에서 매출액을 기반으로 회원등급을 계산하여 저장하라

공통 모듈


수정 전 흐름
Client
관리자용 애플리케이션
회원 수정 요청
Controller
Service
save
Common Module
MemberService
Member Repository
DB

공통 모듈

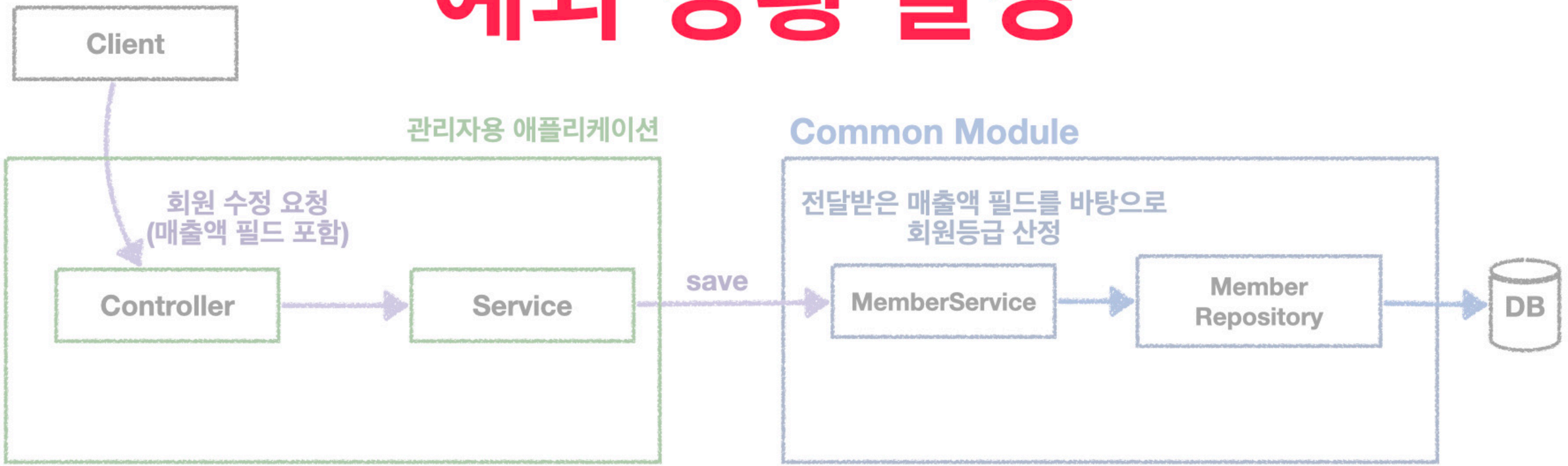

수정 후
Client
관리자용 애플리케이션
회원 수정 요청
(매출액 필드 포함)
Controller
Service
save
Common Module
전달받은 매출액 필드를 바탕으로
회원등급 산정
MemberService
Member
Repository
DB

공통 모듈의 편리성

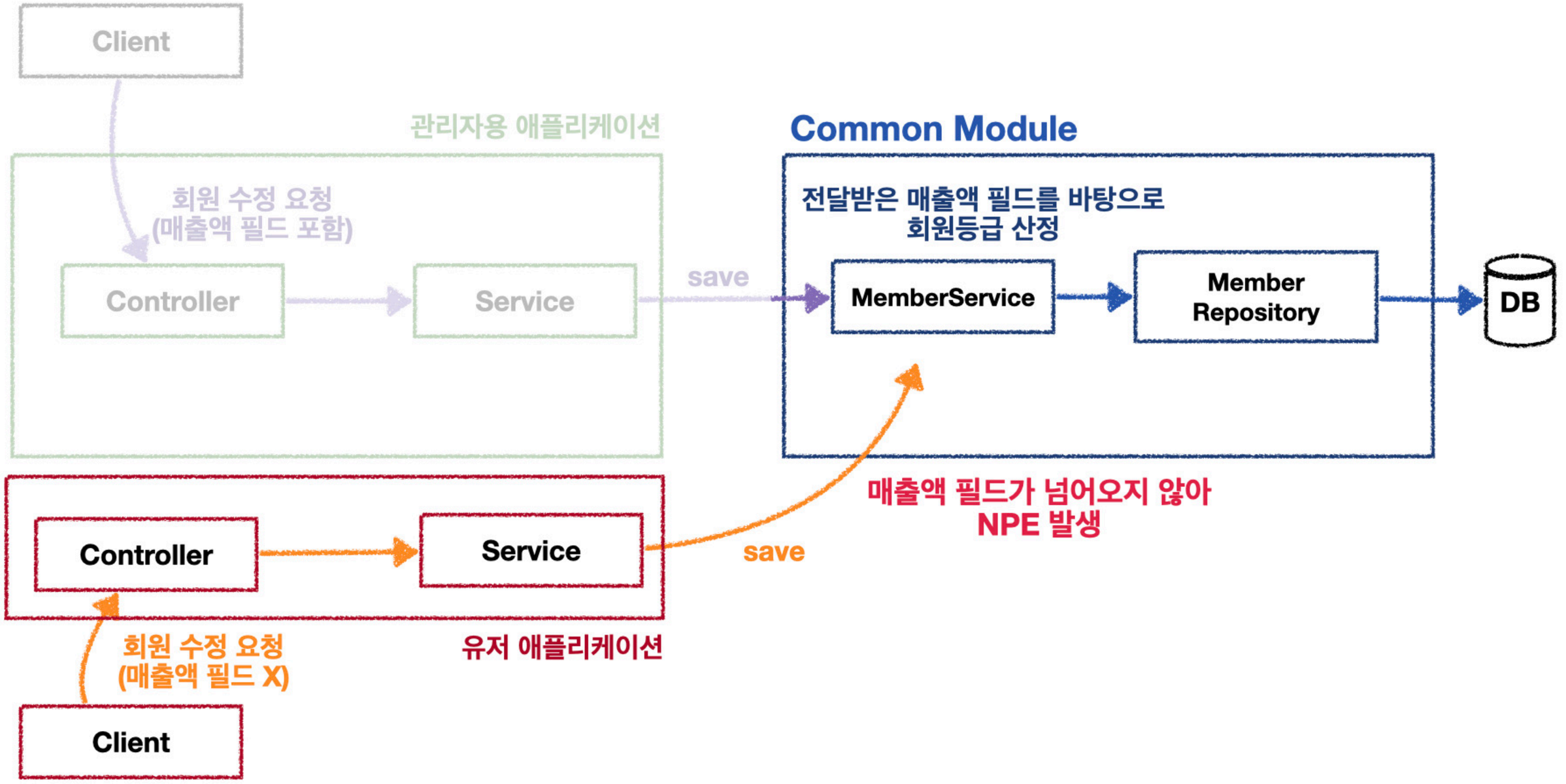
- 코드의 재사용 가능
- 중복 코드 방지 가능

공통 모듈

예외 상황 발생


Client
관리자용 애플리케이션
회원 수정 요청
(매출액 필드 포함)
Controller
Service
save
Common Module
전달받은 매출액 필드를 바탕으로
회원등급 산정
MemberService
Member
Repository
DB

공통 모듈


Client
관리자용 애플리케이션
회원 수정 요청
(매출액 필드 포함)
Controller
Service
save
Common Module
전달받은 매출액 필드를 바탕으로
회원등급 산정
MemberService
Member
Repository
DB
매출액 필드가 넘어오지 않아
NPE 발생
Controller
Service
save
유저 애플리케이션
회원 수정 요청
(매출액 필드 X)
Client

공통 모듈의 불편함

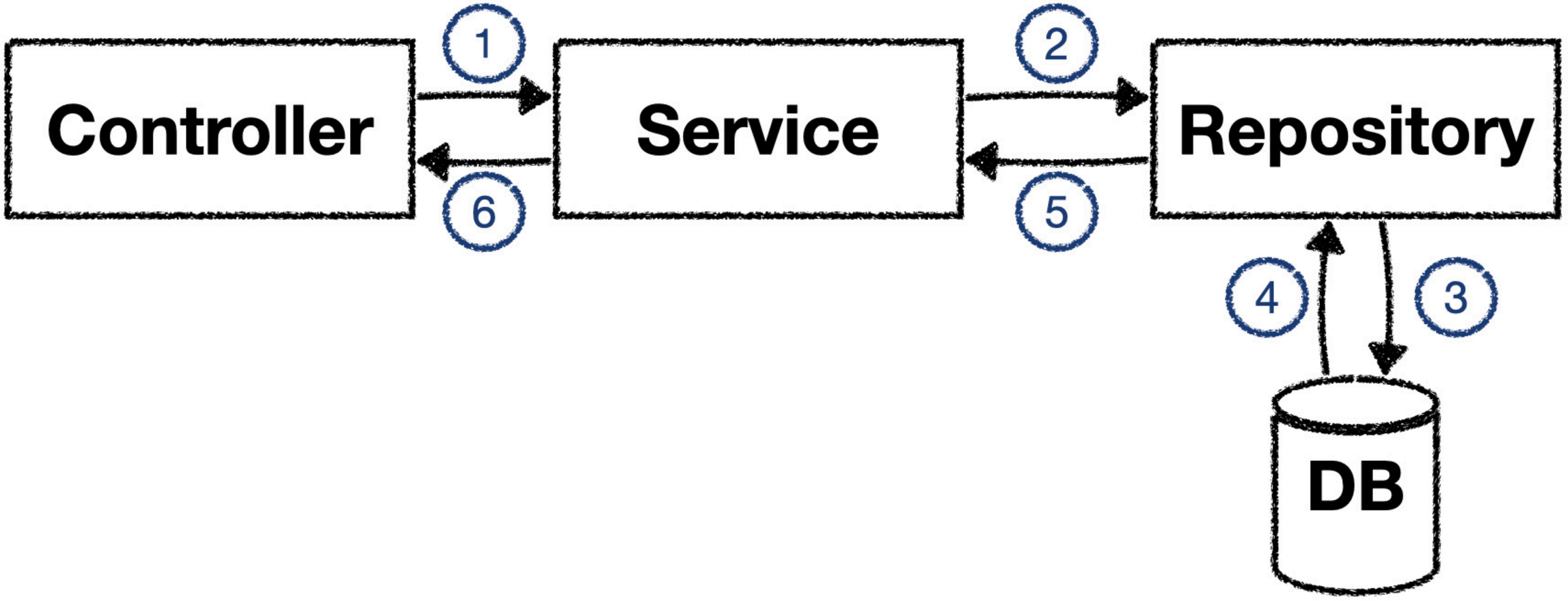
- 의존성 문제
  - 상호 참조
  - 가늠하기 어려운 수정 범위

모호한 역할

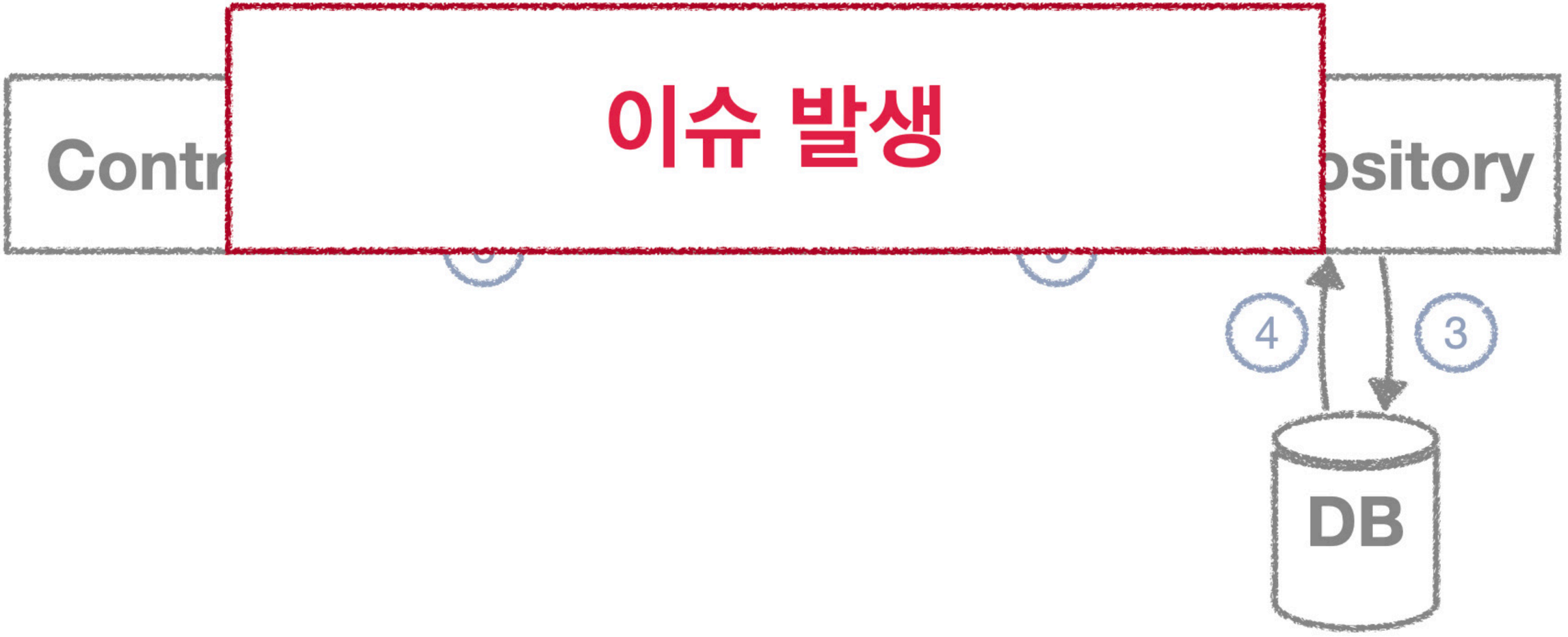
요구사항

- 회원의 수수료율 API 제공

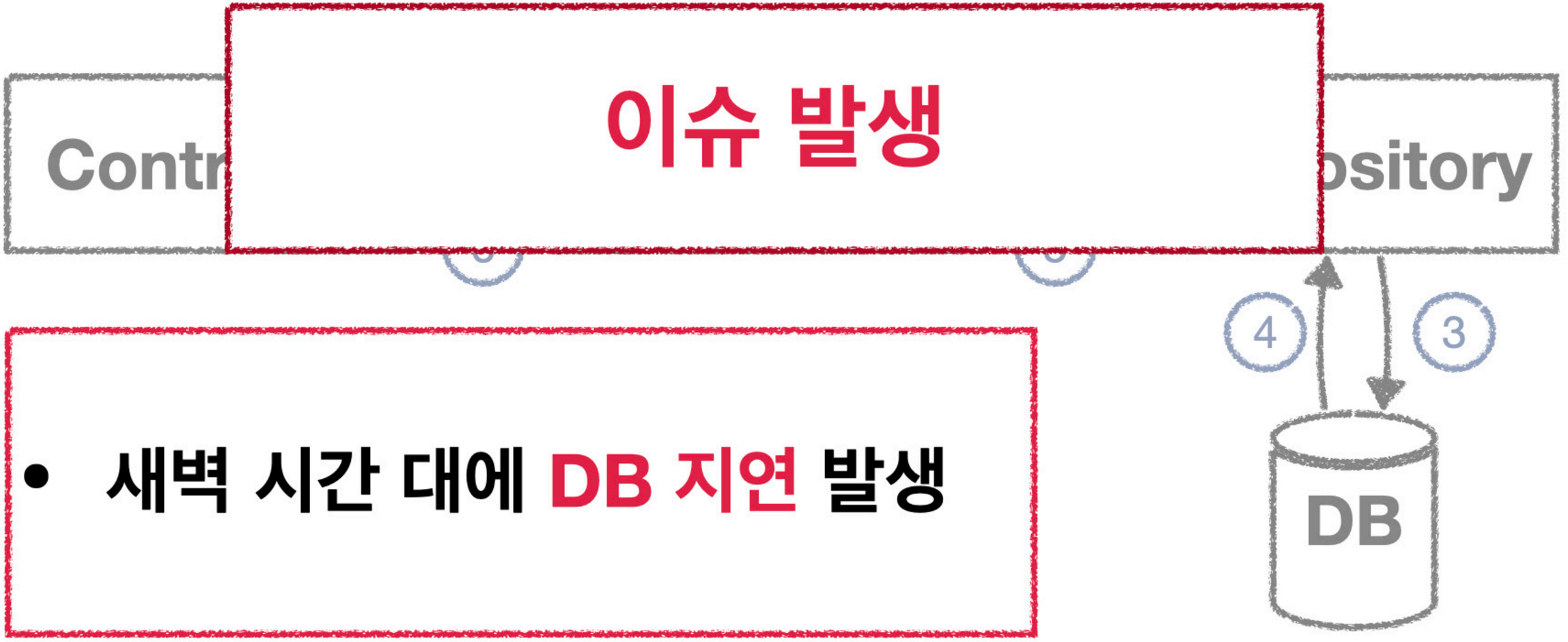
모호한 역할



모호한 역할

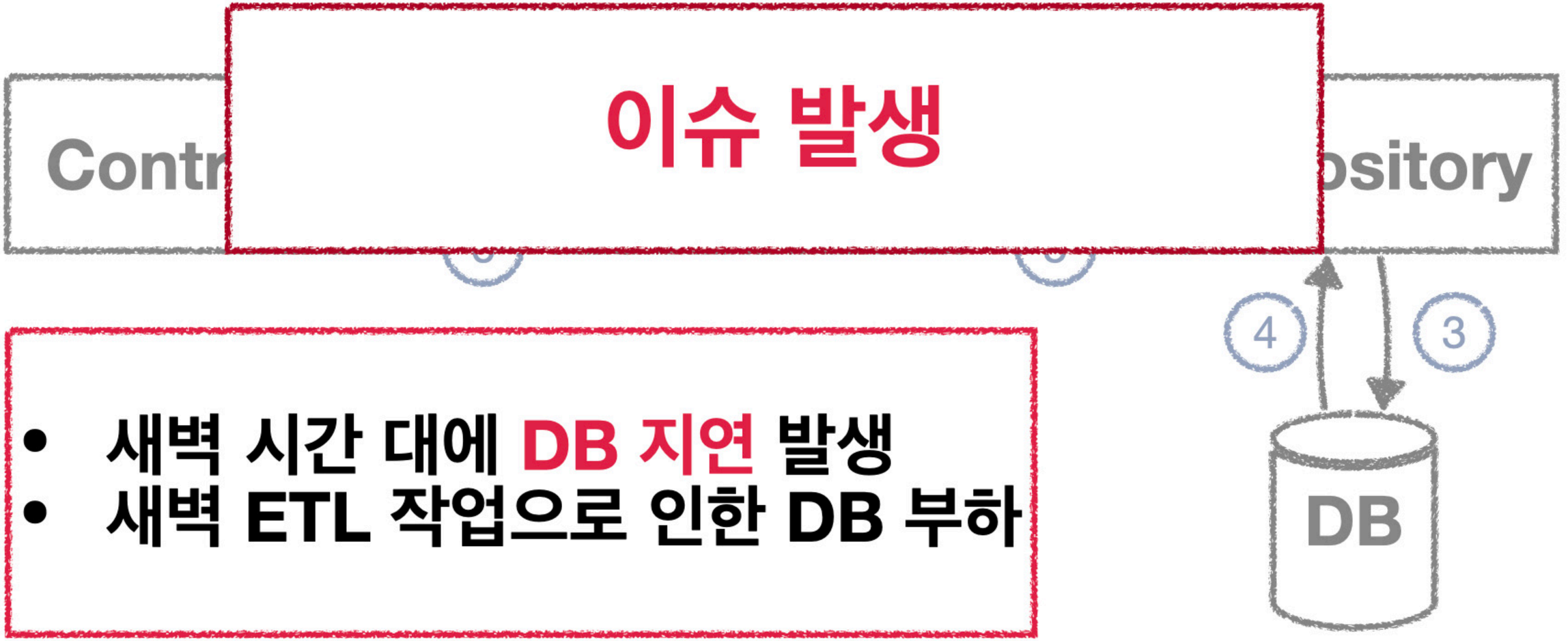


모호한 역할



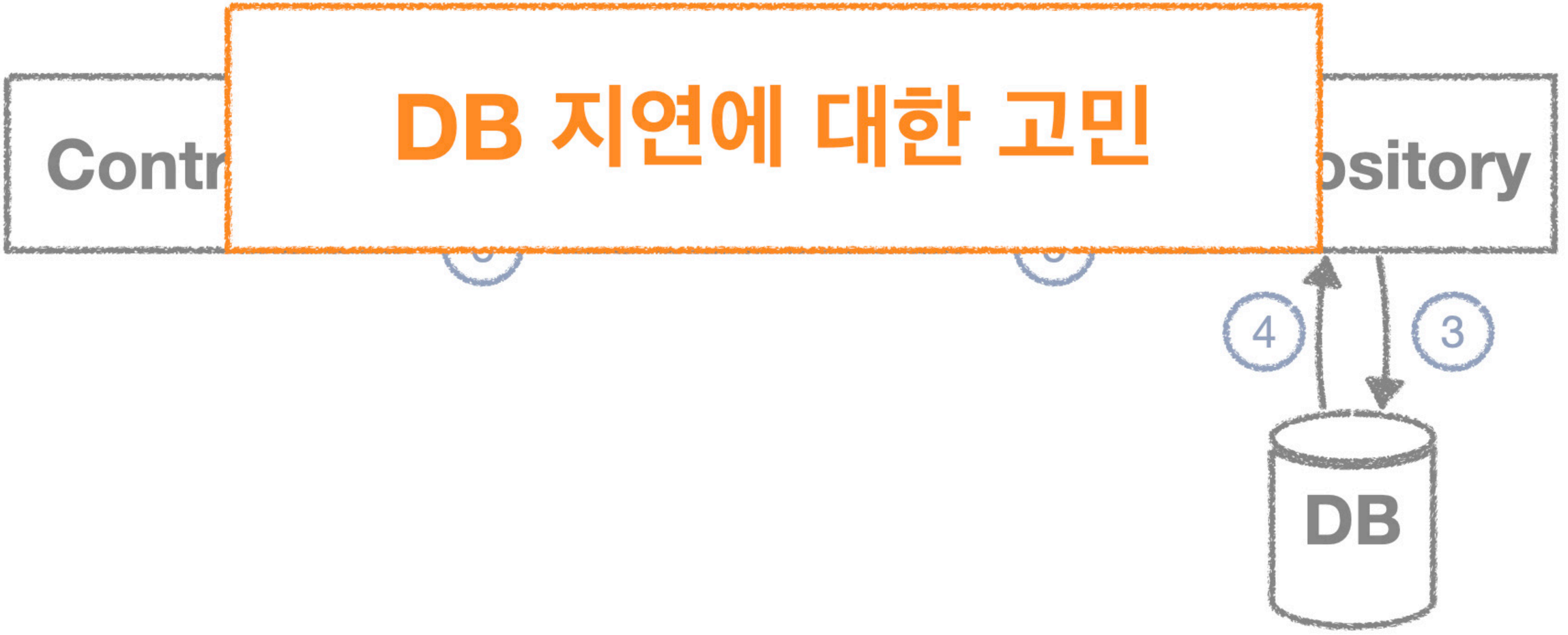
- 새벽 시간 대에 DB 지연 발생

모호한 역할

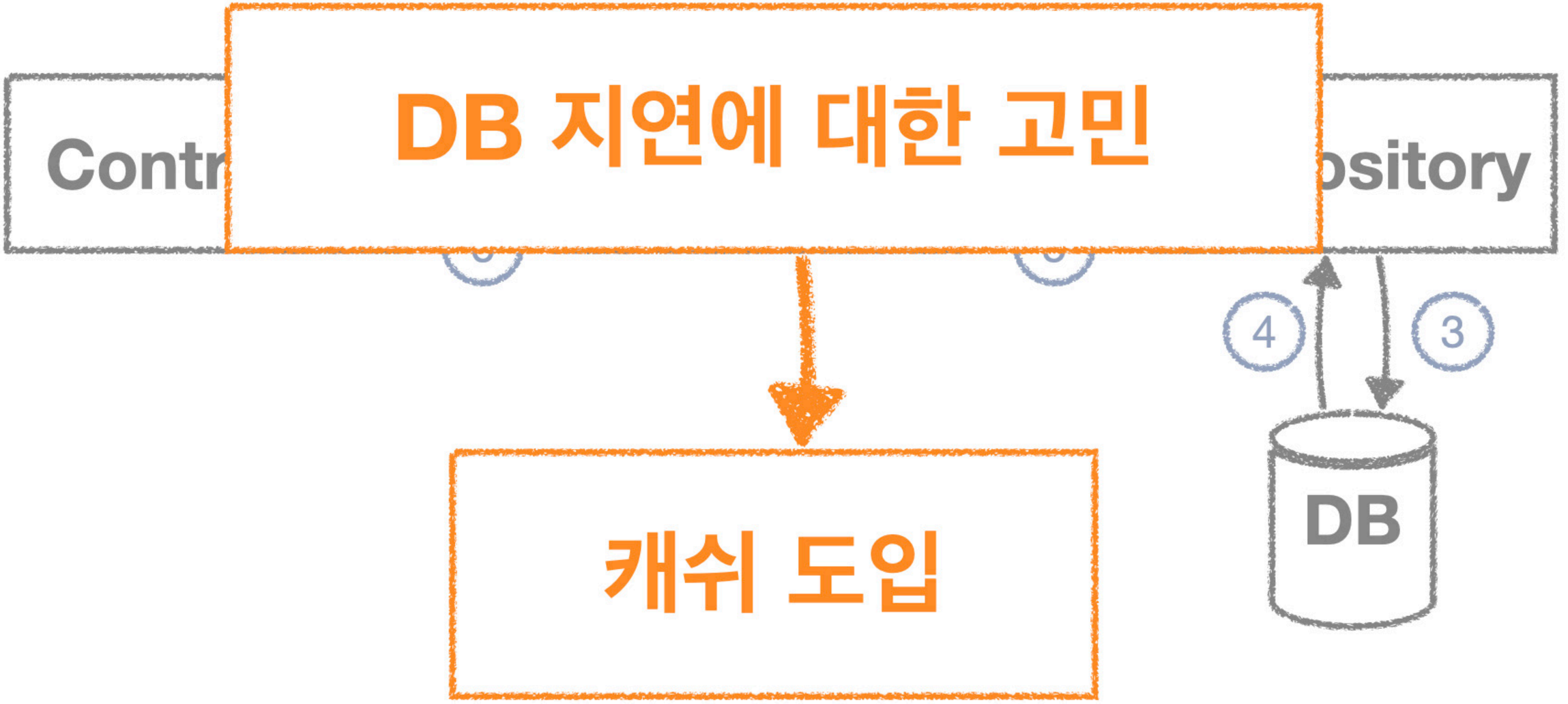


- 새벽 시간 대에 **DB 지연** 발생
- 새벽 ETL 작업으로 인한 DB 부하

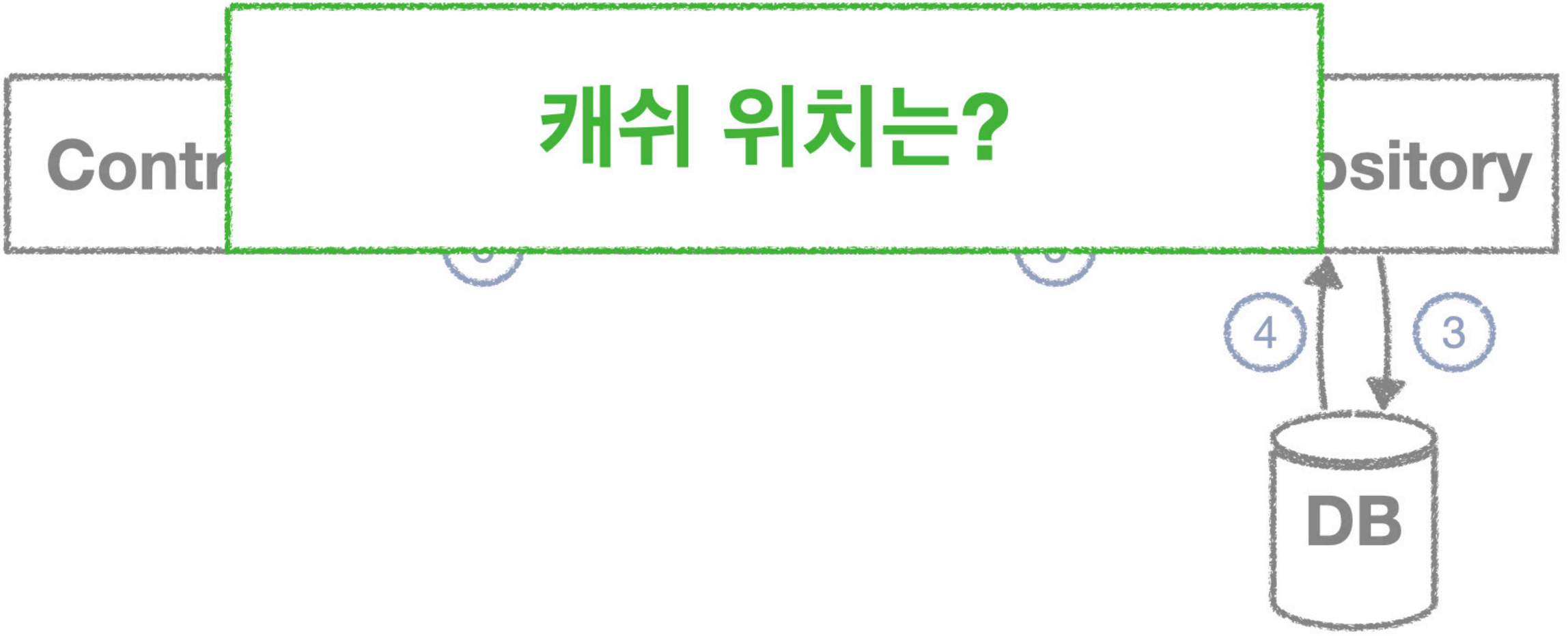
모호한 역할



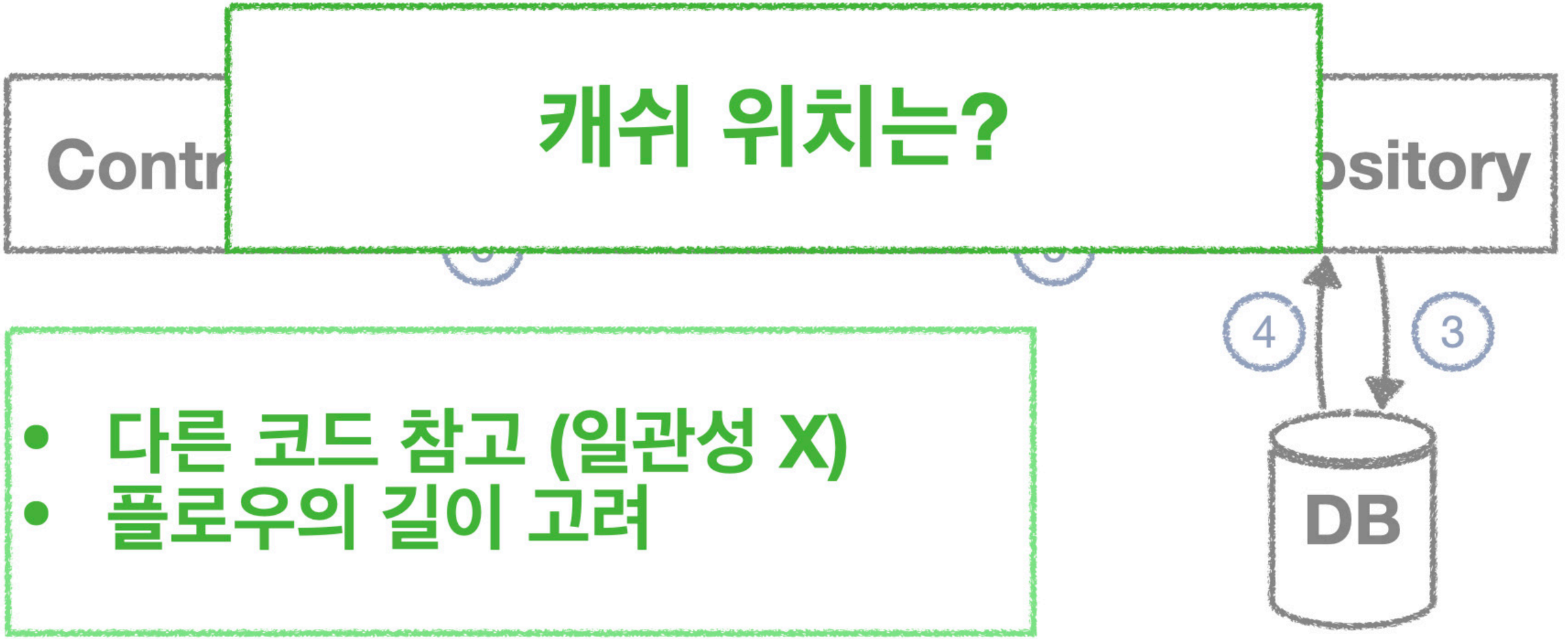
모호한 역할



모호한 역할

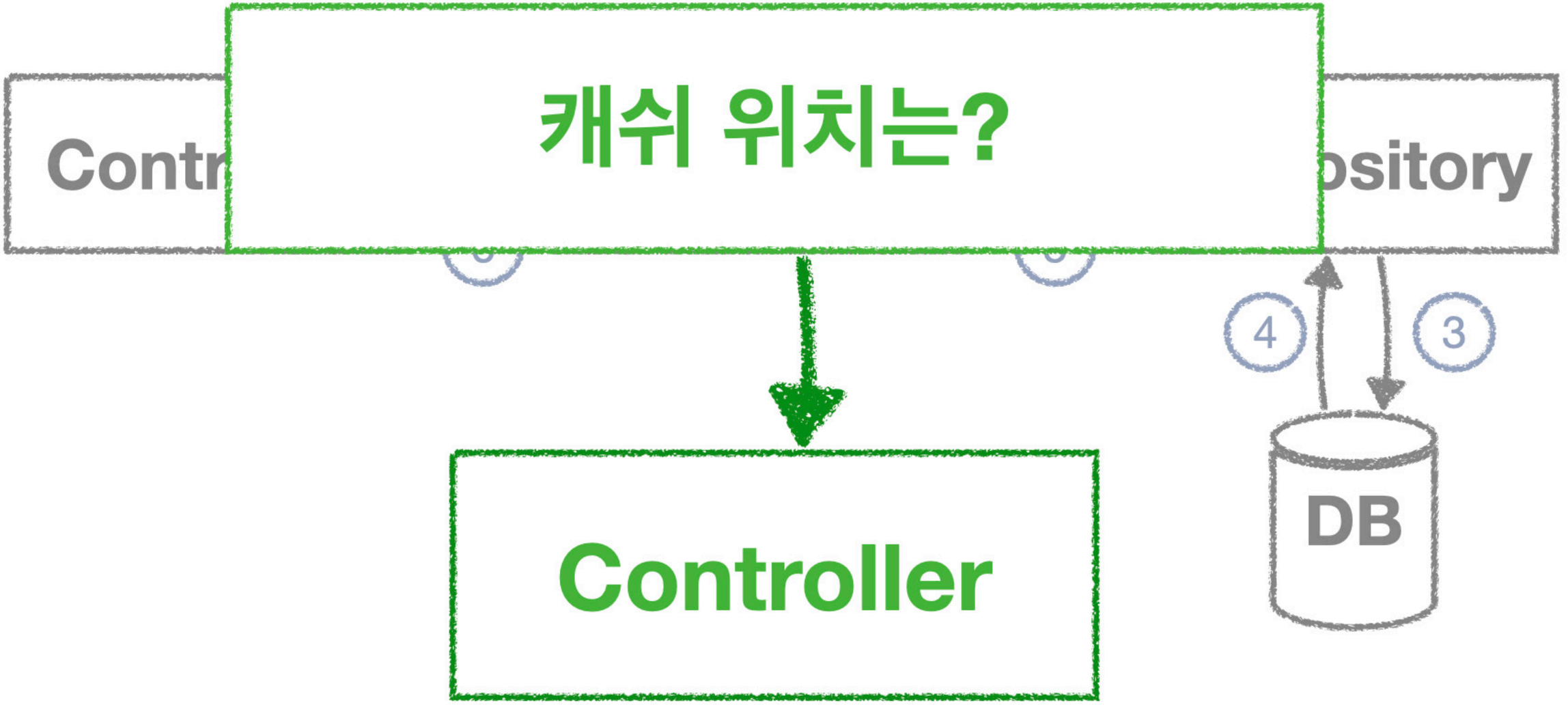


모호한 역할

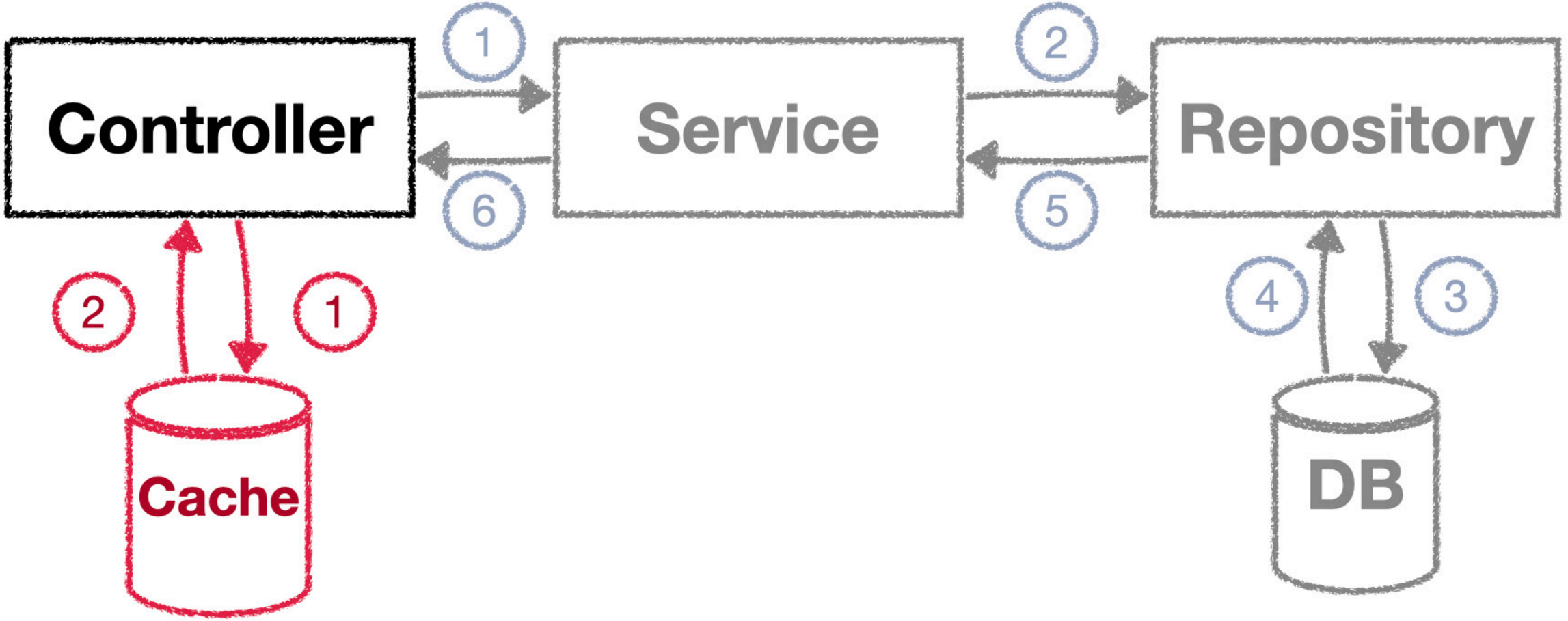


- 다른 코드 참고 (일관성 X)
- 플로우의 길이 고려

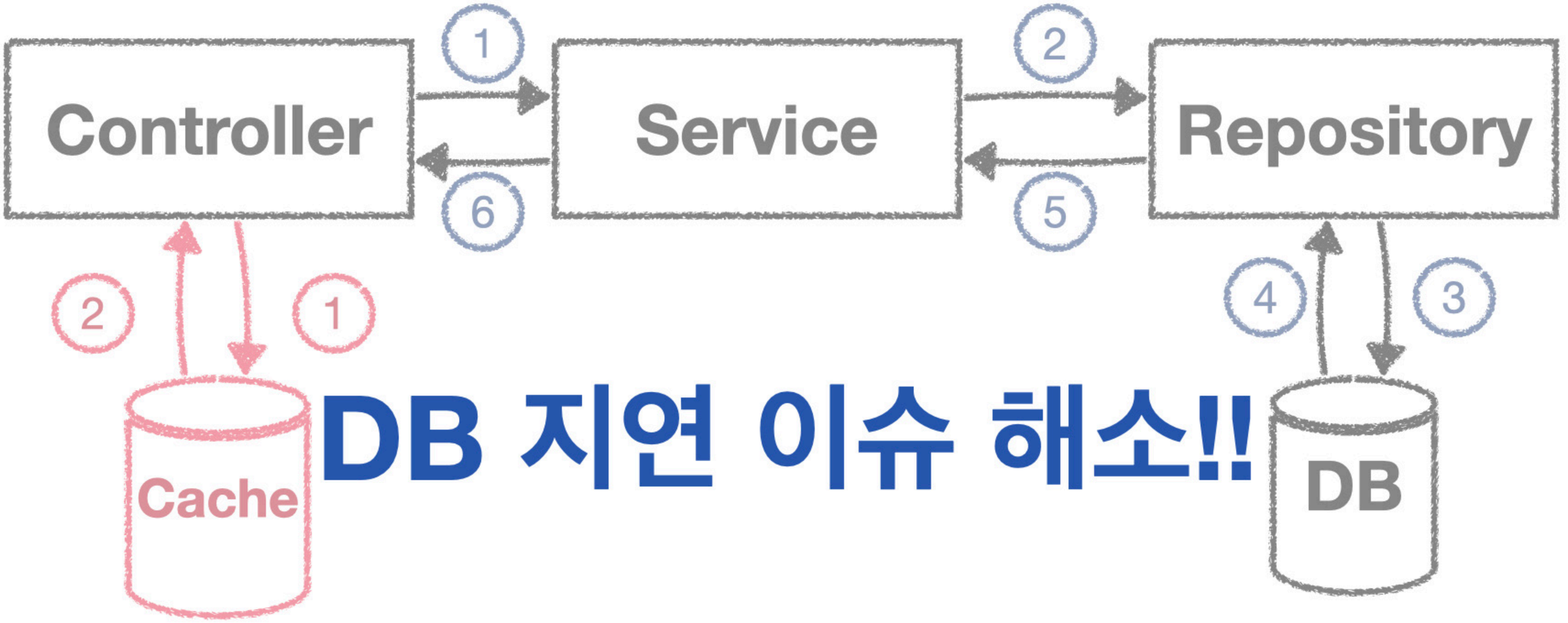
모호한 역할



모호한 역할



모호한 역할


Controller
Service
Repository
1
2
5
6
2
1
4
3
Cache
DB
DB 지연 이슈 해소!!

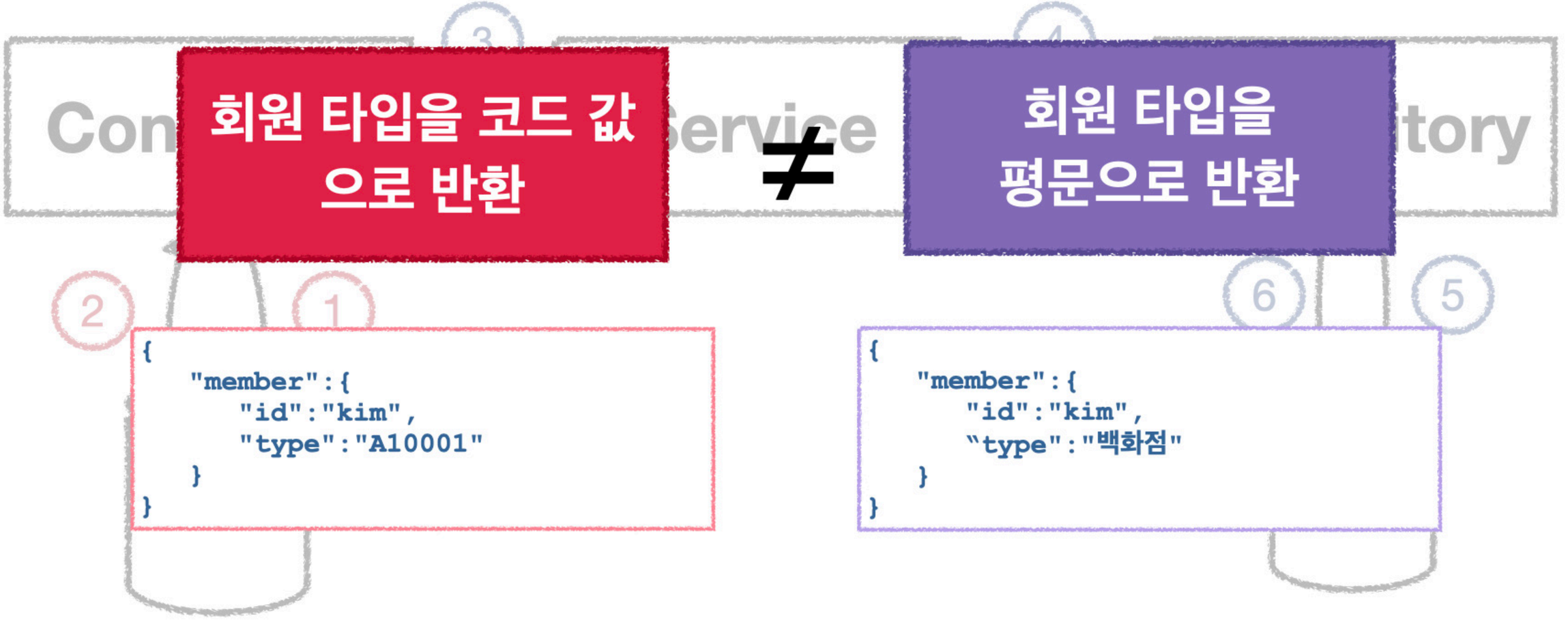
모호한 역할

DB 지연 이슈를 해결한 나는야 갓 주니어

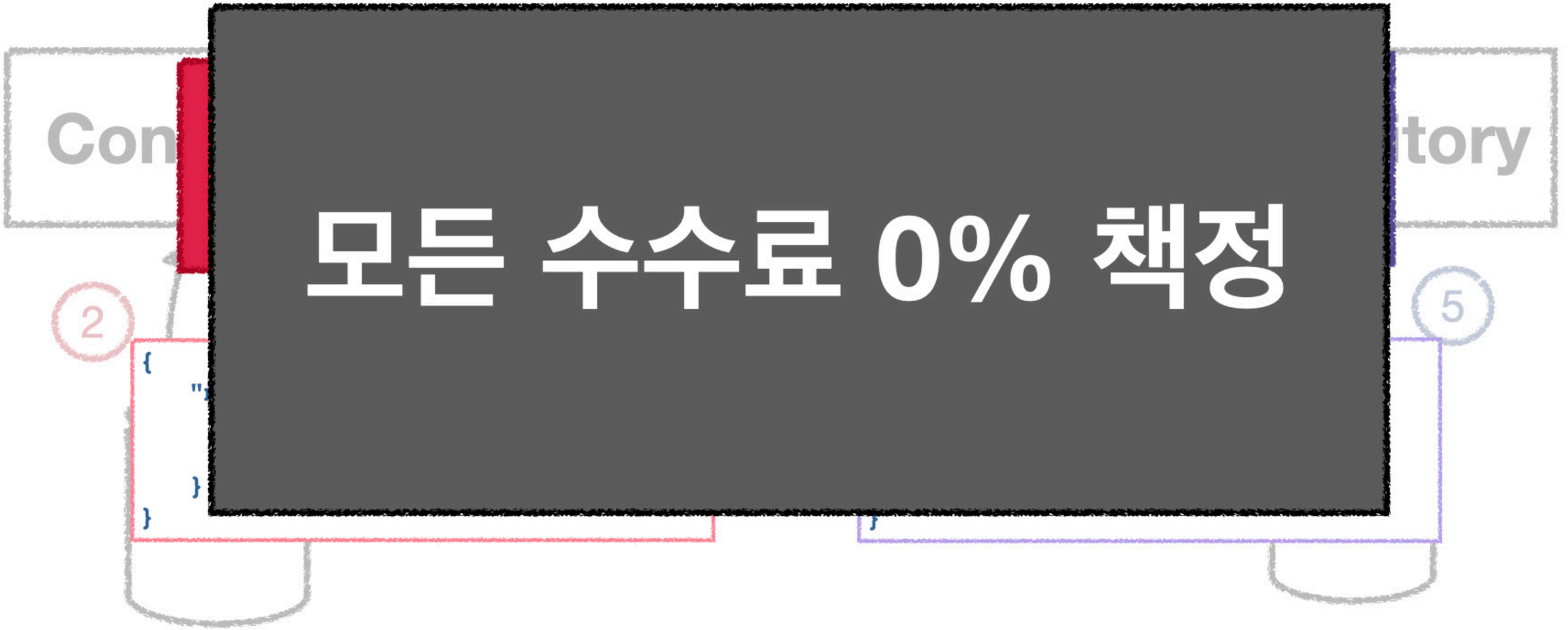
모호한 역할


Controller
Service
Repository
Cache
DB
1
2
3
4
5
6
DB 지연 이슈 해소!!
!장애 발생!

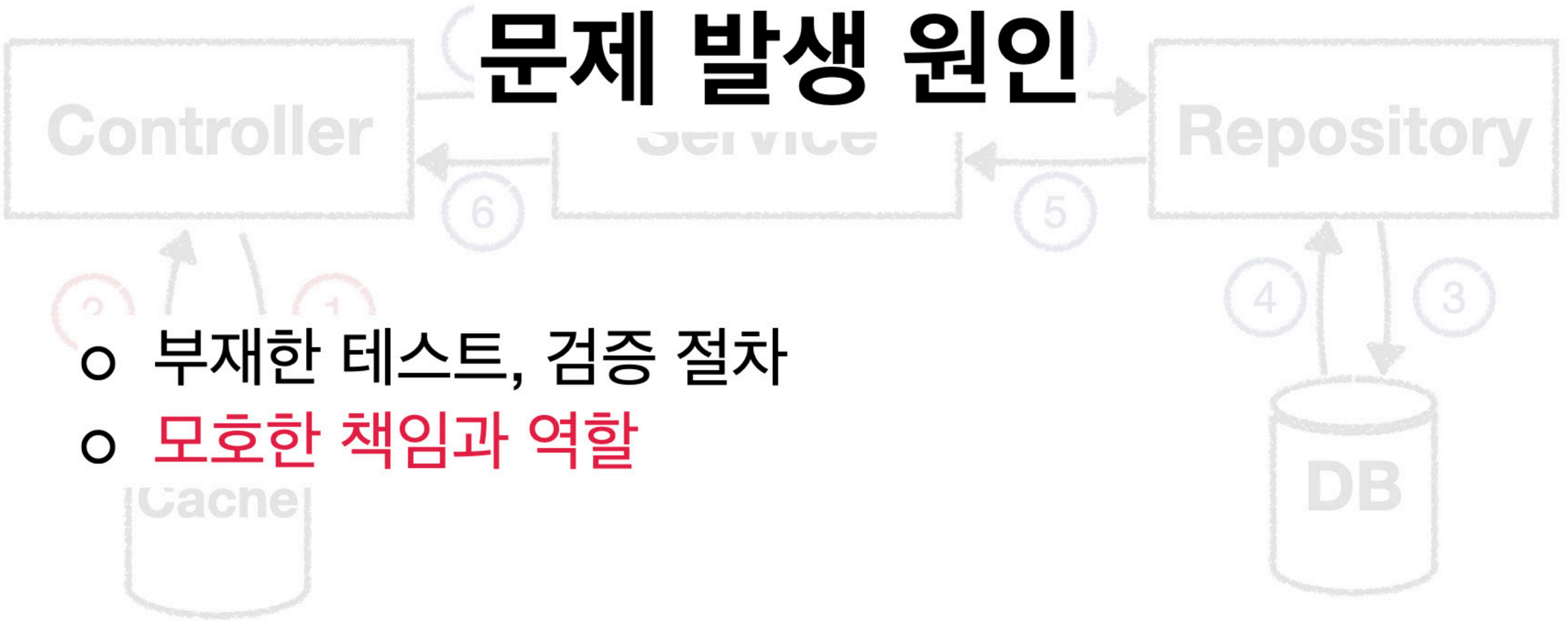
모호한 역할


회원 타입을 코드 값으로 반환
≠
회원 타입을 평문으로 반환
{
"member":{
"id":"kim",
"type":"A10001"
}
}
{
"member":{
"id":"kim",
"type":"백화점"
}
}

모호한 역할

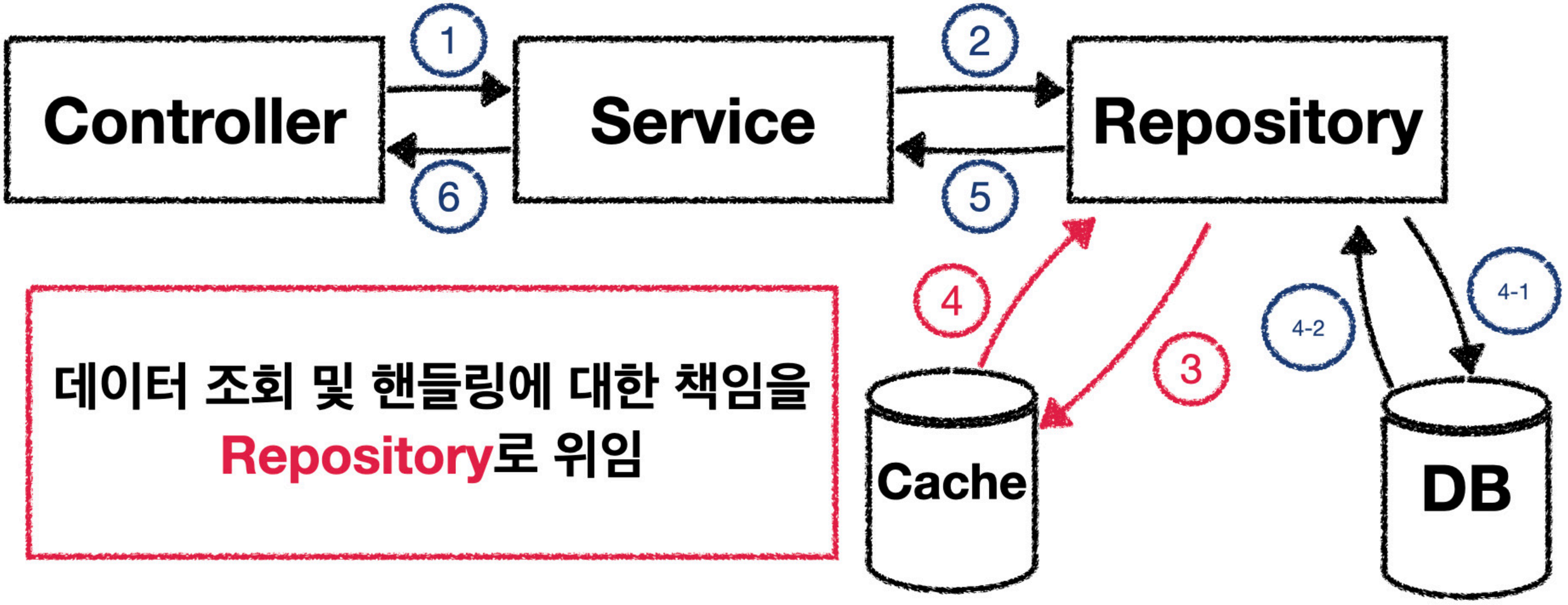


문제 발생 원인



- 부재한 테스트, 검증 절차
- 모호한 책임과 역할

모호한 역할



데이터 조회 및 핸들링에 대한 책임을
Repository로 위임

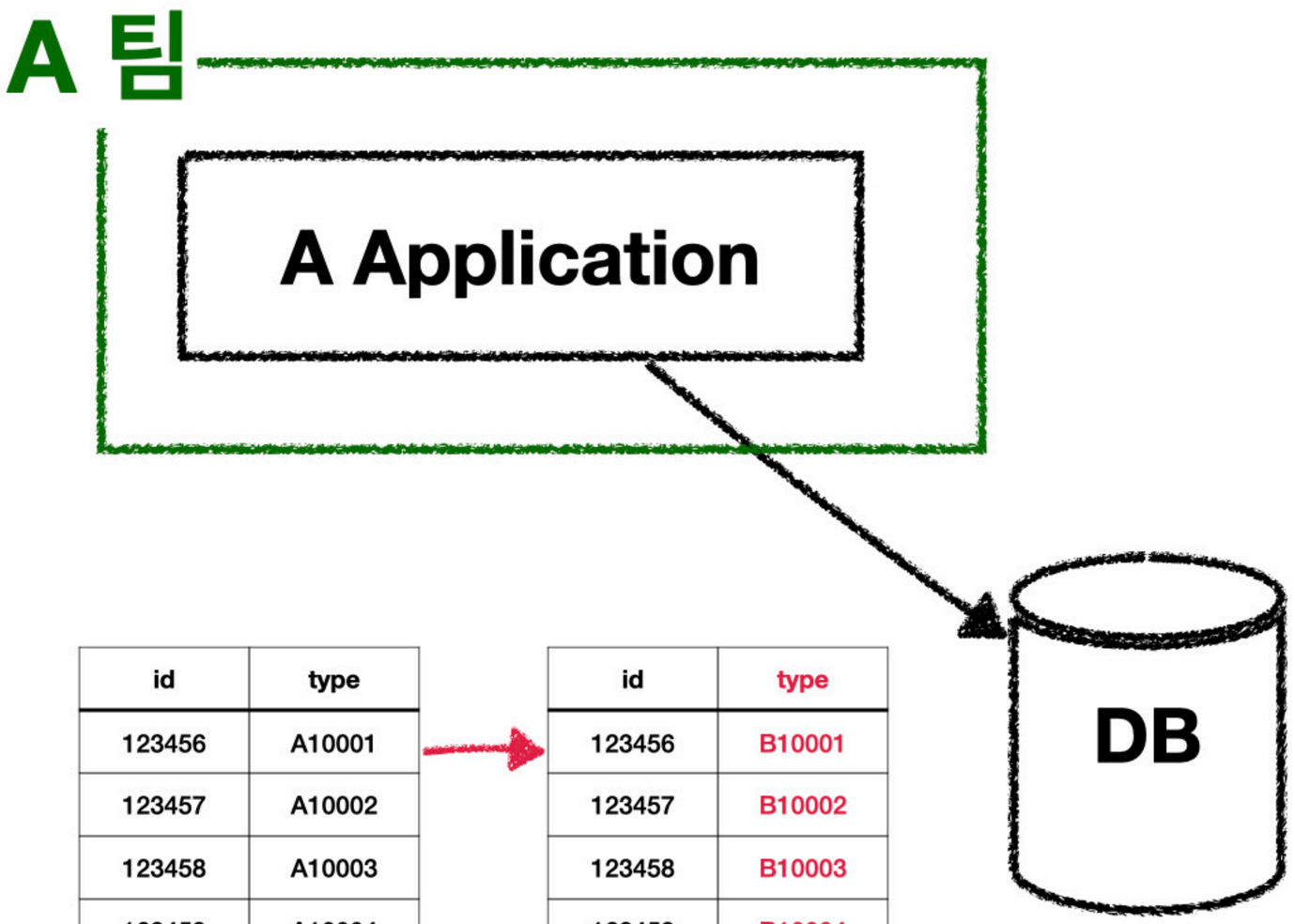

인간의 욕심은 끝이 없고
같은 실수를 반복한다.

공용 데이터베이스

요구사항

- A 팀 내에서 A10 코드값 사용이 중복
- 공용 데이터베이스 내 테이블 필드의 타입 값
  A10에서 B10으로 마이그레이션 진행

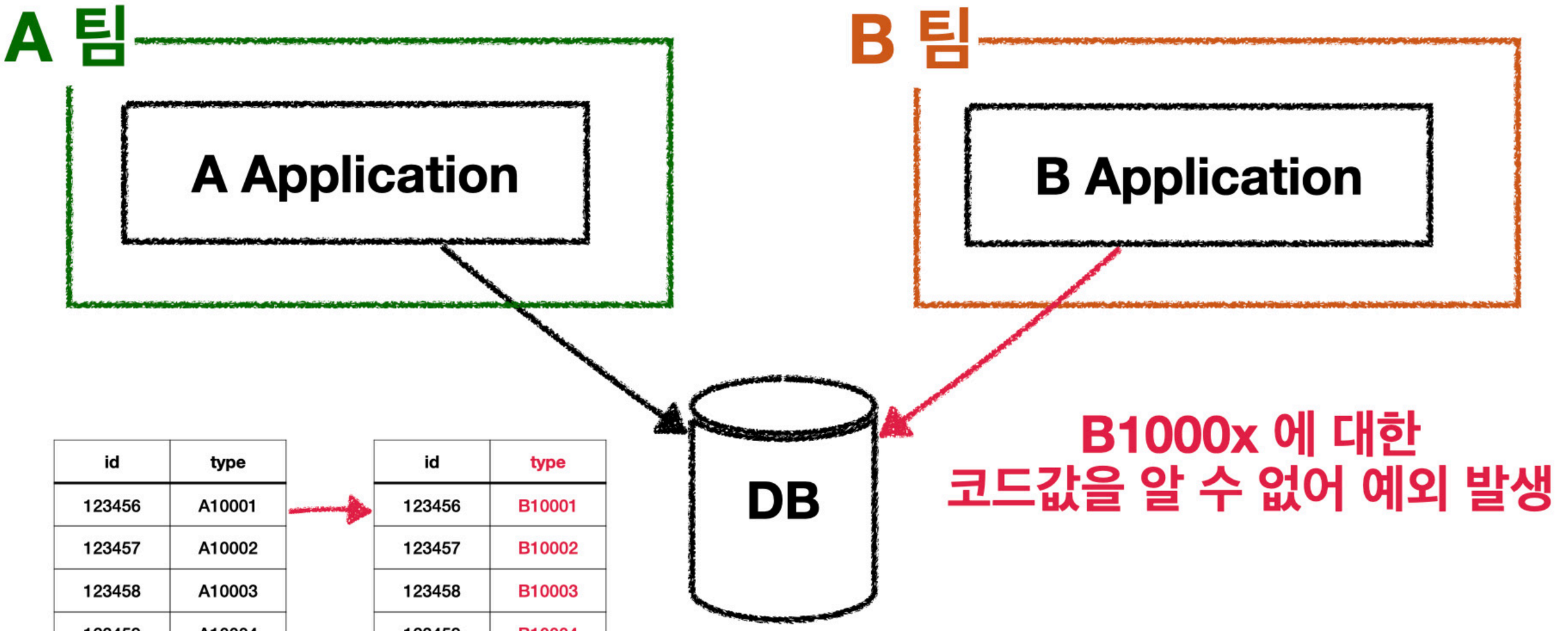
공용 데이터베이스



id	type
123456	A10001
123457	A10002
123458	A10003
123459	A10004

id	type
123456	B10001
123457	B10002
123458	B10003
123459	B10004

공용 데이터베이스



id	type
123456	A10001
123457	A10002
123458	A10003
123459	A10004

id	type
123456	B10001
123457	B10002
123458	B10003
123459	B10004

개발과 배포가 두려워요...

익숙함에 속아 당연하다고 생각


동료가 돼라!

개념을 하사 받은 주니어들

Port & Adapter Architecture

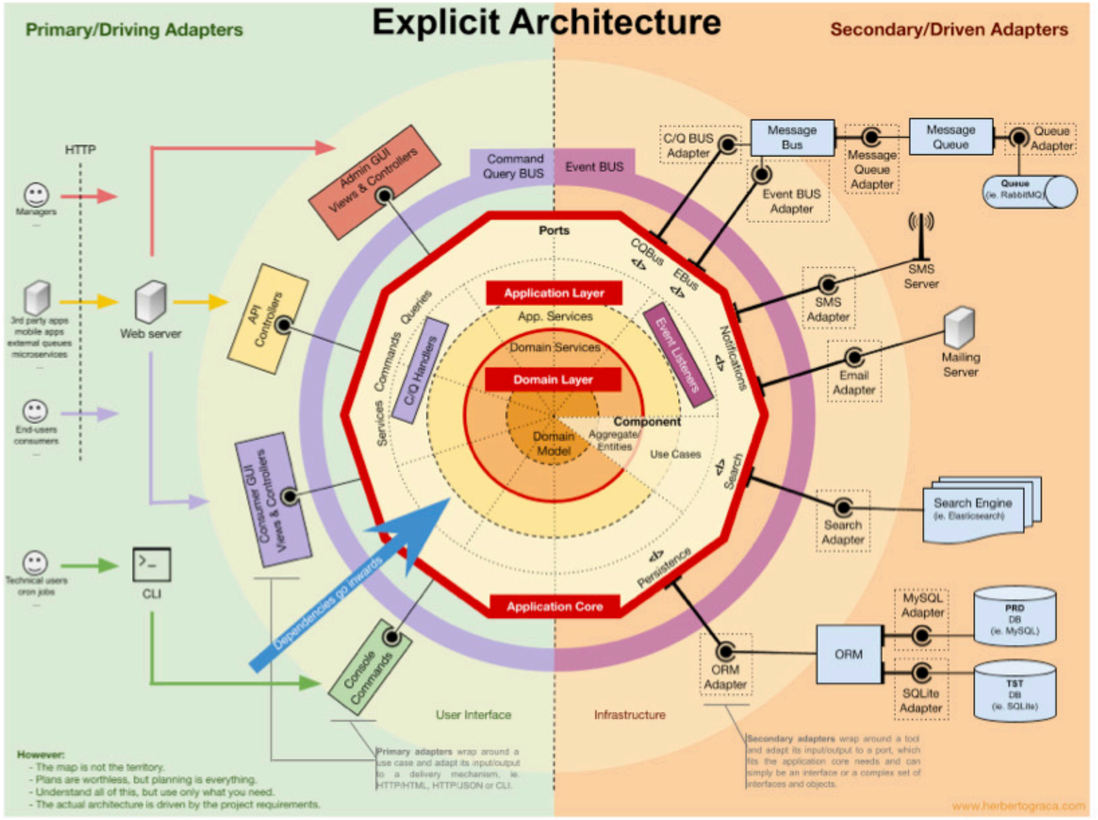
Event Driven Architecture

Domain Driven Architecture

Port & Adapter Architecture

Port & Adapter Architecture

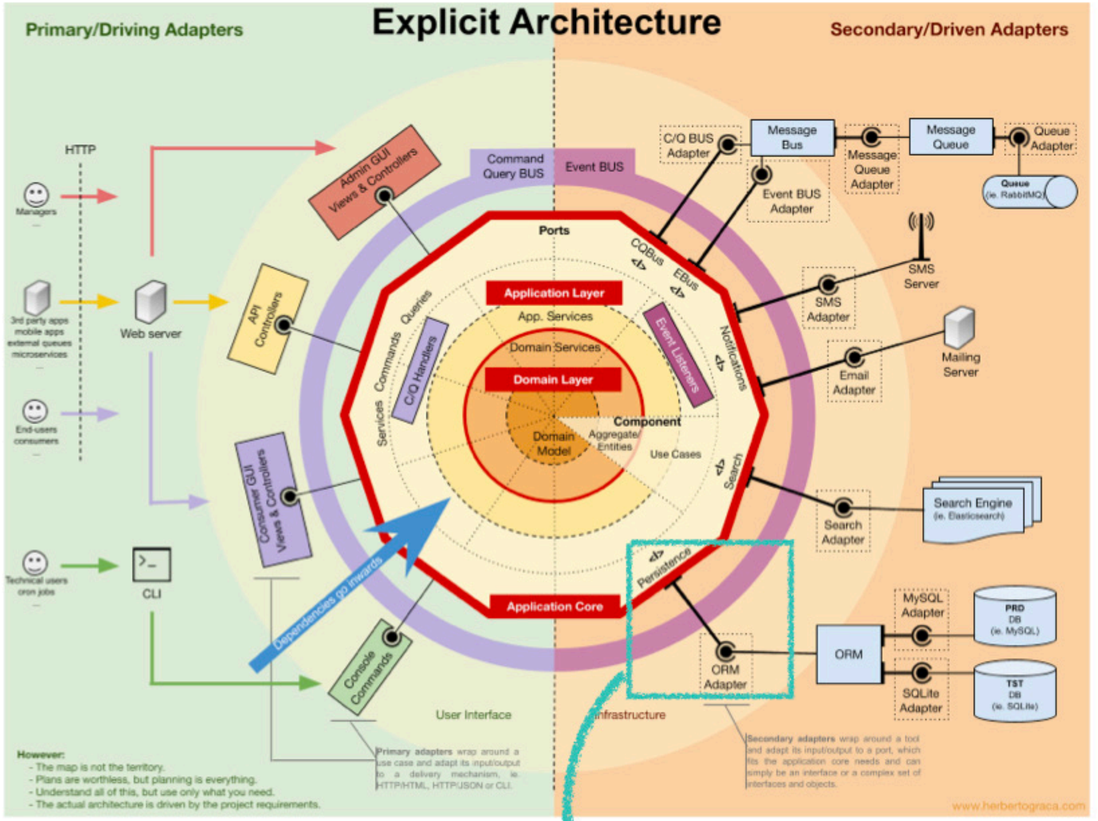
- 역할 별로 레이어 구성
  - Domain
  - Application
  - User Interface, Infrastructure



Port & Adapter Architecture

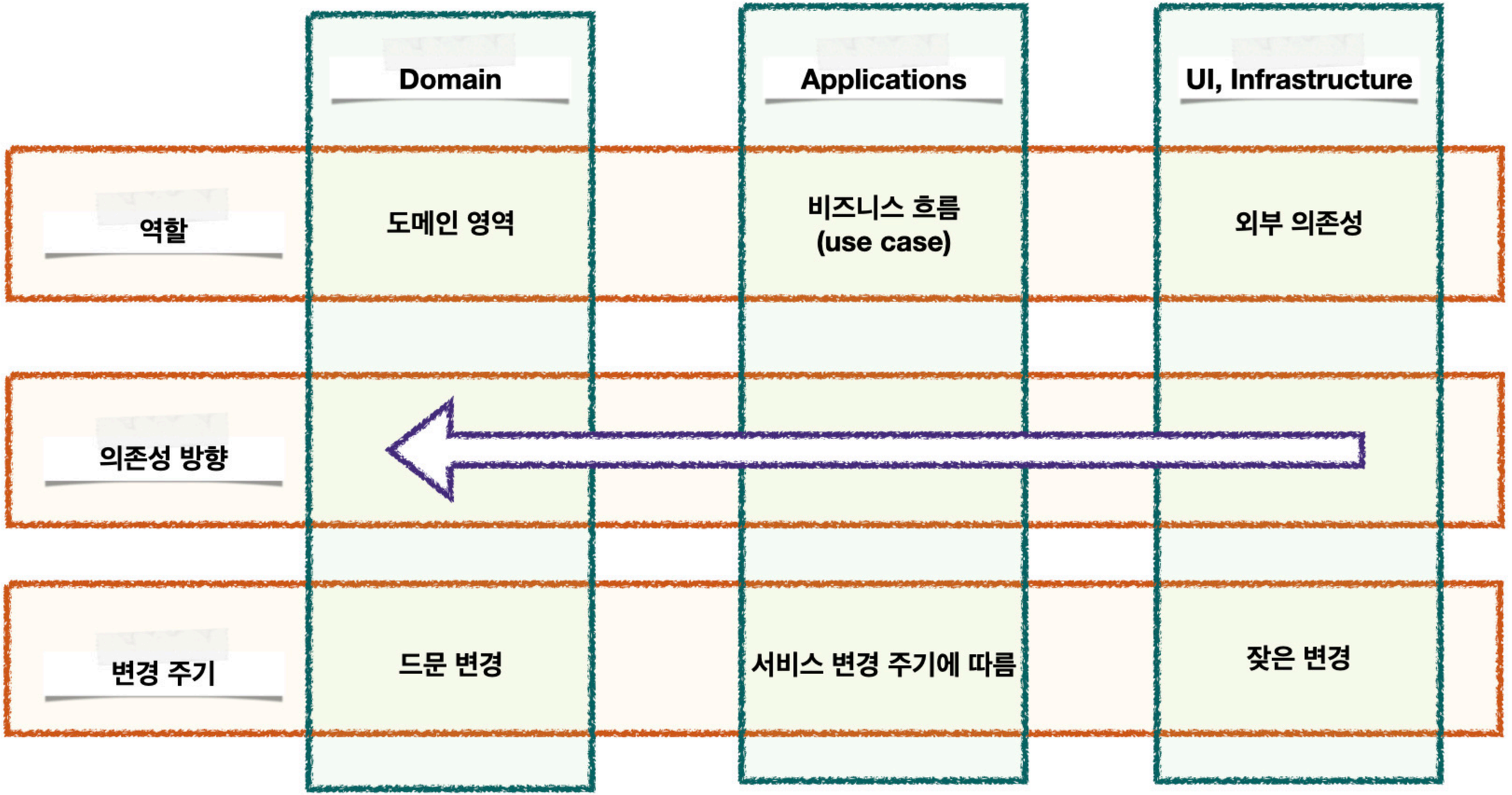
- 역할 별로 레이어 구성
  - Domain
  - Application
  - User Interface, Infrastructure

- 어댑터 패턴
  - 비즈니스 로직과 외부 의존성은 포트에 어댑터를 연결하는 방식으로 통신



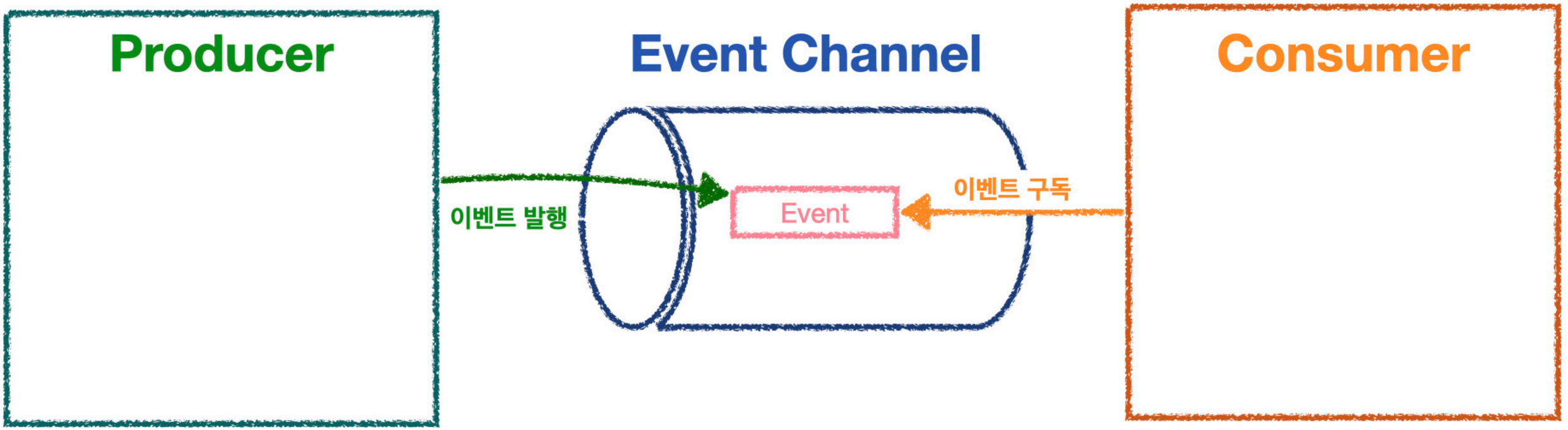
포트와 어댑터

Port & Adapter Architecture

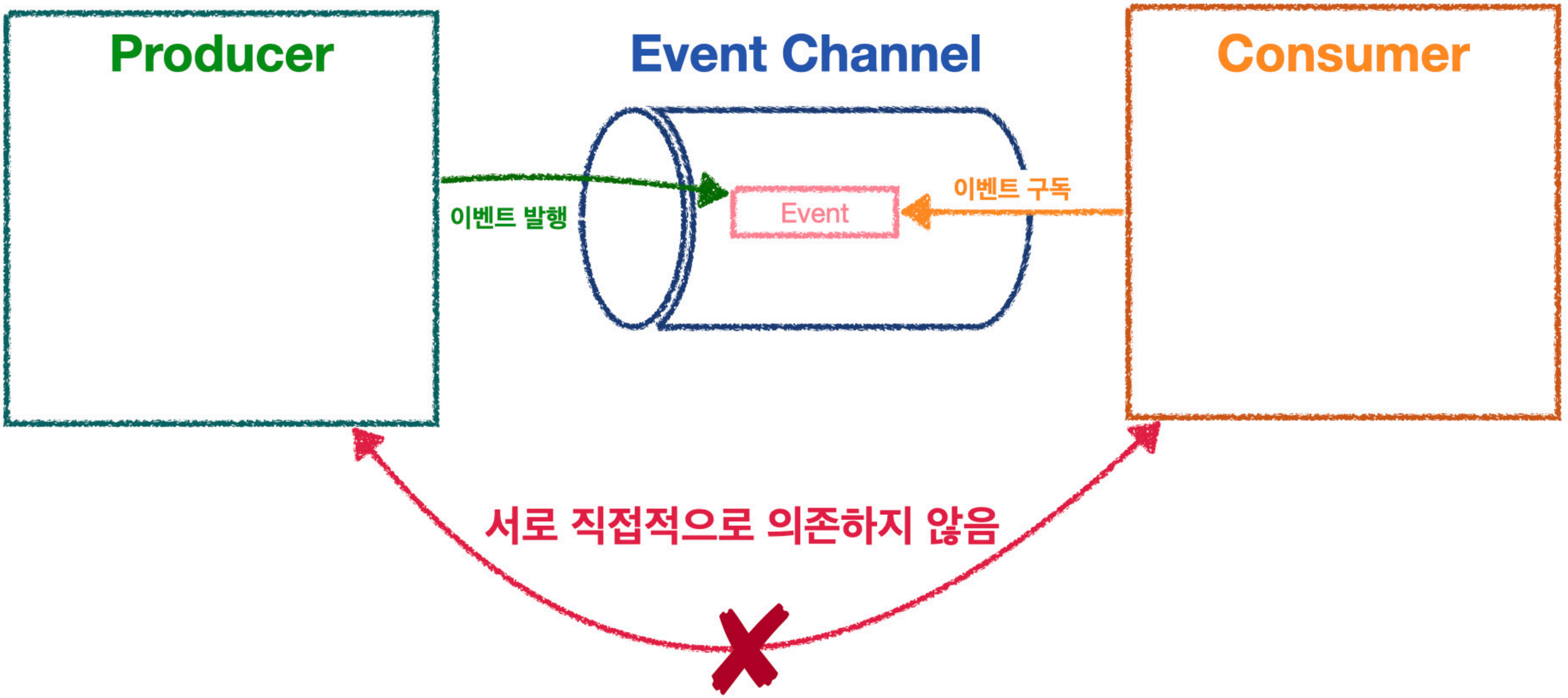


Event Driven Architecture

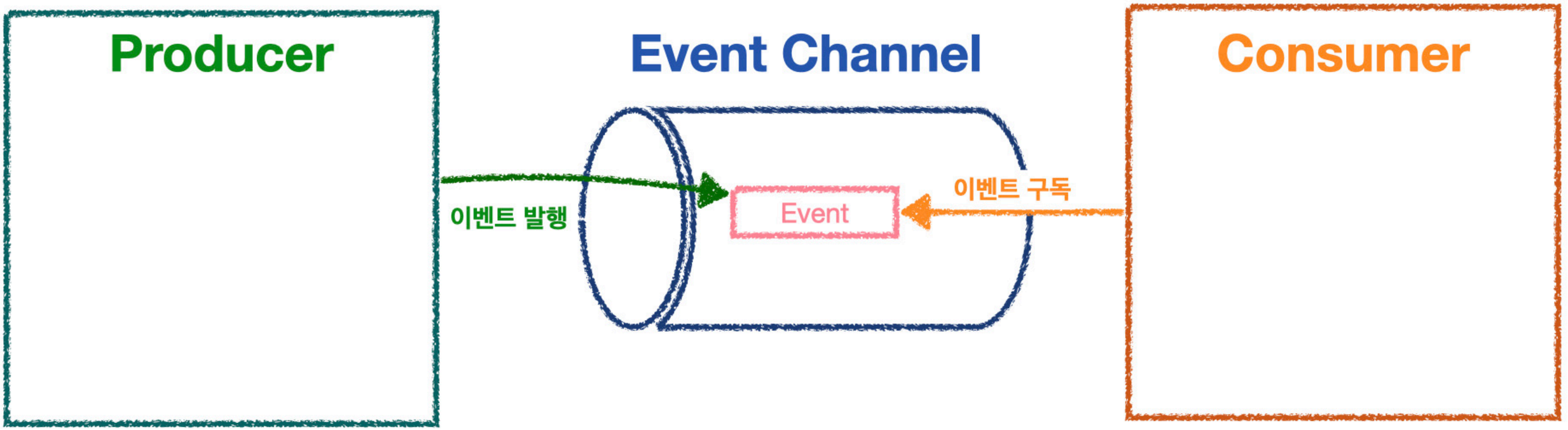
Event Driven Architecture



Event Driven Architecture



Event Driven Architecture



깨진 무결성은 이벤트 재발행으로 복구 가능

Eventual Consistency

Domain Driven Architecture

Domain Driven Architecture

빈약한(Anemic) 도메인

풍부한(Rich) 도메인

참고: http://martinfowler.com/bliki/AnemicDomainModel.html

Domain Driven Architecture

빈약한(Anemic) 도메인

행위는 거의 없고 getter와 setter만 존재

풍부한(Rich) 도메인

참고: http://martinfowler.com/bliki/AnemicDomainModel.html

Domain Driven Architecture

빈약한(Anemic) 도메인

행위는 거의 없고 getter와 setter만 존재

풍부한(Rich) 도메인

참고: http://martinfowler.com/bliki/AnemicDomainModel.html

Domain Driven Architecture

빈약한(Anemic) 도메인

행위는 거의 없고 getter와 setter만 존재

참고: http://martinfowler.com/bliki/AnemicDomainModel.html

풍부한(Rich) 도메인

도메인 객체에 행위를 위임


Port & Adapter
domain driven
event driven
총 60kg
군장 세 개 둘러메고 행군 다시 시작

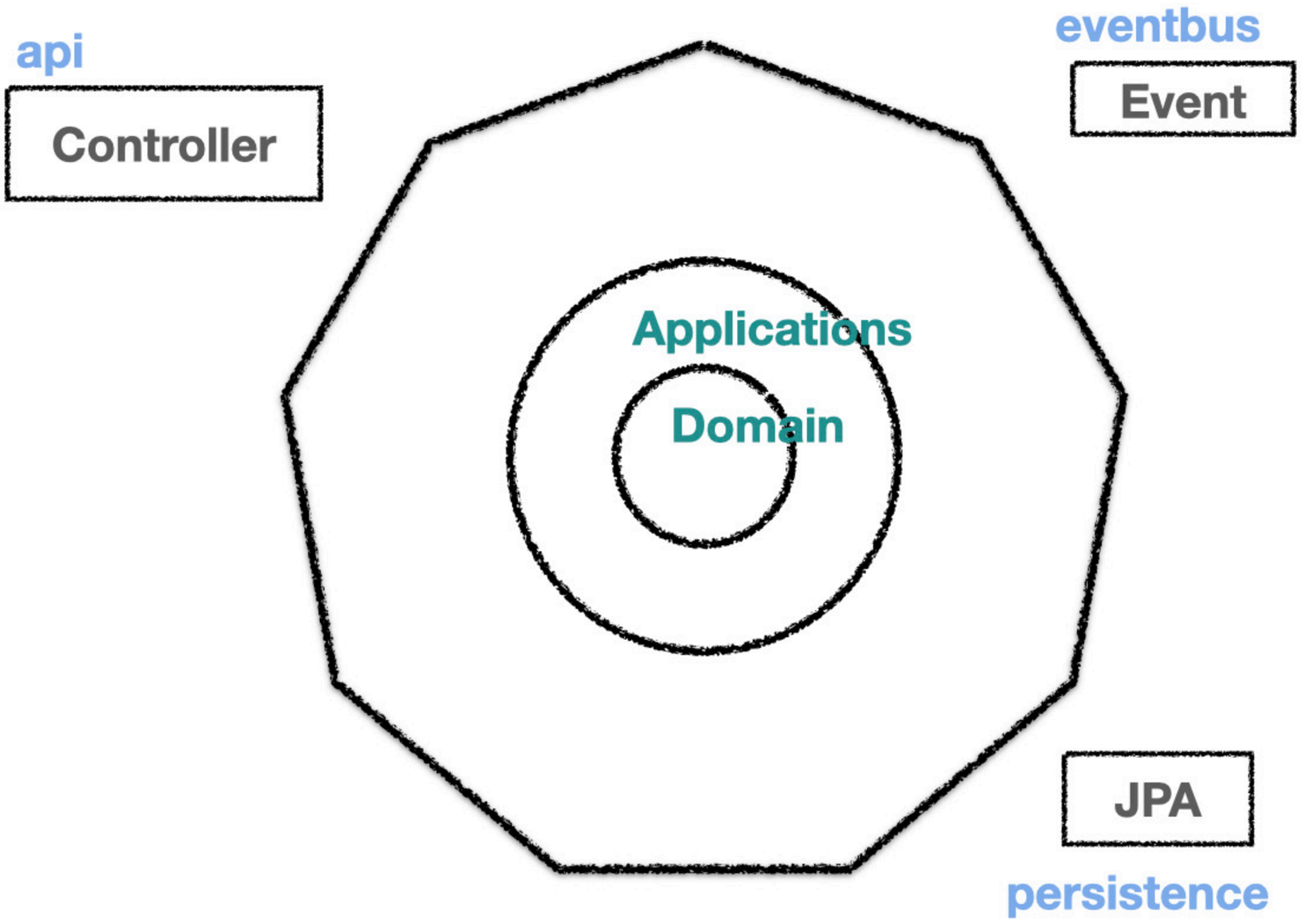
새로운 시스템 구성

- 패키지 구성
- 이벤트 구성
- 도메인 구성

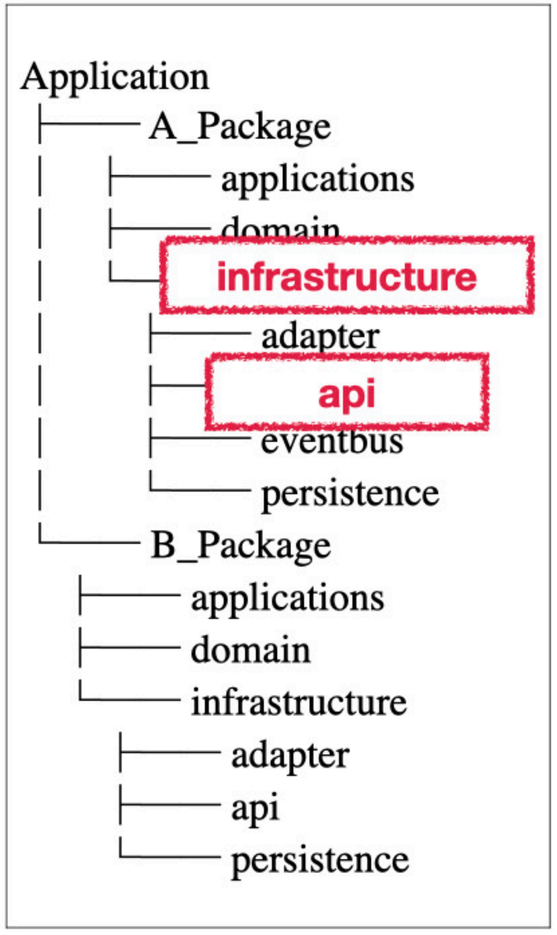
패키지 구성

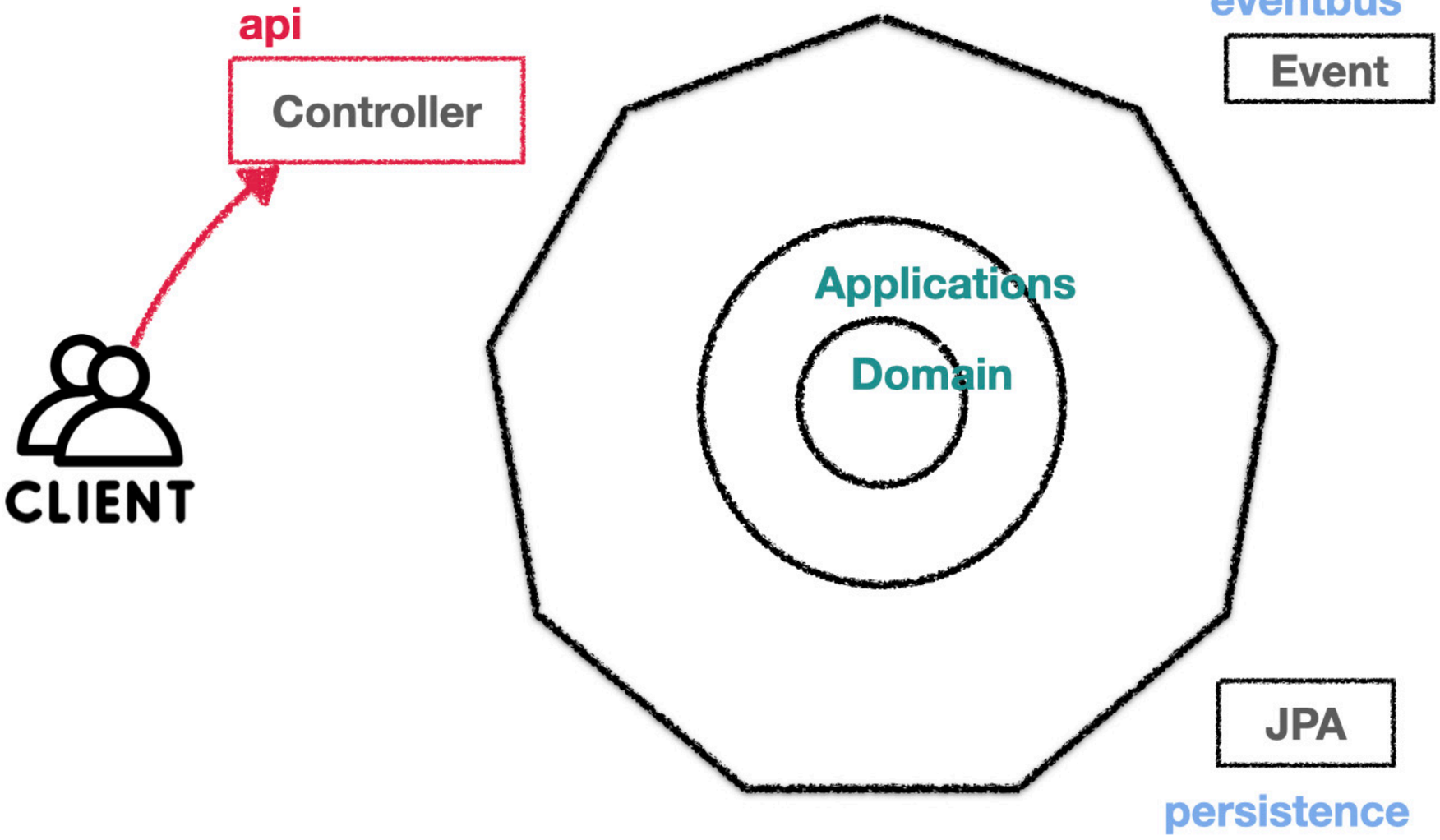
패키지 구성

```
Application
├── A_Package
│   ├── applications
│   ├── domain
│   └── infrastructure
│       ├── adapter
│       ├── api
│       ├── eventbus
│       └── persistence
└── B_Package
    ├── applications
    ├── domain
    └── infrastructure
        ├── adapter
        ├── api
        └── persistence
```

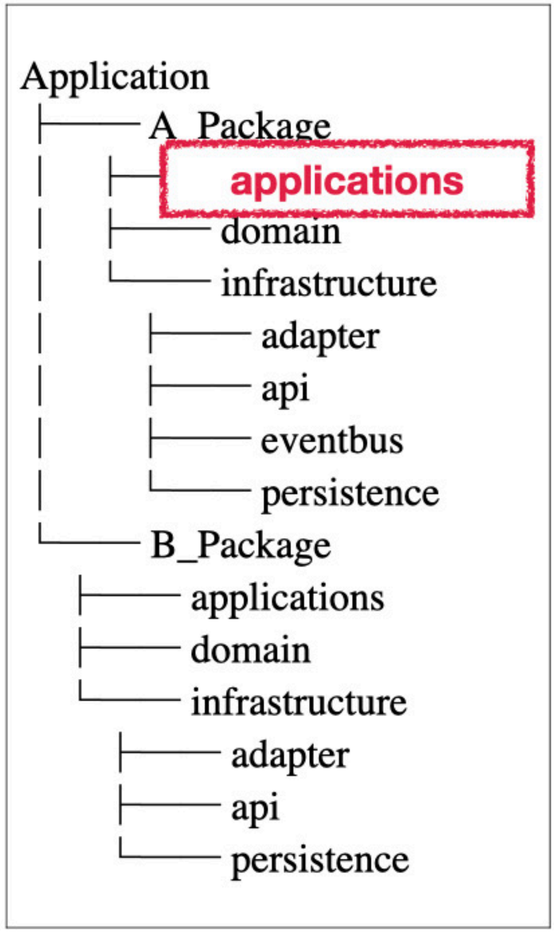


패키지 구성

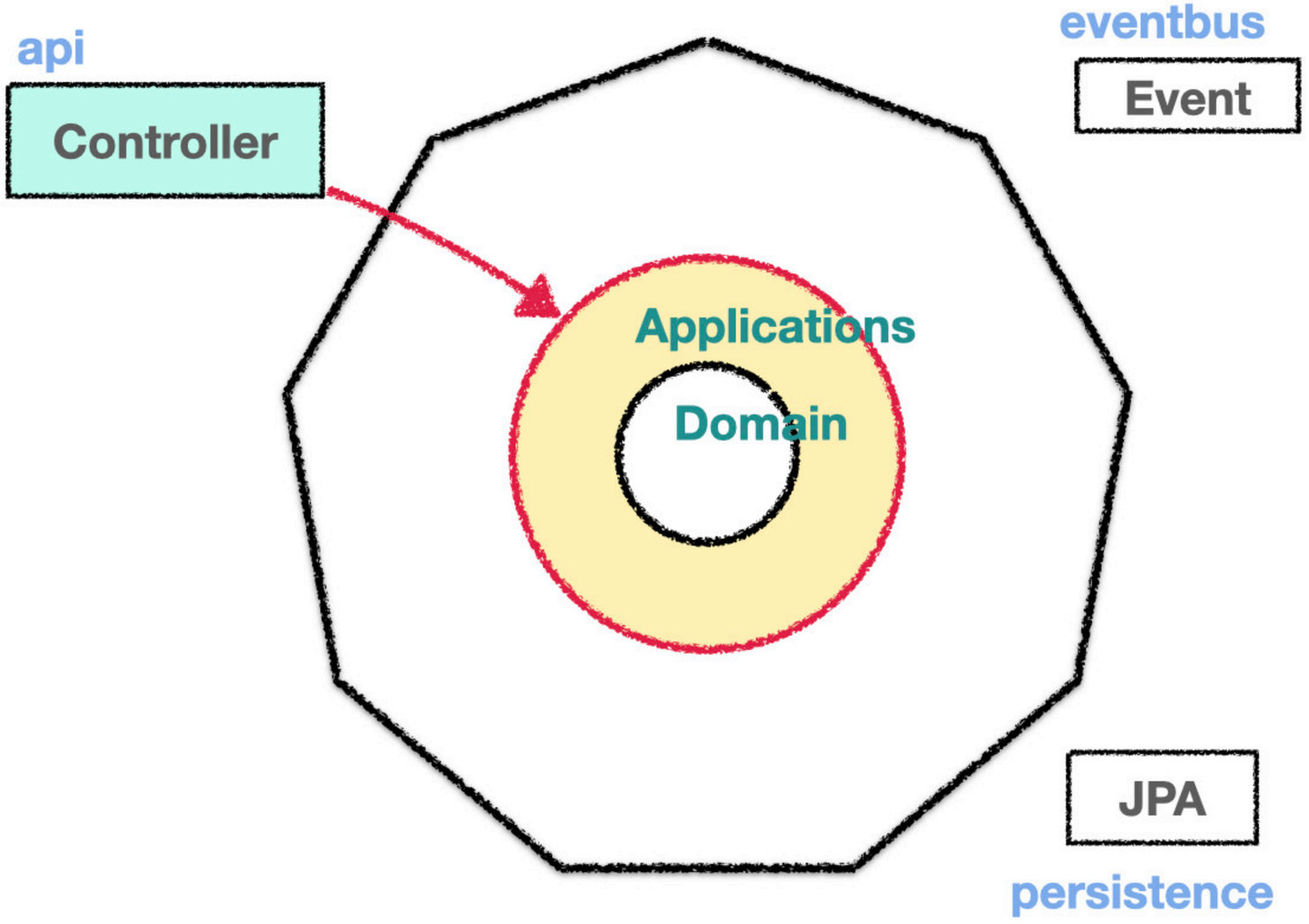

Application
A_Package
applications
domain
infrastructure
adapter
api
eventbus
persistence
B_Package
applications
domain
infrastructure
adapter
api
persistence


api
Controller
CLIENT
eventbus
Event
Applications
Domain
JPA
persistence

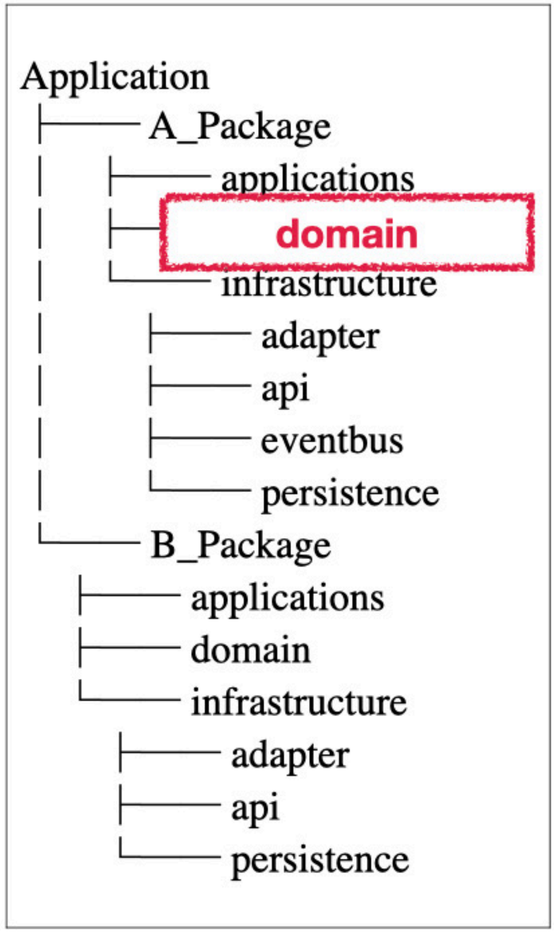
패키지 구성


Application
A_Package
applications
domain
infrastructure
adapter
api
eventbus
persistence
B_Package
applications
domain
infrastructure
adapter
api
persistence

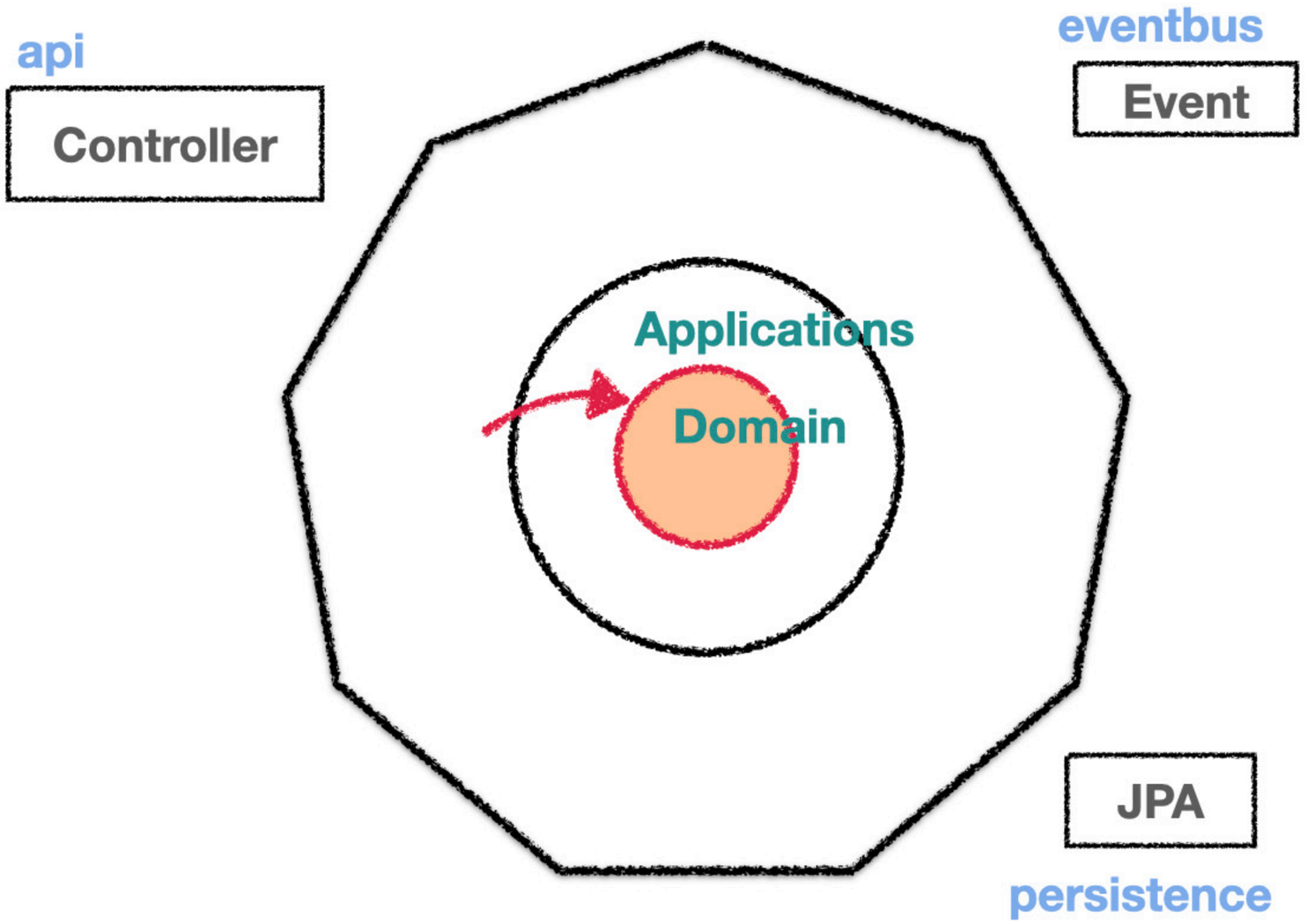

CLIENT


api
Controller
eventbus
Event
Applications
Domain
JPA
persistence

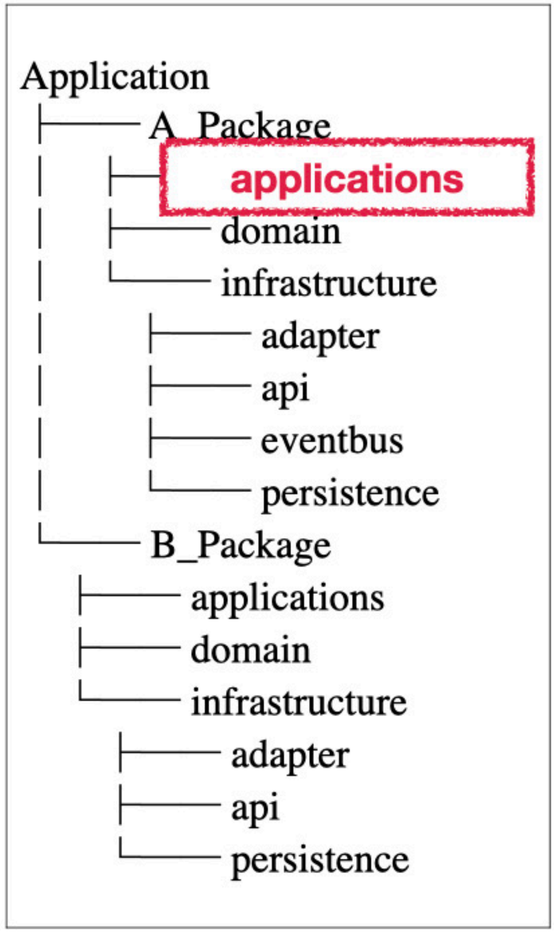
패키지 구성


Application
A_Package
applications
domain
infrastructure
adapter
api
eventbus
persistence
B_Package
applications
domain
infrastructure
adapter
api
persistence

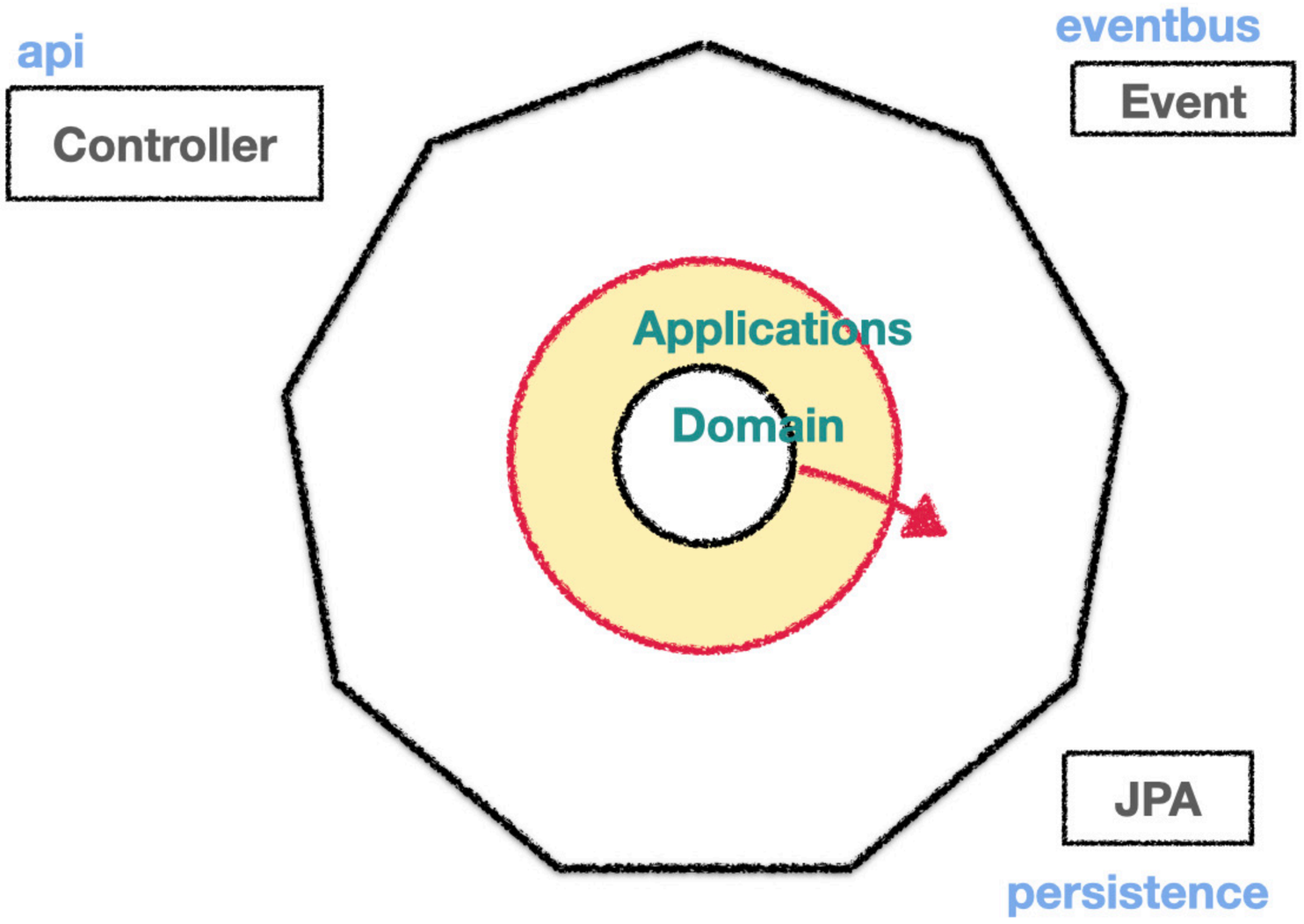

CLIENT


api
Controller
eventbus
Event
Applications
Domain
JPA
persistence

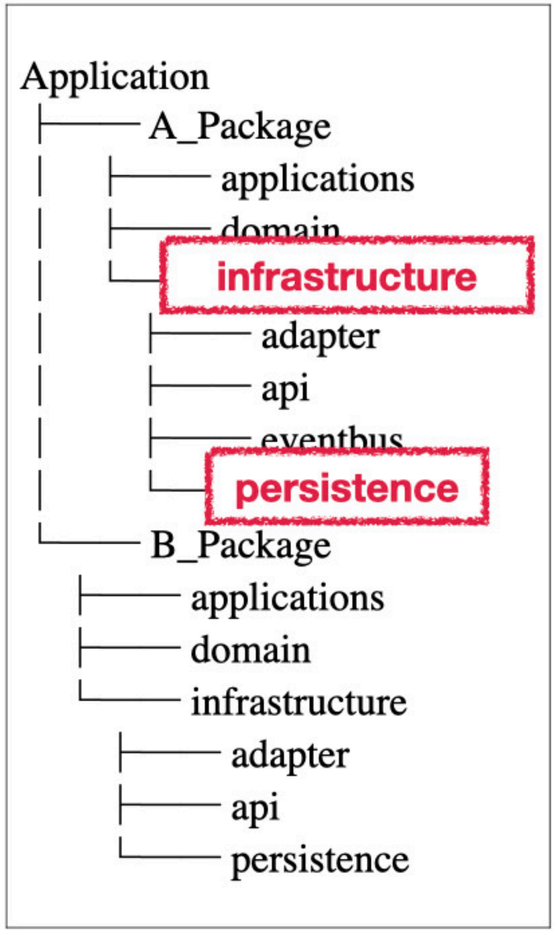
패키지 구성


Application
A_Package
applications
domain
infrastructure
adapter
api
eventbus
persistence
B_Package
applications
domain
infrastructure
adapter
api
persistence

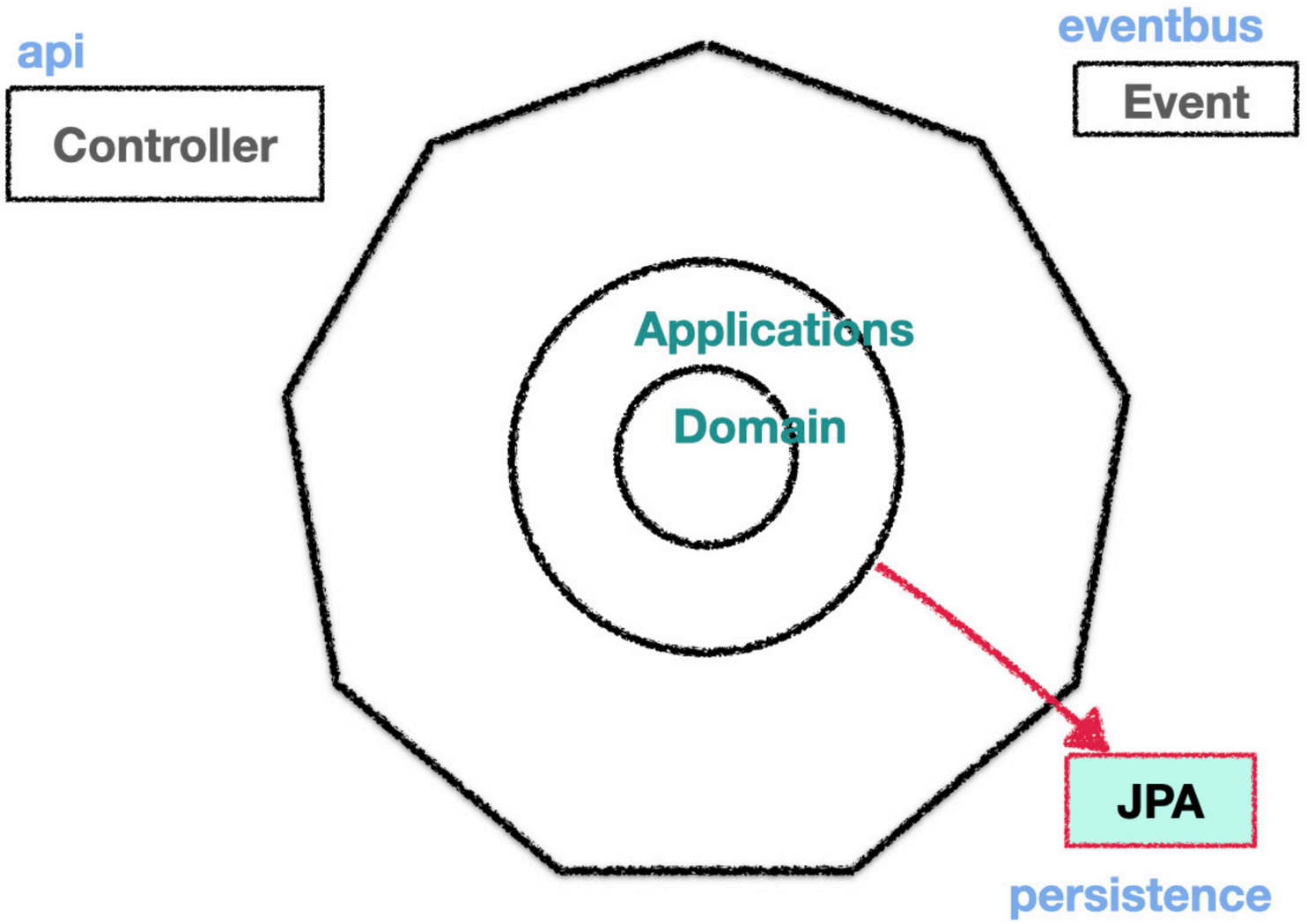

CLIENT


api
Controller
eventbus
Event
Applications
Domain
JPA
persistence

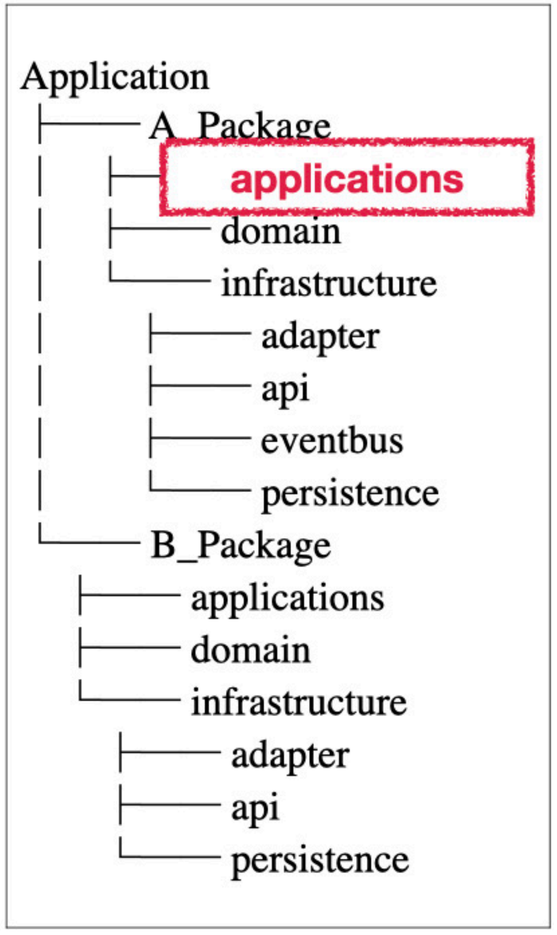
패키지 구성


Application
A_Package
applications
domain
infrastructure
adapter
api
eventbus
persistence
B_Package
applications
domain
infrastructure
adapter
api
persistence

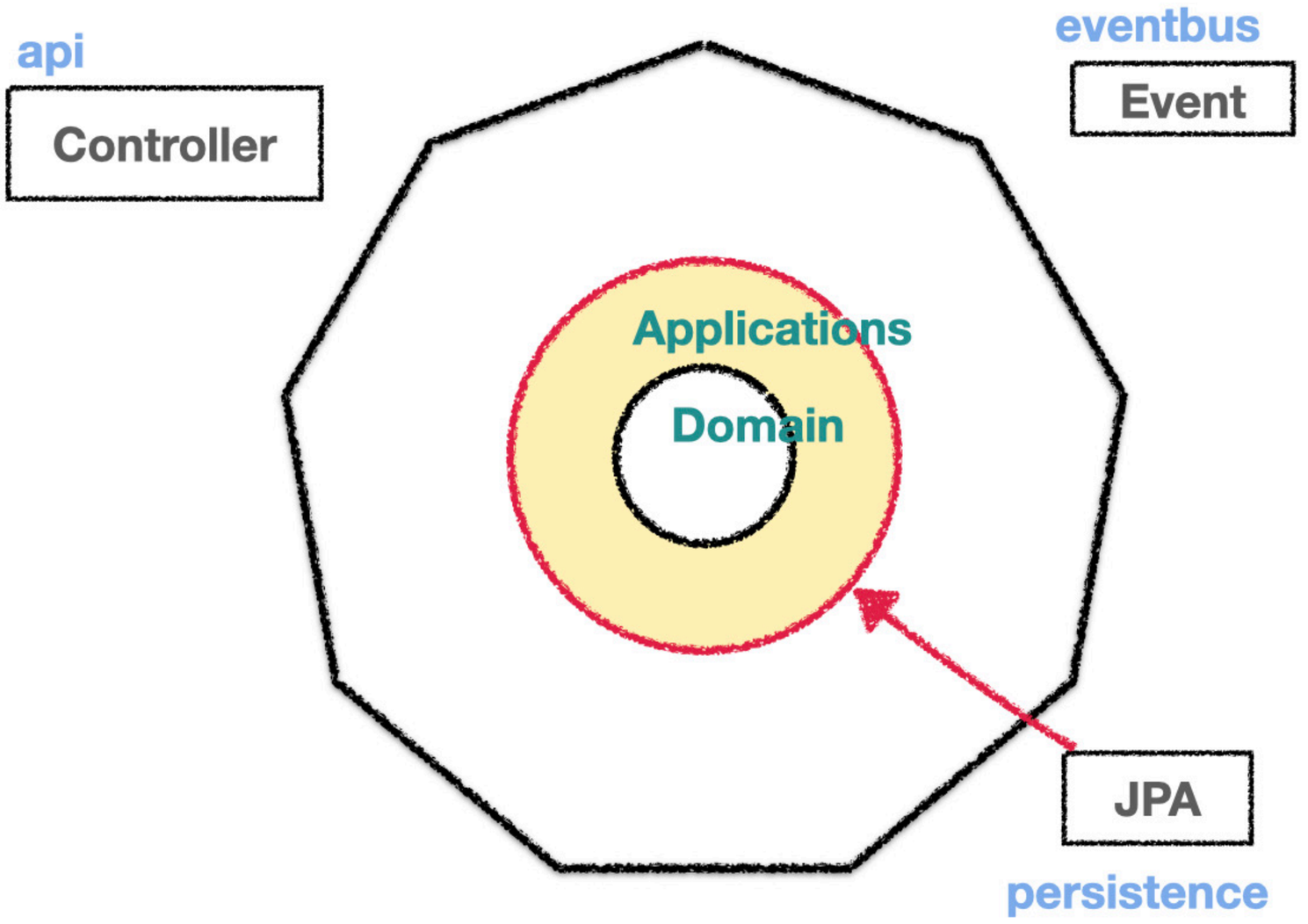

CLIENT


api
Controller
eventbus
Event
Applications
Domain
JPA
persistence

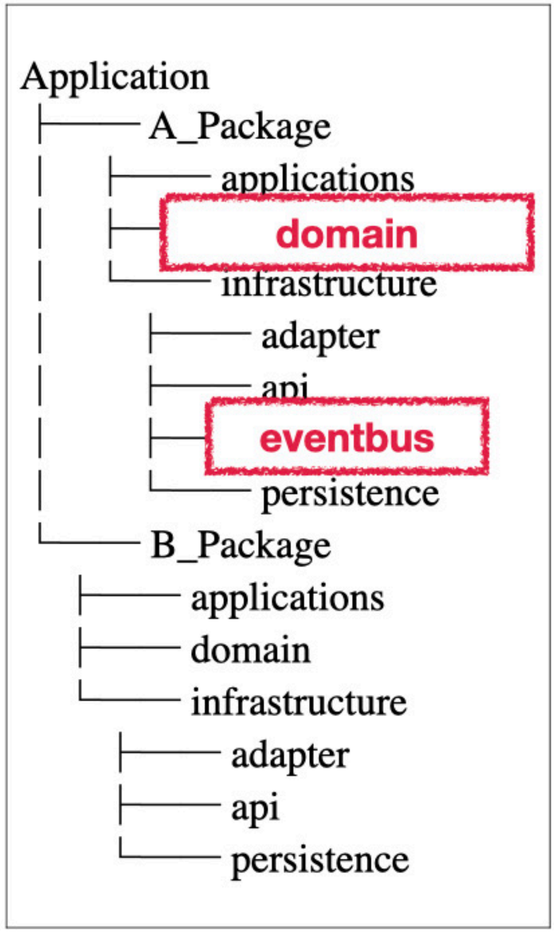
패키지 구성


Application
A_Package
applications
domain
infrastructure
adapter
api
eventbus
persistence
B_Package
applications
domain
infrastructure
adapter
api
persistence

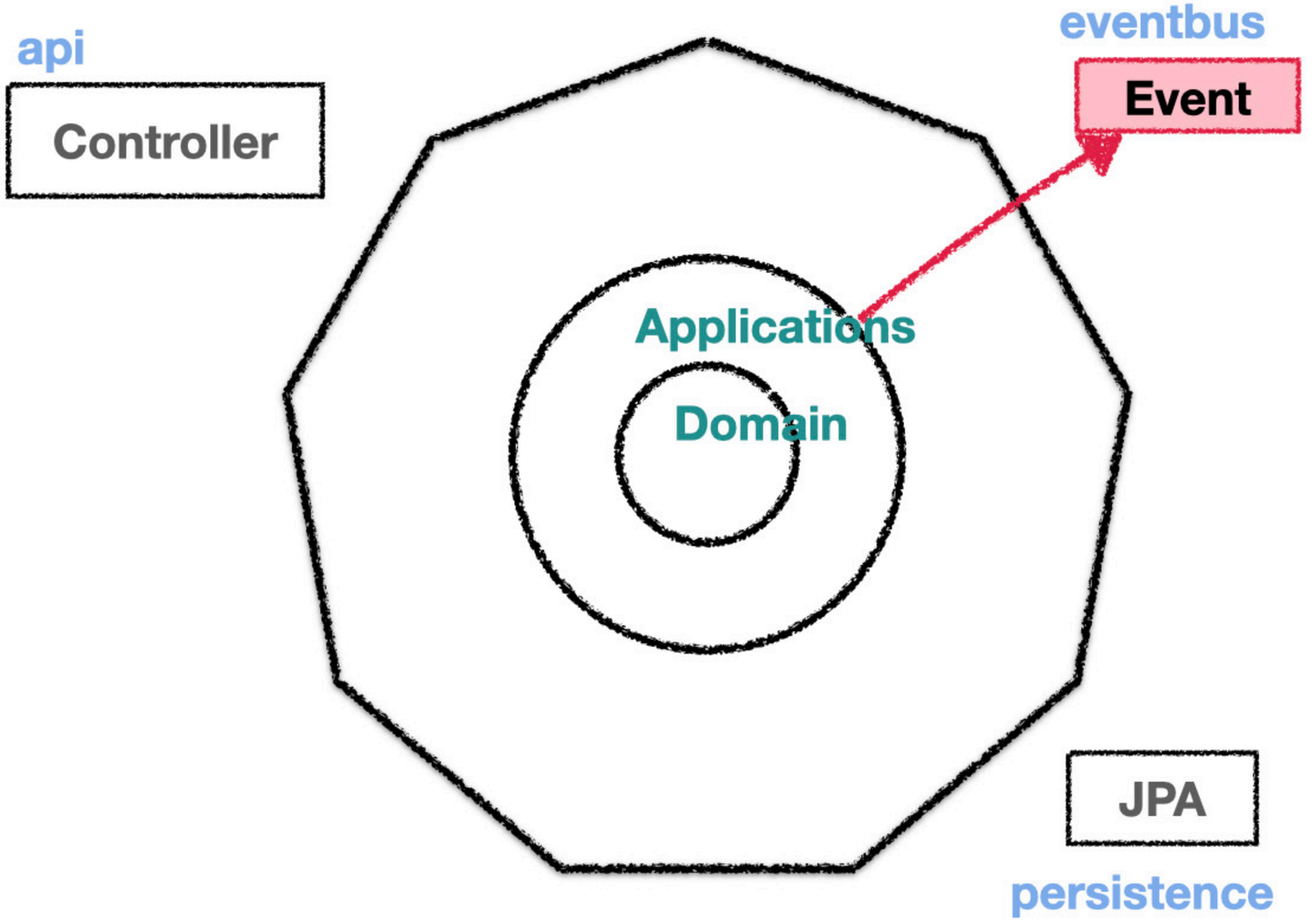

CLIENT


api
Controller
eventbus
Event
Applications
Domain
JPA
persistence

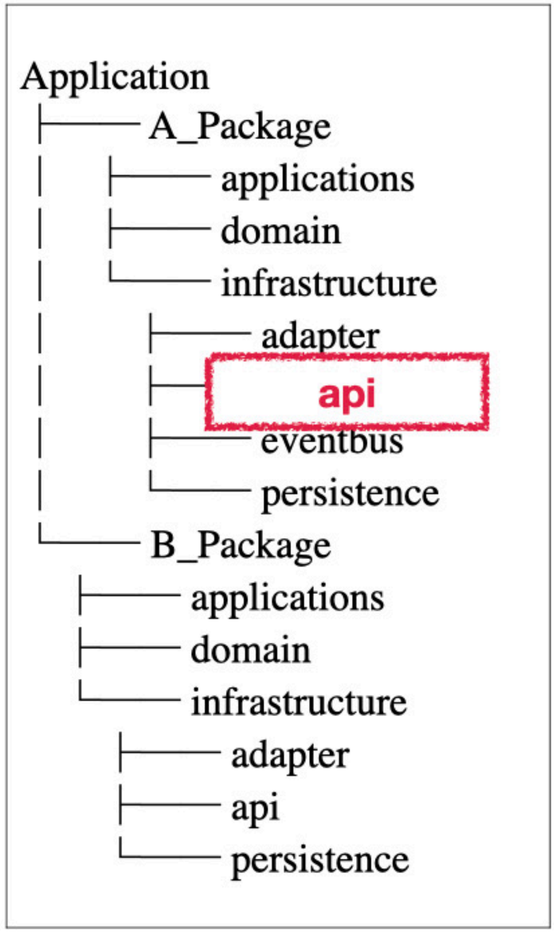
패키지 구성


Application
A_Package
applications
domain
infrastructure
adapter
api
eventbus
persistence
B_Package
applications
domain
infrastructure
adapter
api
persistence

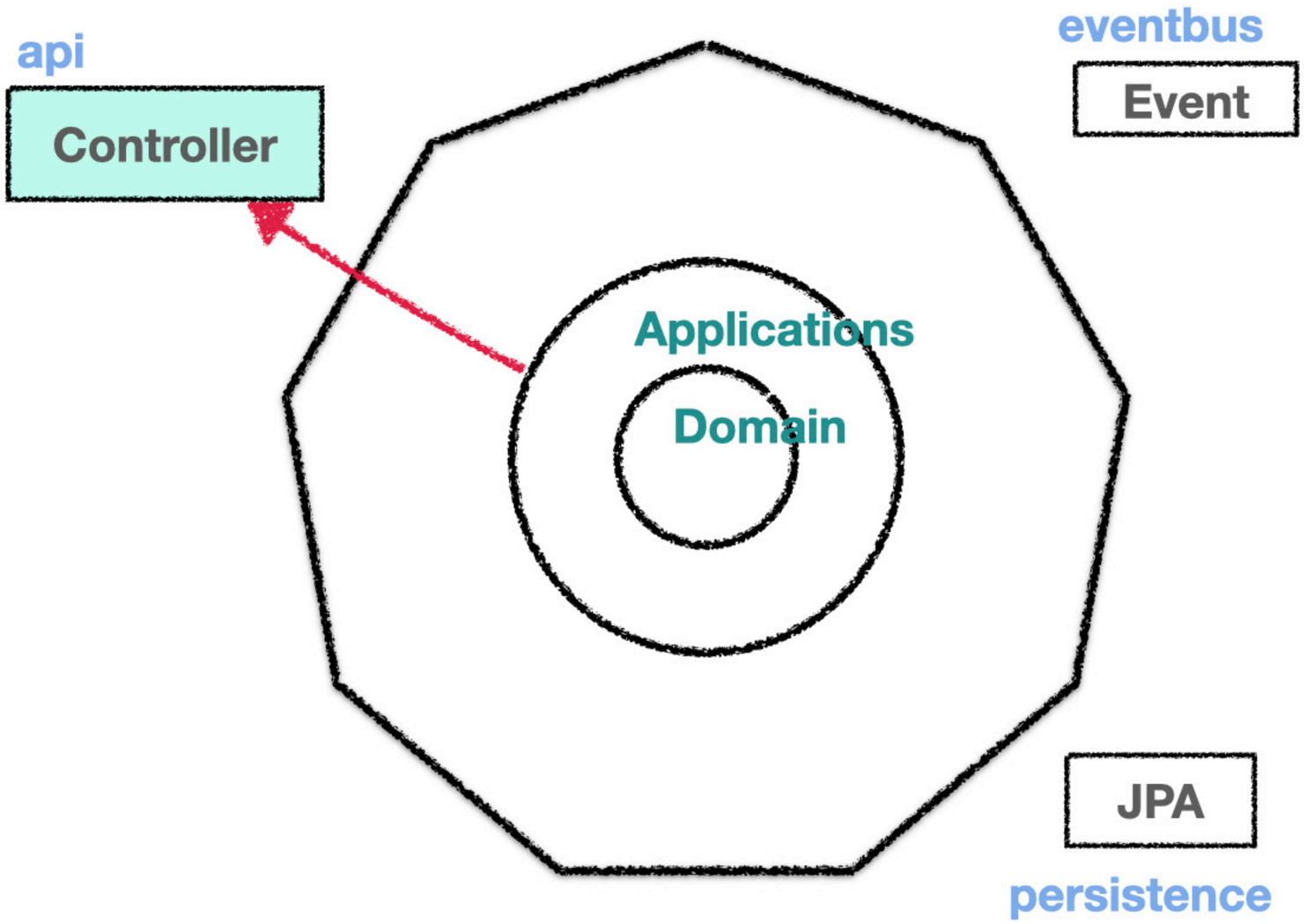

CLIENT


api
Controller
eventbus
Event
Applications
Domain
JPA
persistence

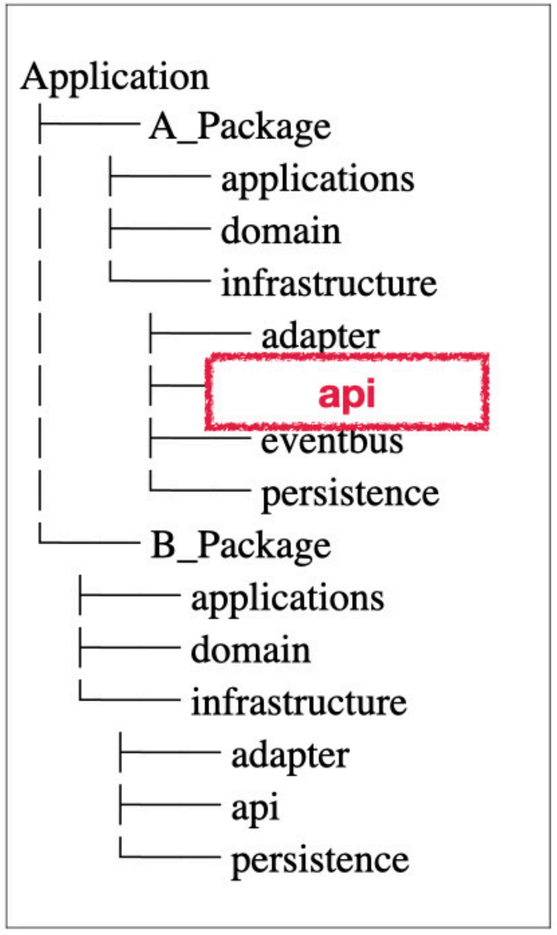
패키지 구성

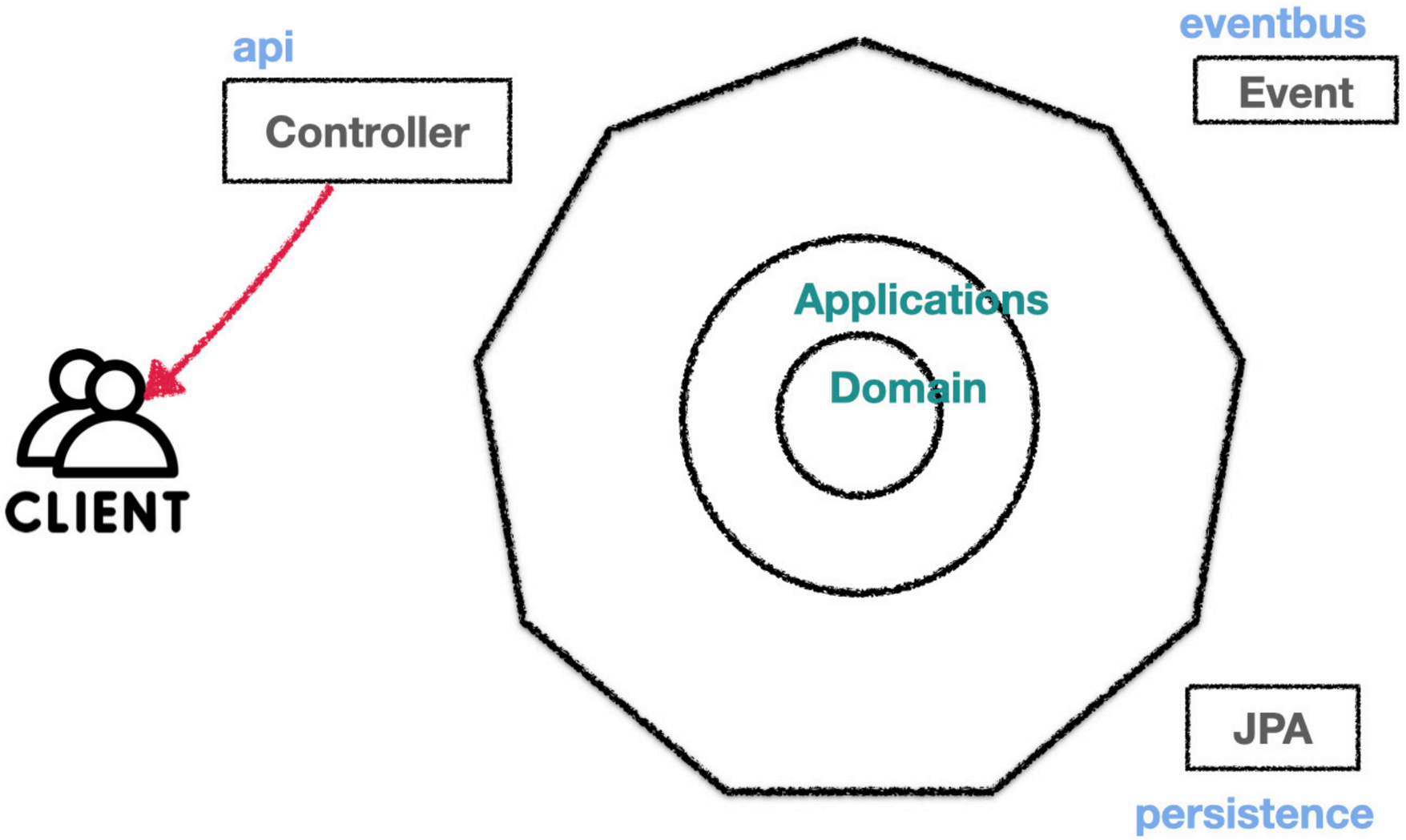

Application
A_Package
applications
domain
infrastructure
adapter
api
eventbus
persistence
B_Package
applications
domain
infrastructure
adapter
api
persistence


CLIENT


api
Controller
eventbus
Event
Applications
Domain
JPA
persistence

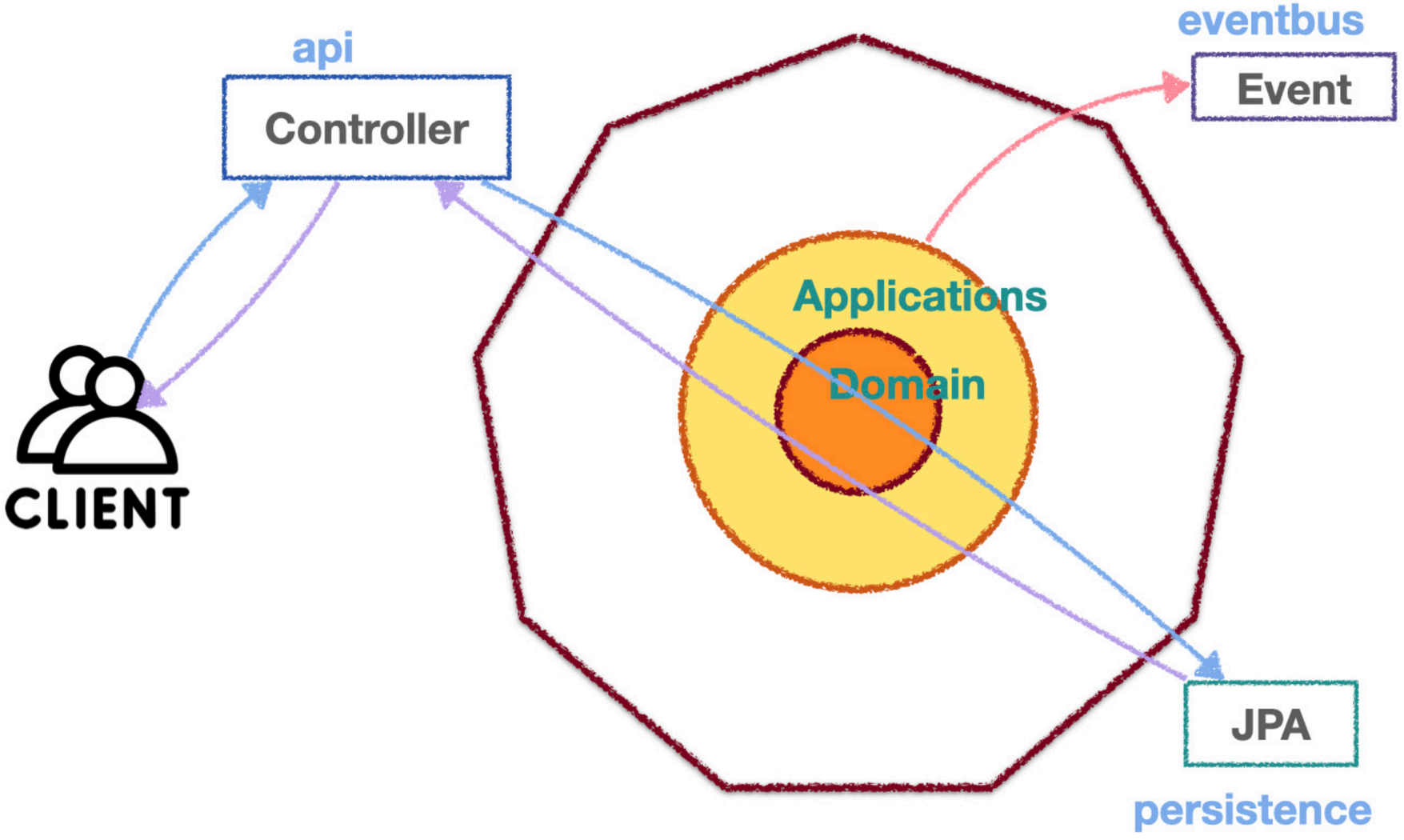
패키지 구성


Application
A_Package
applications
domain
infrastructure
adapter
api
eventbus
persistence
B_Package
applications
domain
infrastructure
adapter
api
persistence

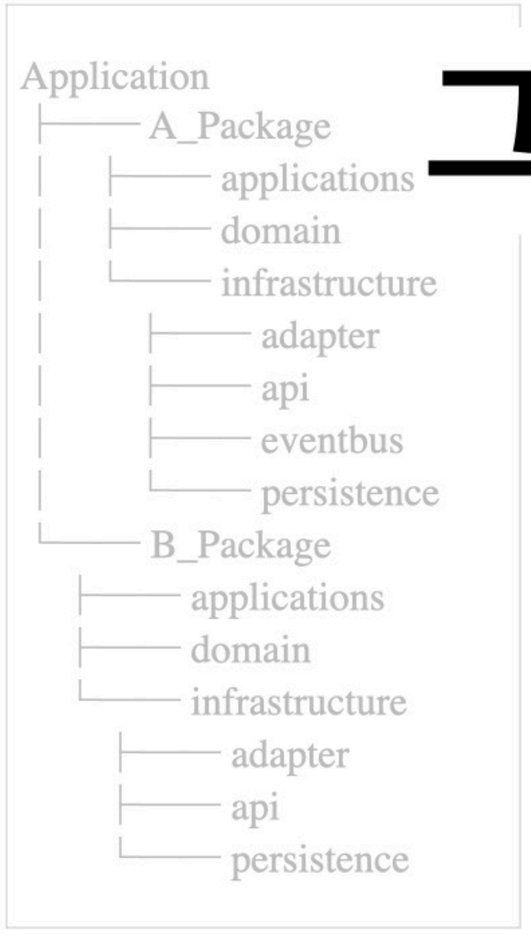

api
Controller
CLIENT
eventbus
Event
Applications
Domain
JPA
persistence

패키지 구성

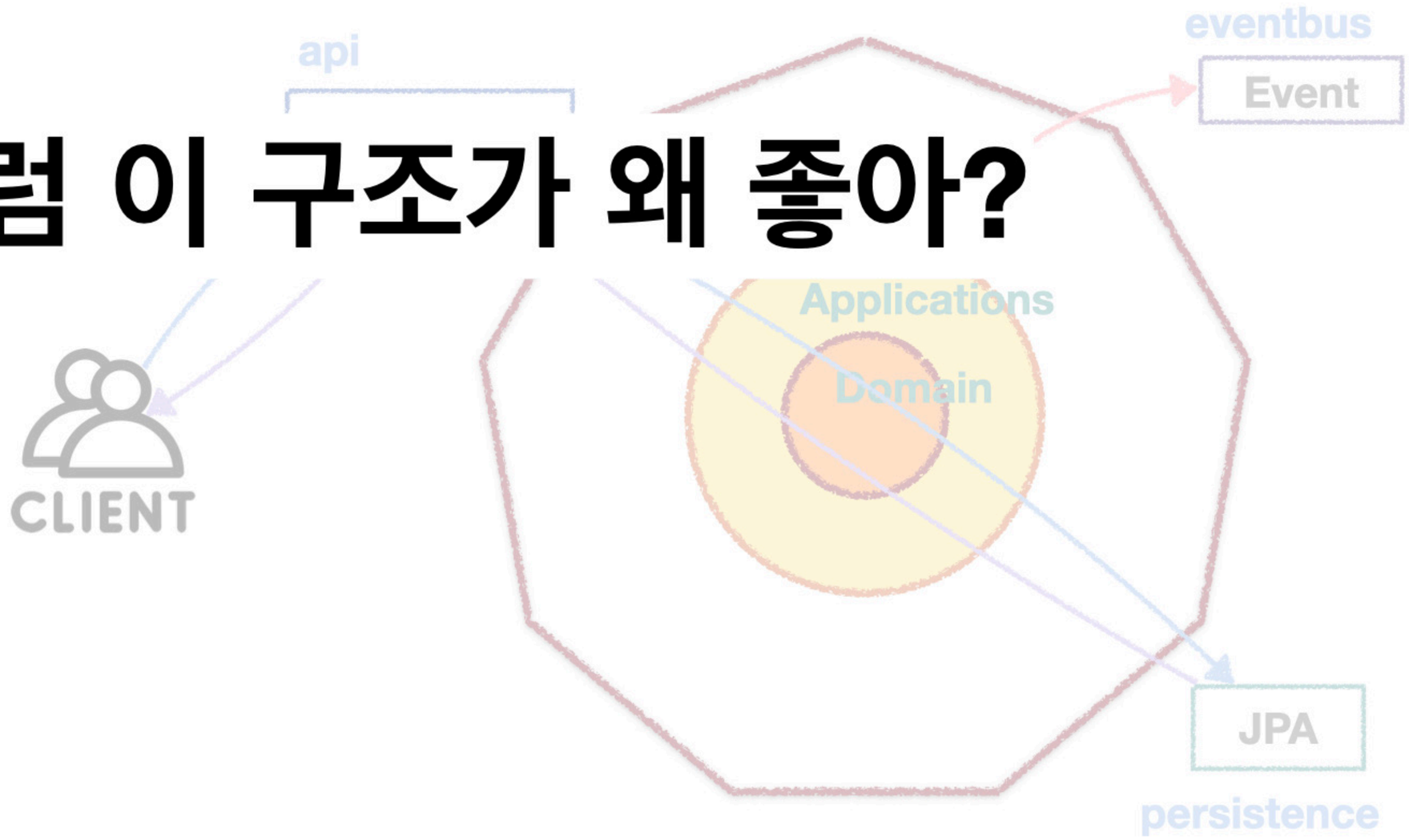
```
Application
├── A_Package
│   ├── applications
│   ├── domain
│   └── infrastructure
│       ├── adapter
│       ├── api
│       ├── eventbus
│       └── persistence
└── B_Package
    ├── applications
    ├── domain
    └── infrastructure
        ├── adapter
        ├── api
        └── persistence
```



패키지 구성

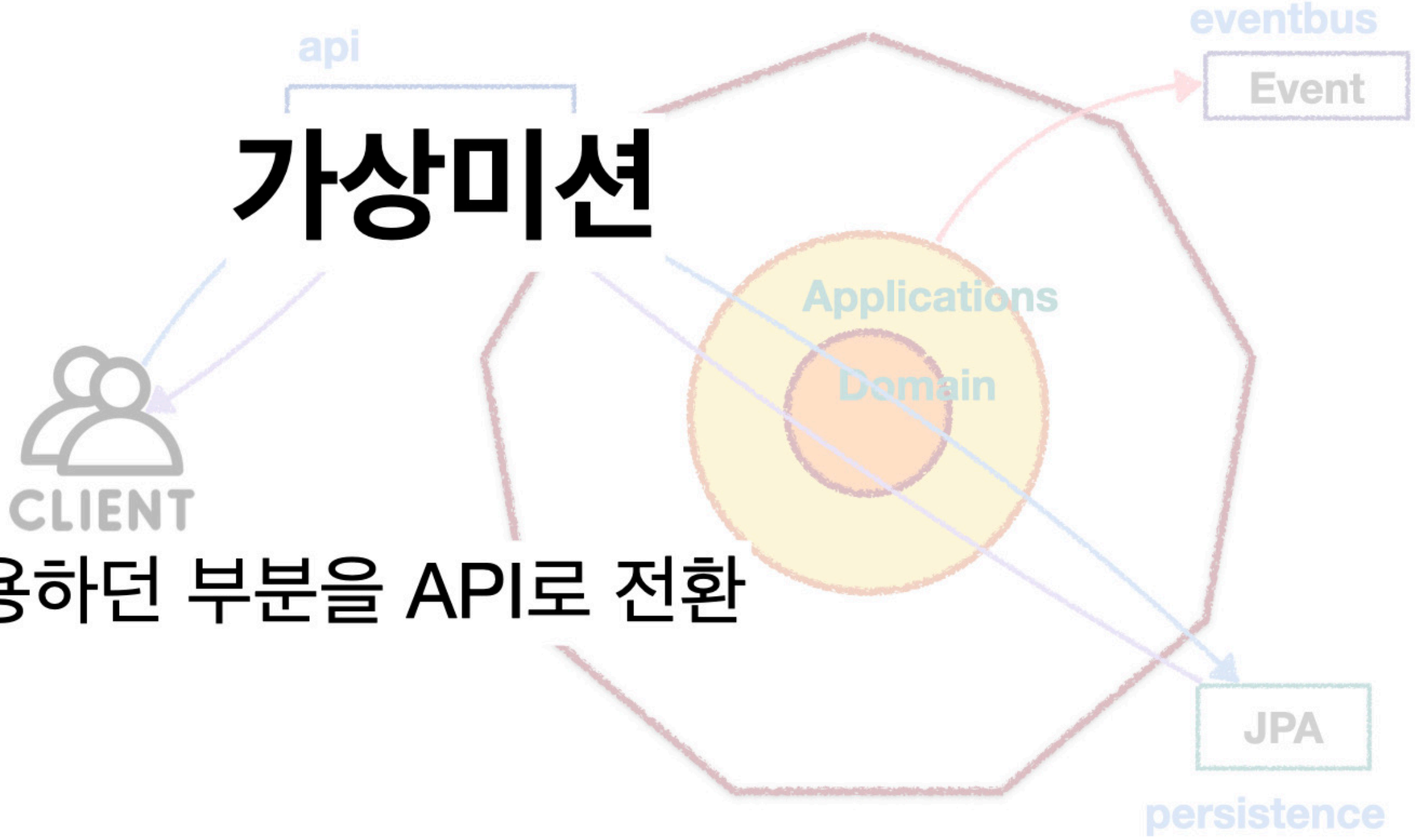


그럼 이 구조가 왜 좋아?



패키지 구성

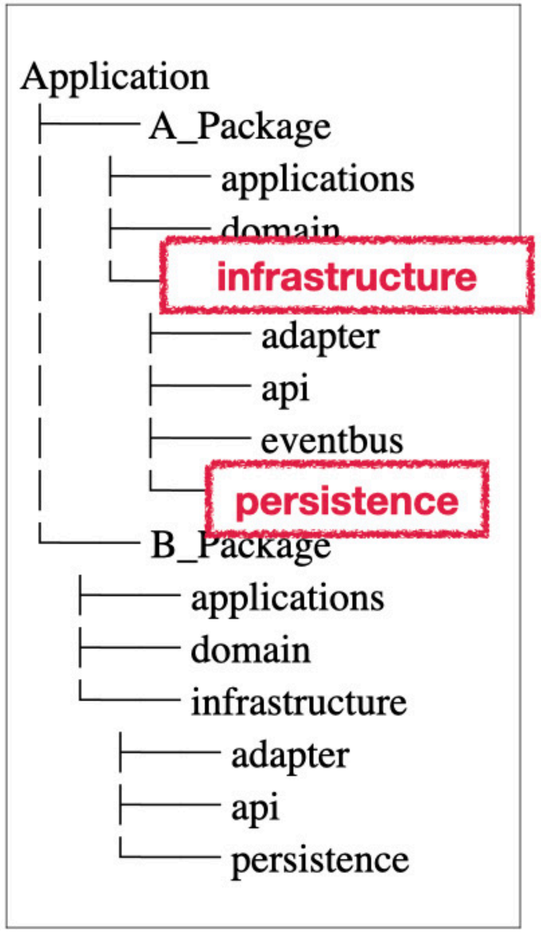
```
Application
├── A_Package
│   ├── applications
│   ├── domain
│   └── infrastructure
│       ├── adapter
│       ├── api
│       ├── eventbus
│       └── persistence
└── B_Package
    ├── ...
    ├── ...
    └── infrastructure
        ├── adapter
        ├── api
        └── persistence
```

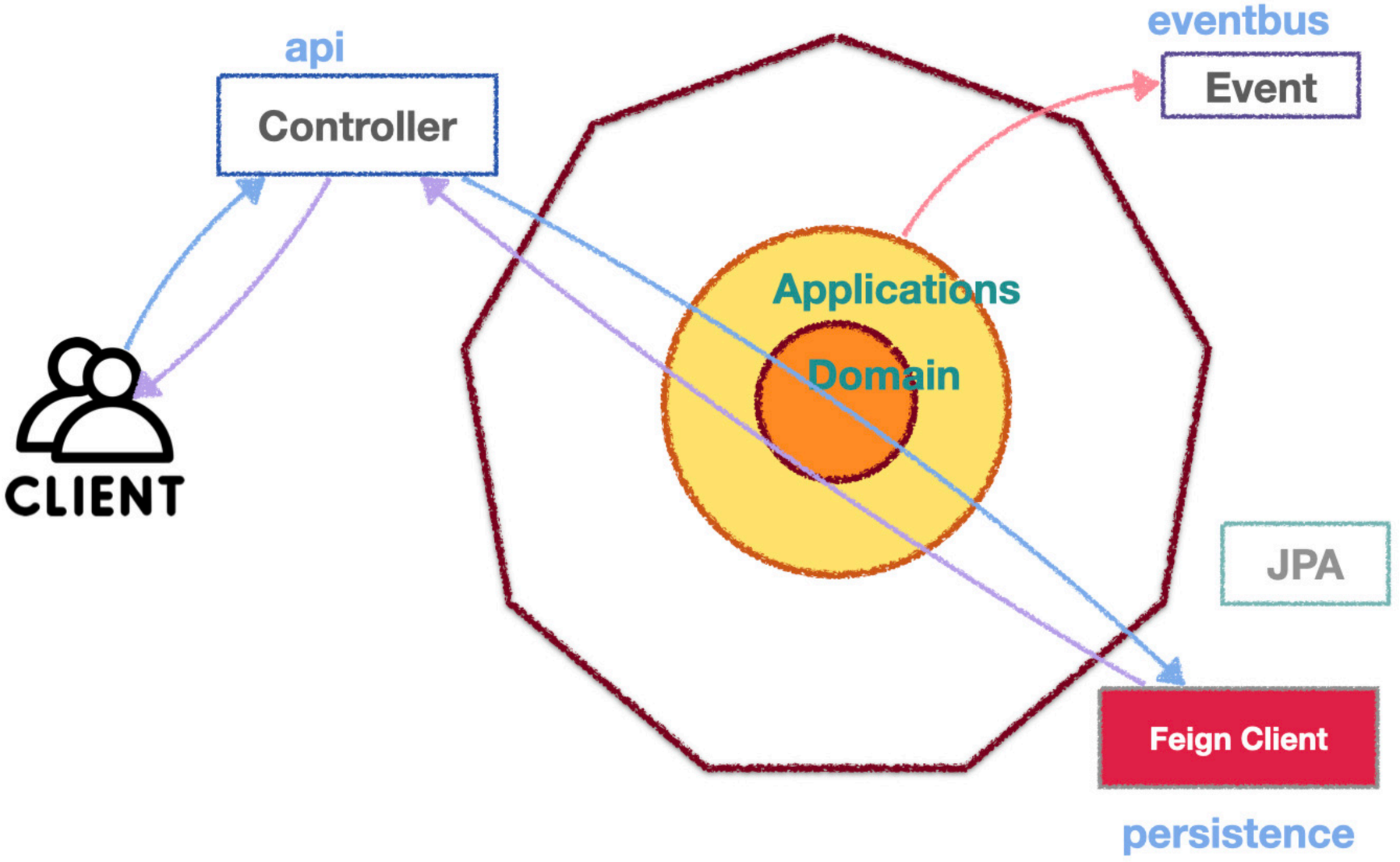


가상미션

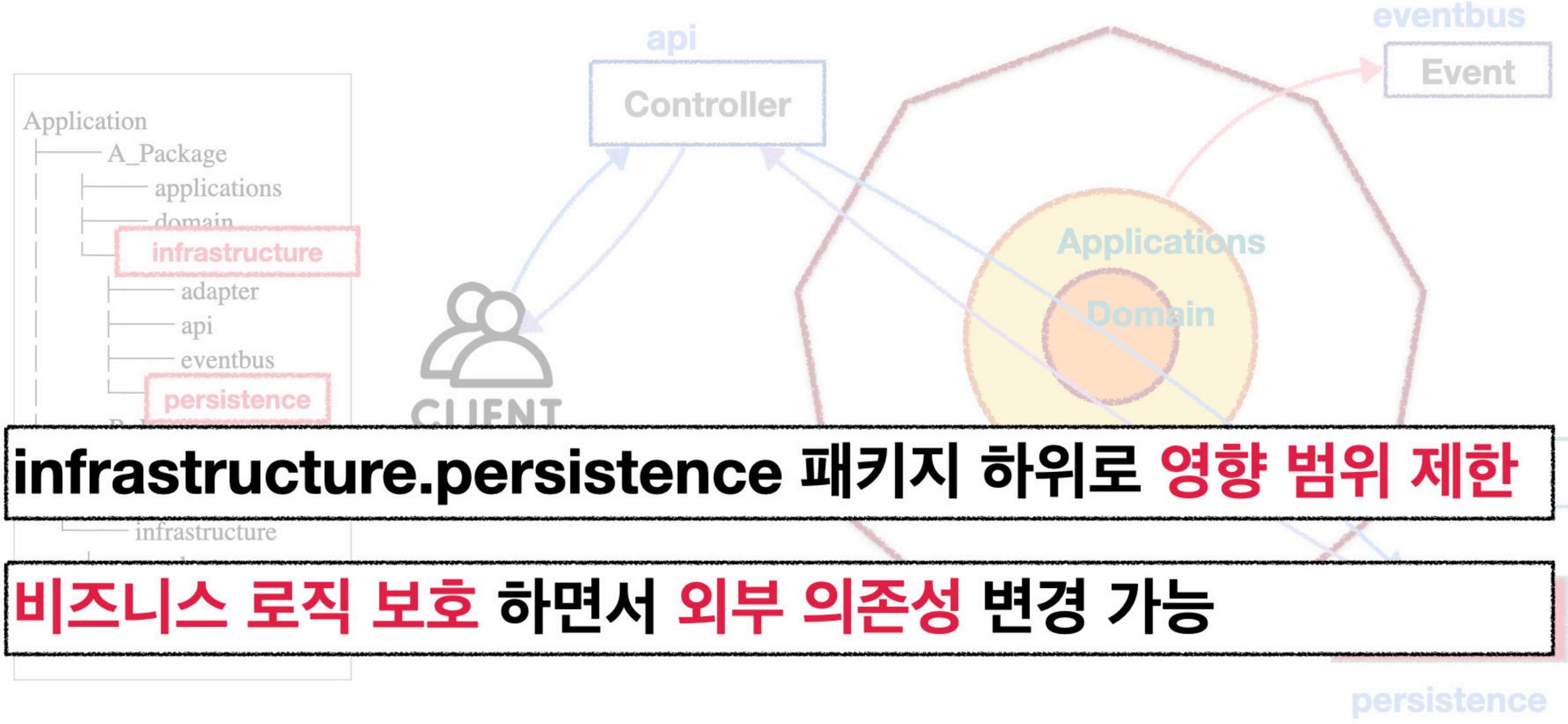
- DB를 사용하던 부분을 API로 전환

패키지 구성


Application
A_Package
applications
domain
infrastructure
adapter
api
eventbus
persistence
B_Package
applications
domain
infrastructure
adapter
api
persistence


api
Controller
eventbus
Event
CLIENT
Applications
Domain
JPA
Feign Client
persistence

패키지 구성



infrastructure.persistence 패키지 하위로 영향 범위 제한

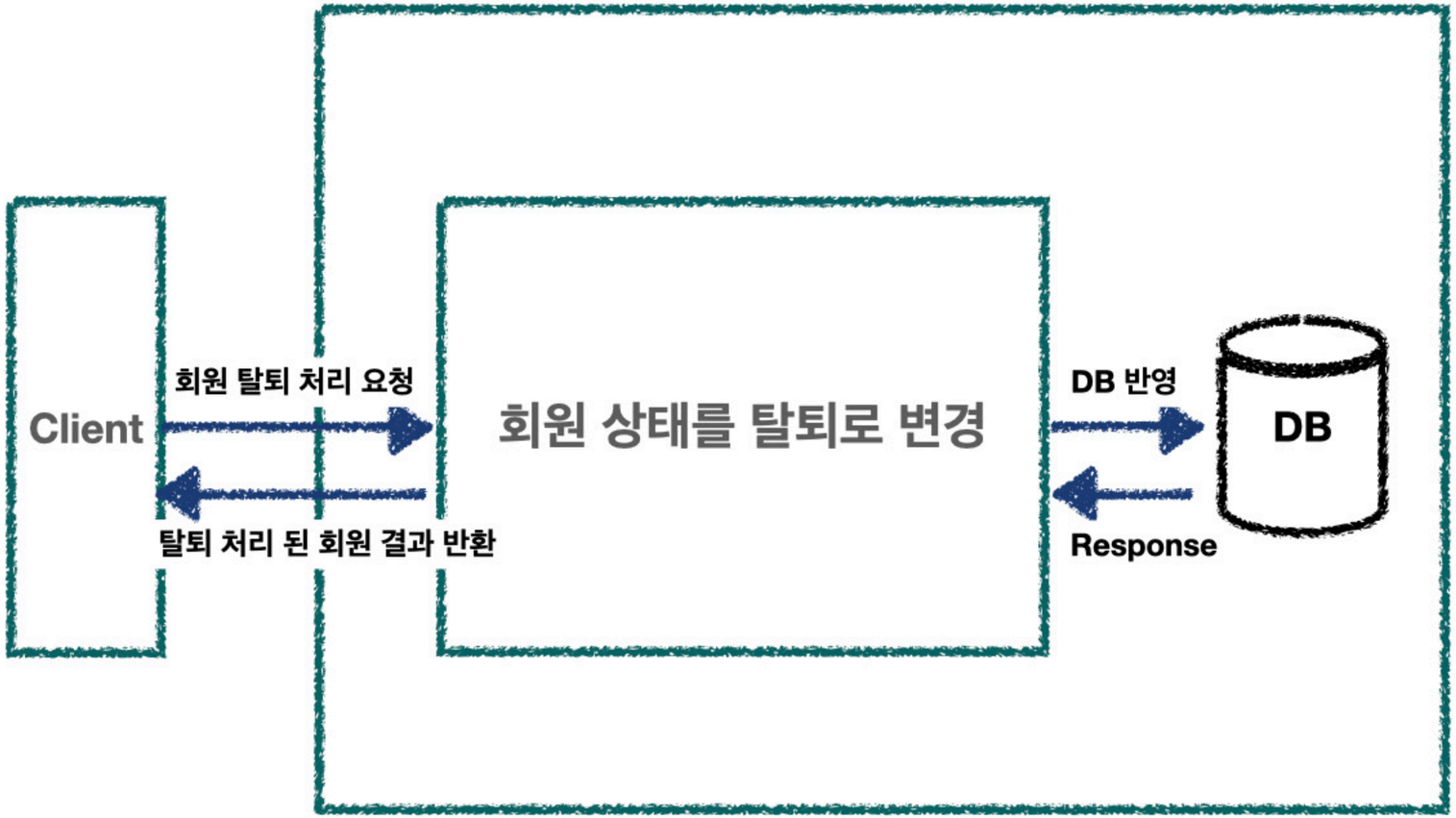
비즈니스 로직 보호 하면서 외부 의존성 변경 가능

이벤트 구성

요구사항

- 판매자의 회원 탈퇴 절차

이벤트 구성



회원 상태 변경 후 회원 DB에 반영

서브 요구사항

- <회원이 탈퇴 되면> 상품을 내려주세요.
- <회원이 탈퇴 되면> 계좌를 닫아주세요.
- <회원이 탈퇴 되면> A 서비스와의 연동을 해지해주세요.

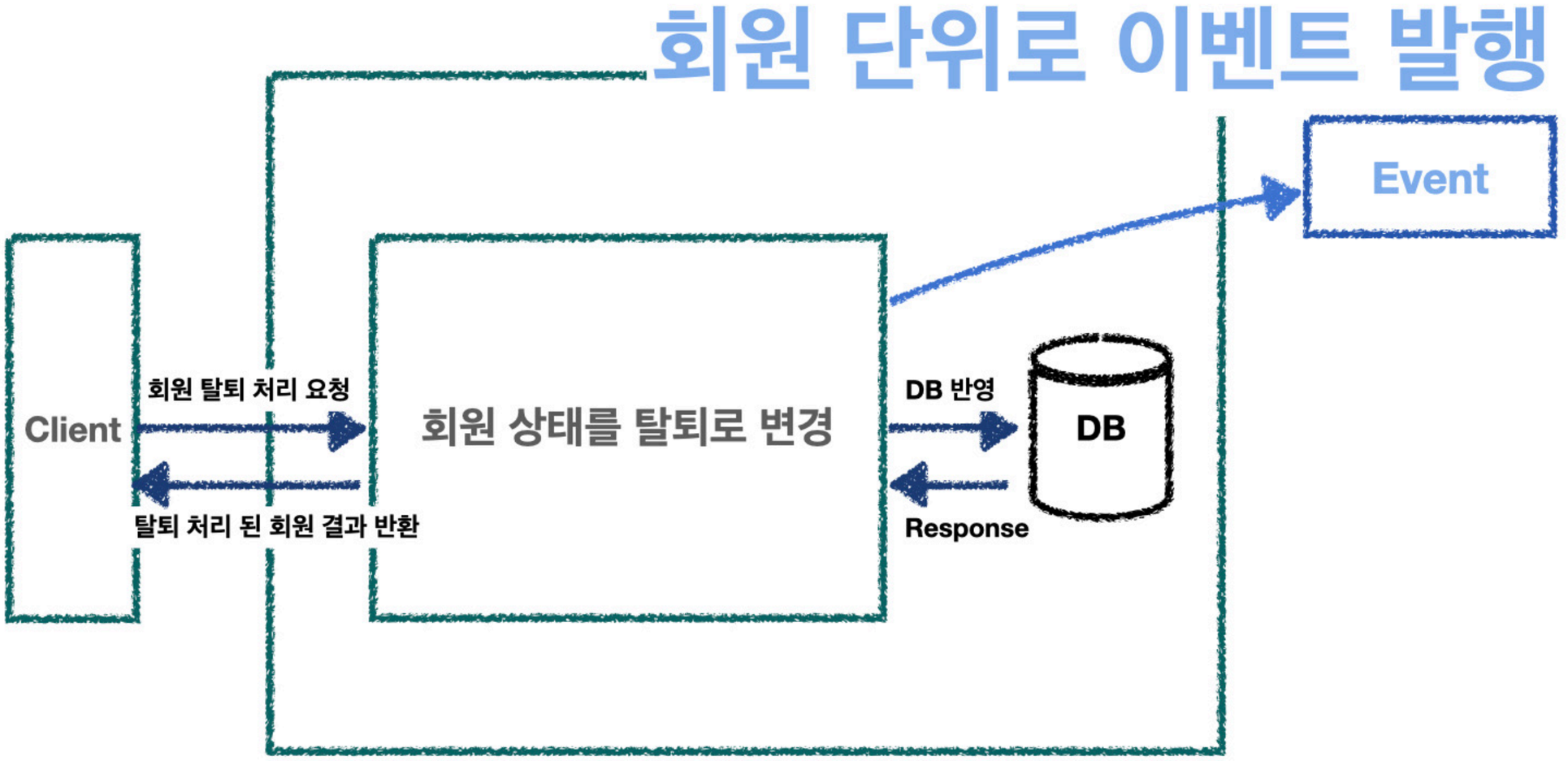
이벤트 구성

서브 요구사항

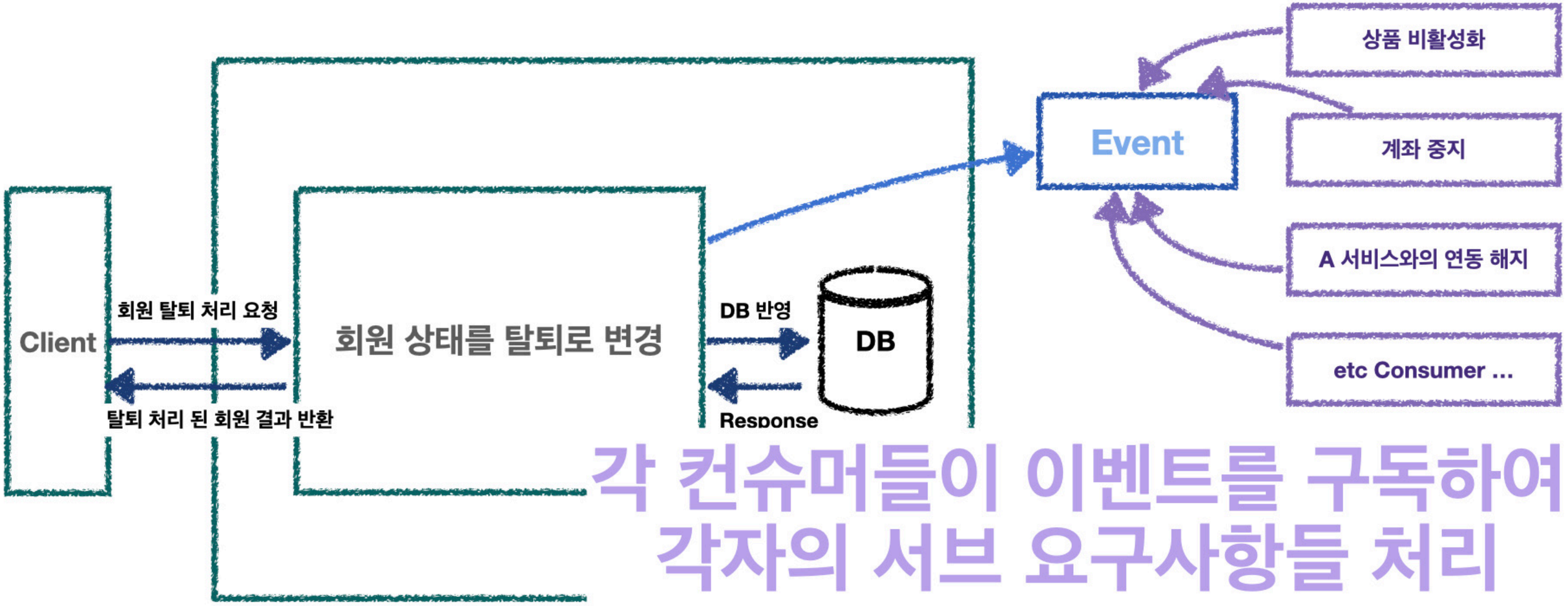
N 개의 추가 요구사항

- <회원
- <회원
- <회원이 탈퇴 되면> A 서비스와의 연동을 해지해주세요.

이벤트 구성

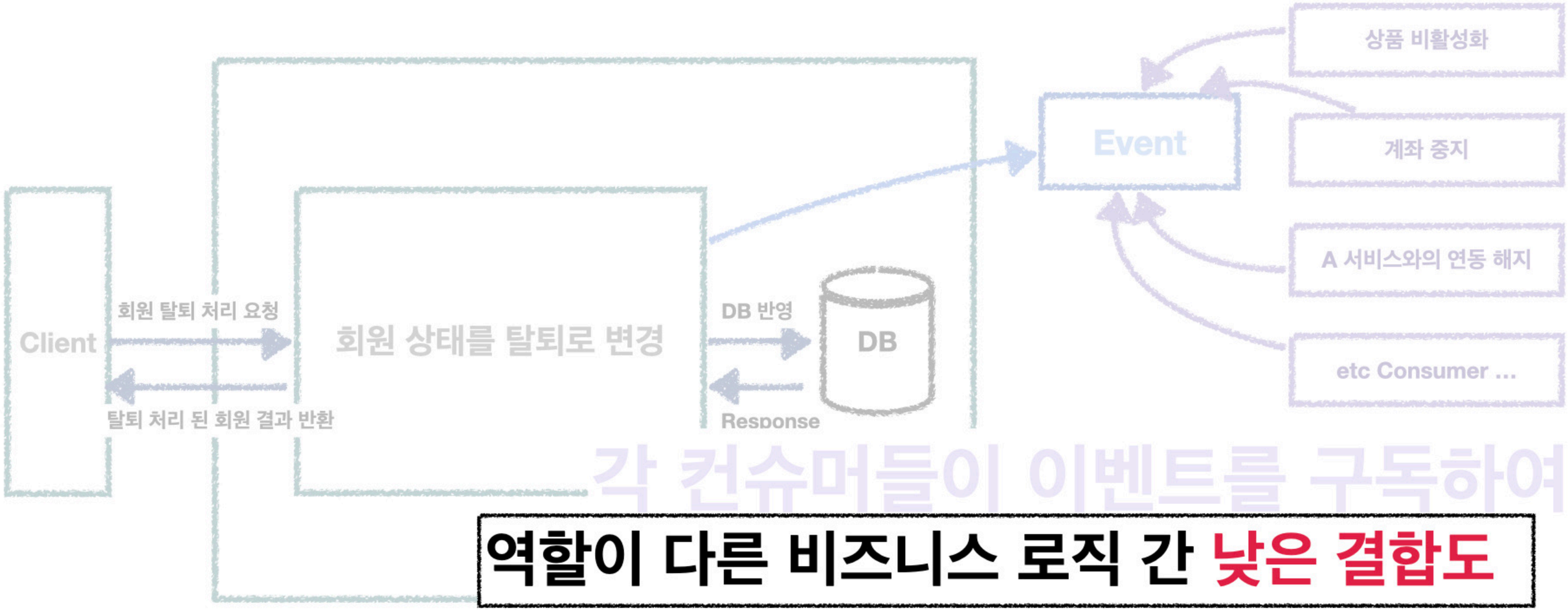

회원 단위로 이벤트 발행
Event
Client
회원 탈퇴 처리 요청
회원 상태를 탈퇴로 변경
DB 반영
DB
탈퇴 처리 된 회원 결과 반환
Response

이벤트 구성



각 컨슈머들이 이벤트를 구독하여
각자의 서브 요구사항들 처리

이벤트 구성



역할이 다른 비즈니스 로직 간 낮은 결합도

도메인 구성

요구사항

- 1년 동안 로그인 하지 않은 회원을 철회 상태로 변경

도메인 구성

빈곤한(Anemic) 도메인

```
    class Service {
        Member withdraw(String identifier) {
            Member member =
repository.getMember(identifier);
            member.setStatus(Status.Withdraw);
            member.setWithDrawalDateTime(OffsetDateTime.now
());
            return repository.save(member);
        }
    }
```

```
    class Member {
        void setStatus(String status) {
            this.status = status;
        }

        void setWithDrawalDateTime(OffsetDateTime
withDrawalDateTime) {
            this.withDrawalDateTime = withDrawalDateTime;
        }
    }
```

도메인 구성

빈곤한(Anemic) 도메인

```
    class Service {
        Member withdraw(String identifier) {
            Member member =
repository.getMember(identifier);
            member.setStatus(Status.Withdraw);
            member.setWithDrawalDateTime(OffsetDateTime.now())
    }
```

```
    class Member {
        void setStatus(String status) {
            this.status = status;
        }

        void setWithDrawalDateTime(OffsetDateTime
withDrawalDateTime) {
            this.withDrawalDateTime = withDrawalDateTime;
        }
    }
```

도메인 구성

빈곤한(Anemic) 도메인

```
    class Service {
        Member withdraw(String identifier) {
            Member member =
repository.getMember(identifier);
            member.setStatus(Status.Withdraw);
            member.setWithDrawalDateTime(OffsetDateTime.now
());
            return repository.save(member);
        }
    }
```

```
    class Member {
        void setStatus(String status) {
            this.status = status;
        }

        void setWithDrawalDateTime(OffsetDateTime
withDrawalDateTime) {
            this.withDrawalDateTime = withDrawalDateTime;
        }
    }
```

도메인 구성

빈곤한(Anemic) 도메인

```
class Service {
    Member withdraw(String identifier) {
        Member member =
repository.getMember(identifier);
        member.setStatus(Status.Withdraw);
        member.setWithDrawalDateTime(OffsetDateTime.now
());
        return repository.save(member);
    }
}
```

```
class Membe
    void se
        thi
    }

    void
withDrawalDat
        th
    }
}
```

Set, Get 만 존재하는 도메인 객체

비즈니스 로직은 서비스 객체에 존재

도메인 구성

풍성한(Rich) 도메인

```
    class Service {
        Member withdraw(String identifier) {
            Member member =
repository.getMember(identifier);
            return repository.save(member.withdraw());
        }
    }
```

```
class Member {
    void withdraw() {
        status = Status.Withdraw
        withdrawalDateTime = OffsetDateTime.now()
    }
}
```

도메인 구성

풍성한(Rich) 도메인

```
class Service {
    Member withdraw(String identifier) {
        Member member =
repository.getMember(identifier);
return repository.save(member.withdraw());
```


```
class Member {
void withdraw() {
   status = Status.Withdraw;
   withdrawalDateTime = OffsetDateTime.now();
}
```

도메인 구성

풍성한(Rich) 도메인

```
    class Service {
        Member withdraw(String identifier) {
            Member member =
repository.getMember(identifier);
            return repository.save(member.withdraw());
        }
    }
```

```
class Member {
    void withdraw() {
        status = Status.Withdraw
        member.withdrawalDateTime = OffsetDateTime.now()
    }
}
```

도메인에 관련된 개념, 관계 및 비즈니스 규칙 풀소유

도메인 구성

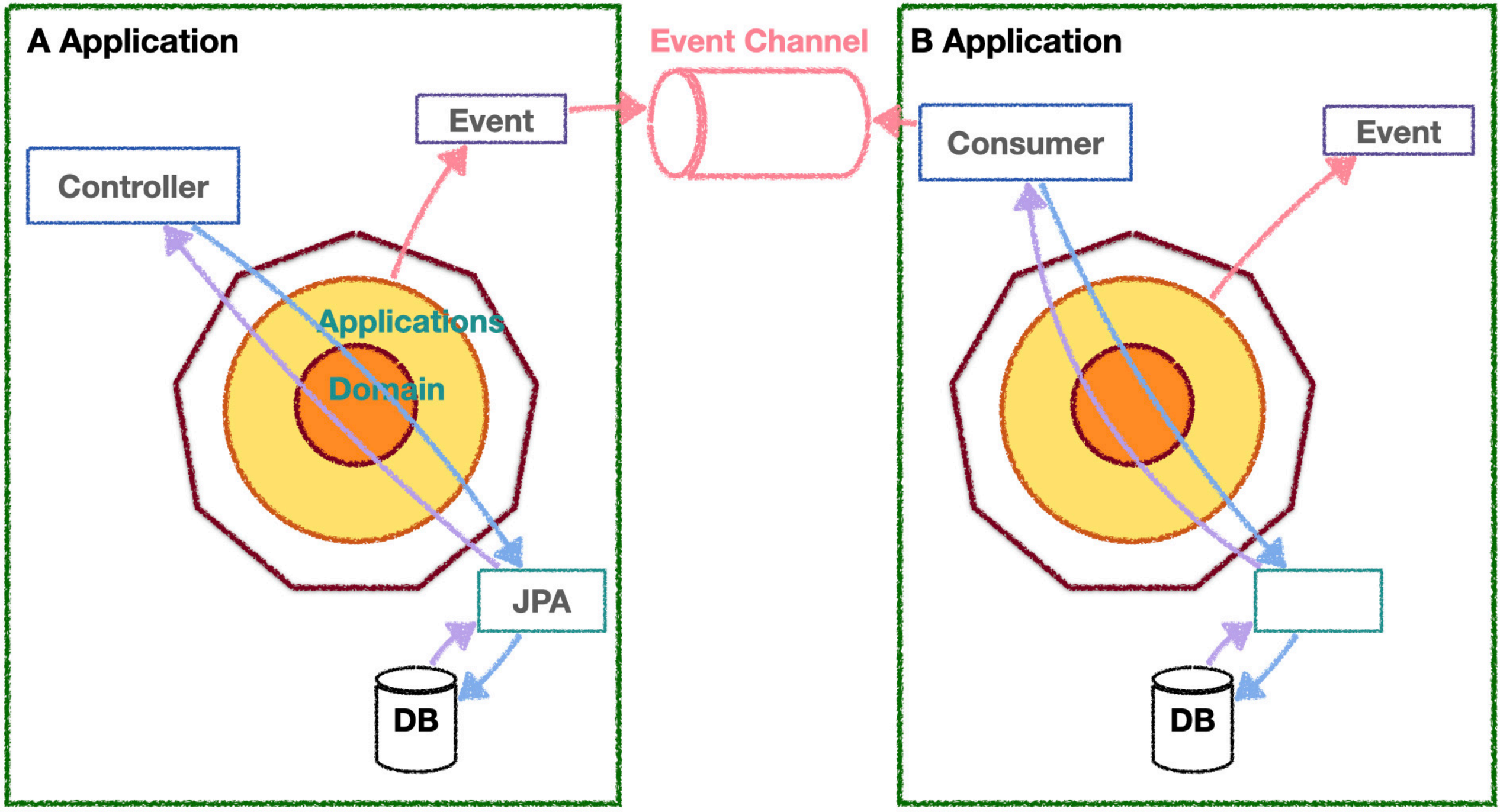
풍성한(Rich) 도메인

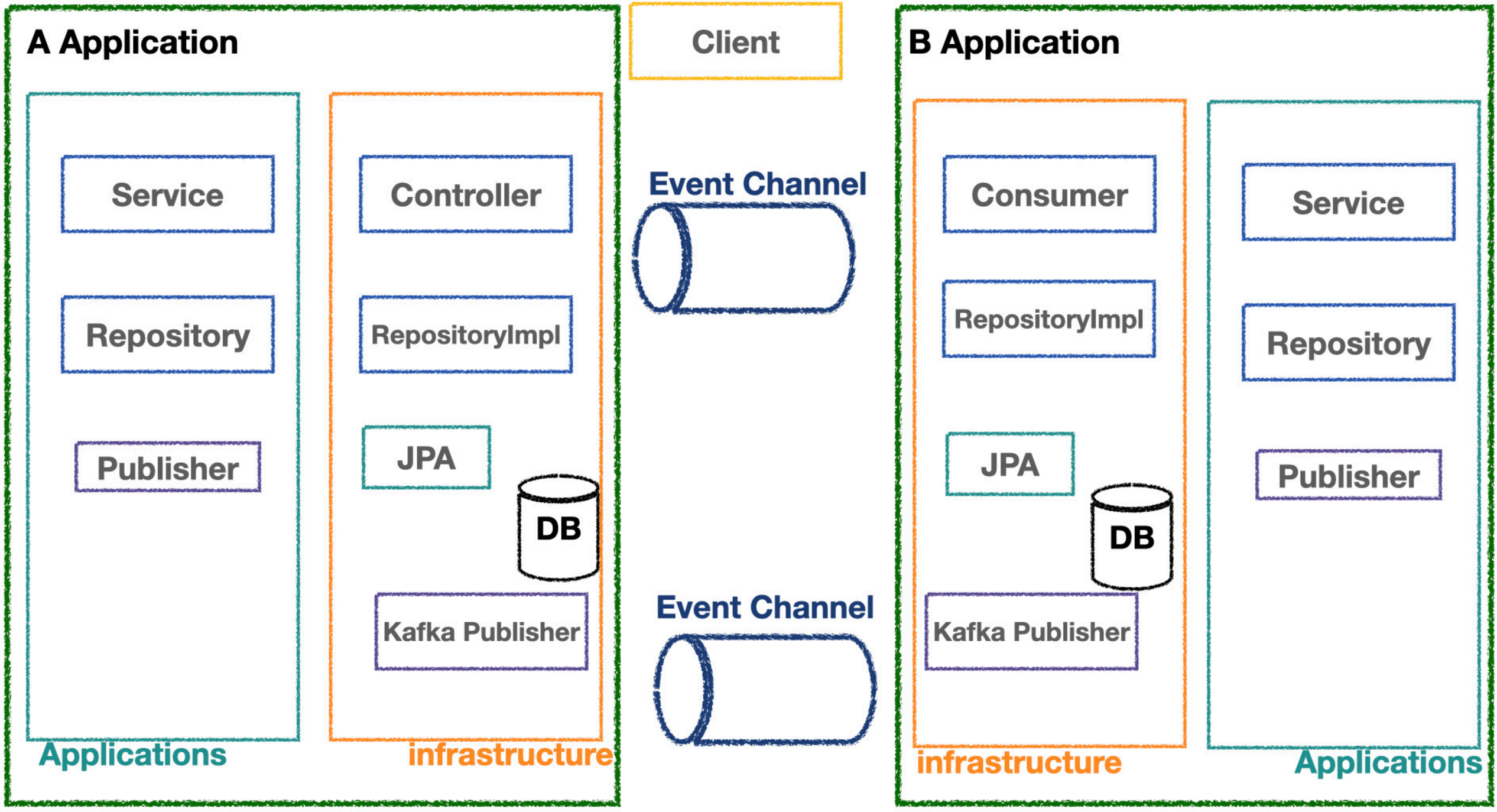
```
    class Service {
        Member withdraw(String identifier) {
            Member member =
repository.getMember(identifier);
            return repository.save(member.withdraw());
        }
    }
```

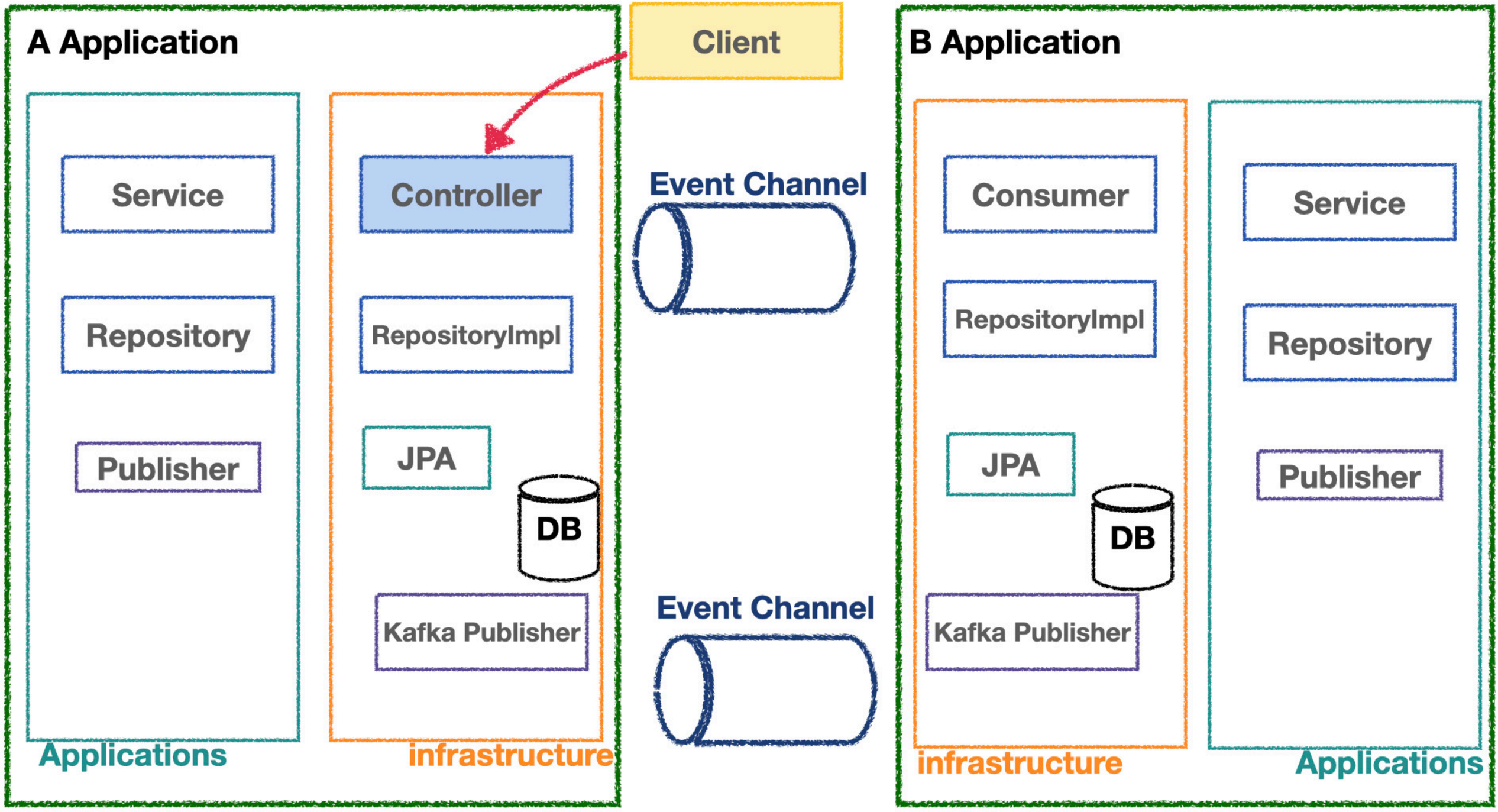
```
class Member {
    void withdraw() {
        status = Status.Withdraw
        member.withdrawalDateTime = OffsetDateTime.now()
    }
}
```

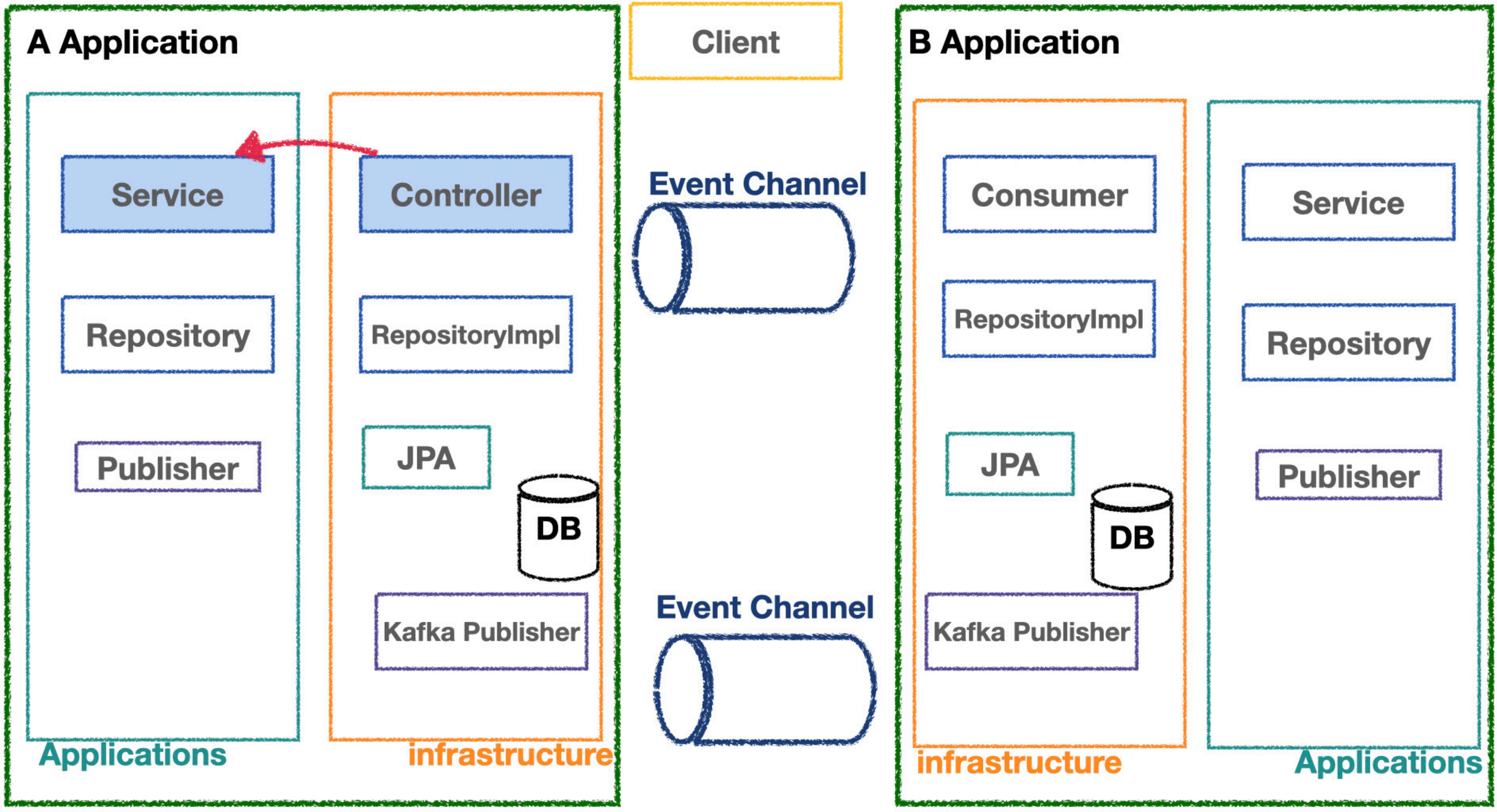
철회에 대한 행위를 객체에게 위임

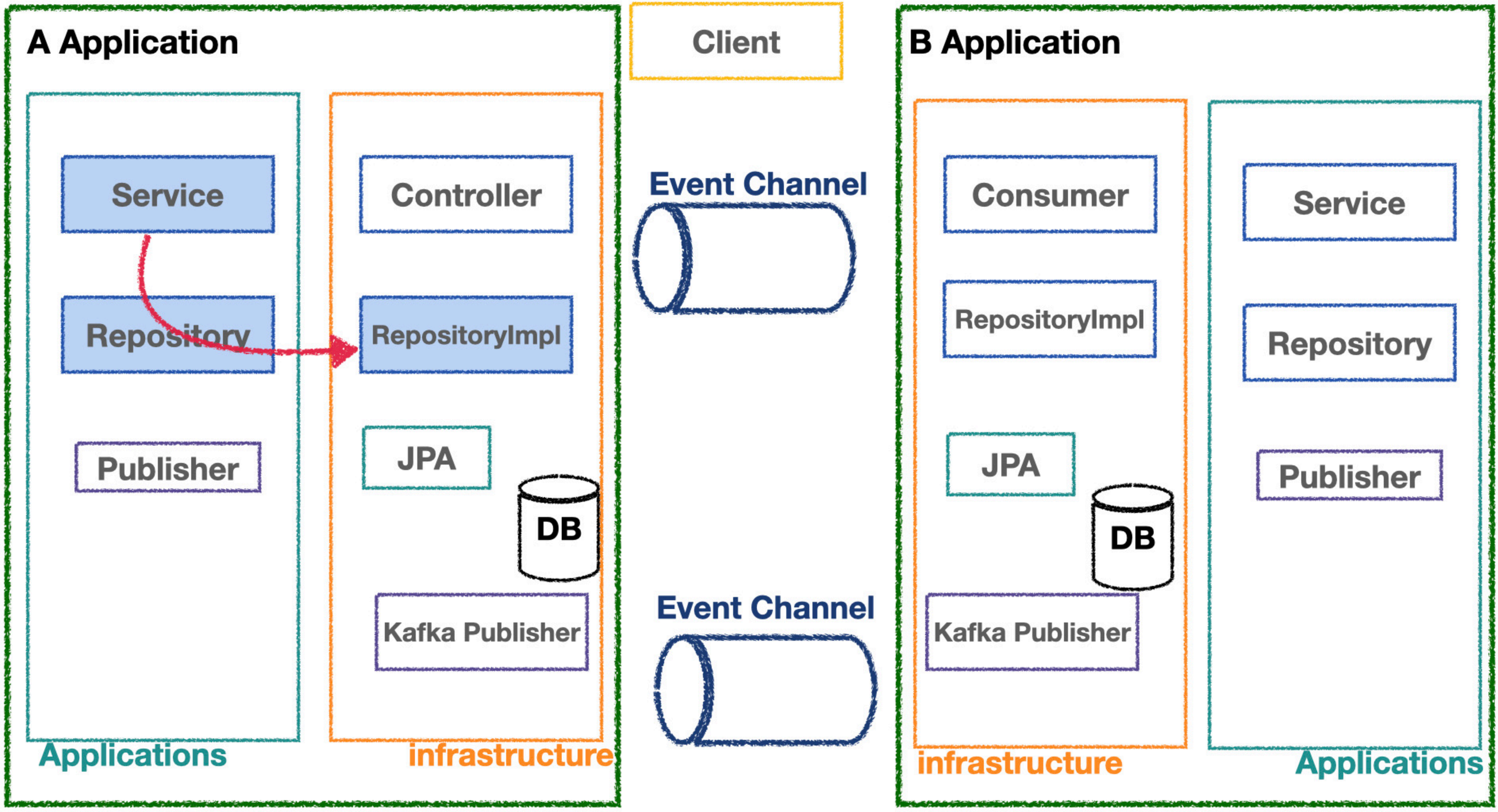
전체 구성

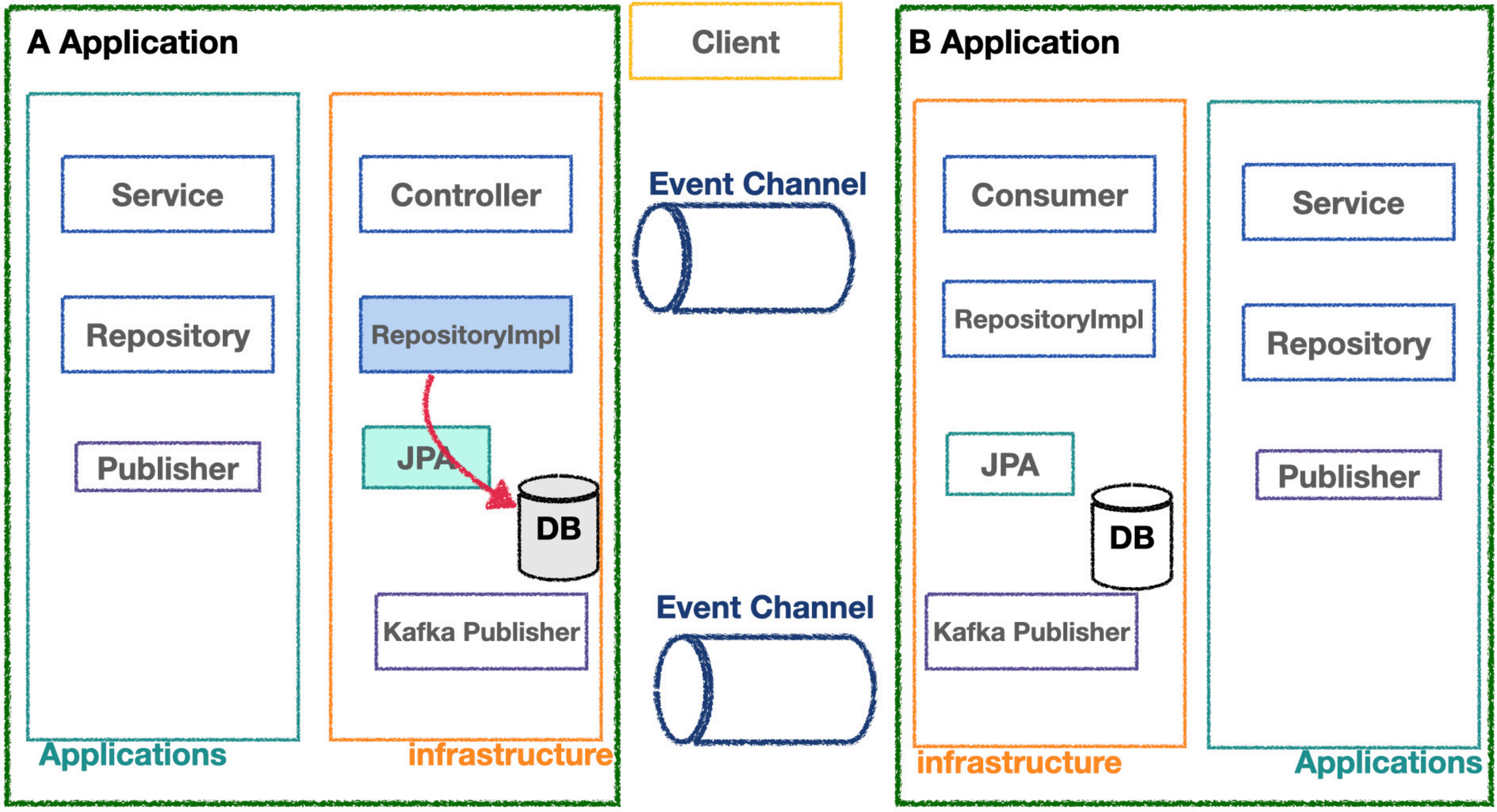

A Application
Event Channel
B Application
Event
Consumer
Event
Controller
Applications
Domain
JPA
DB
DB

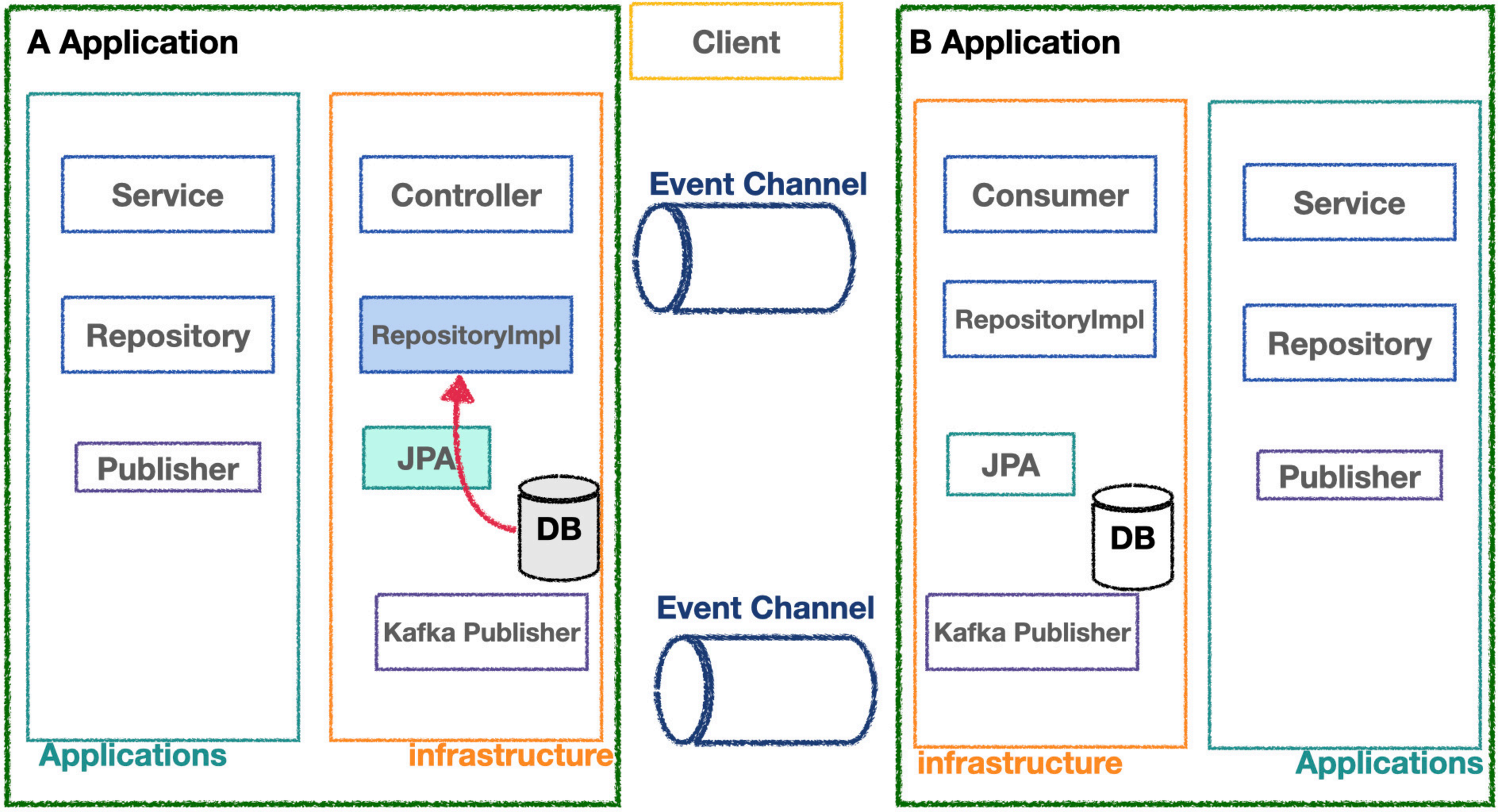

A Application
Service
Repository
Publisher
Applications
Controller
RepositoryImpl
JPA
DB
Kafka Publisher
infrastructure
Client
Event Channel
Event Channel
B Application
Consumer
RepositoryImpl
JPA
DB
Kafka Publisher
infrastructure
Service
Repository
Publisher
Applications

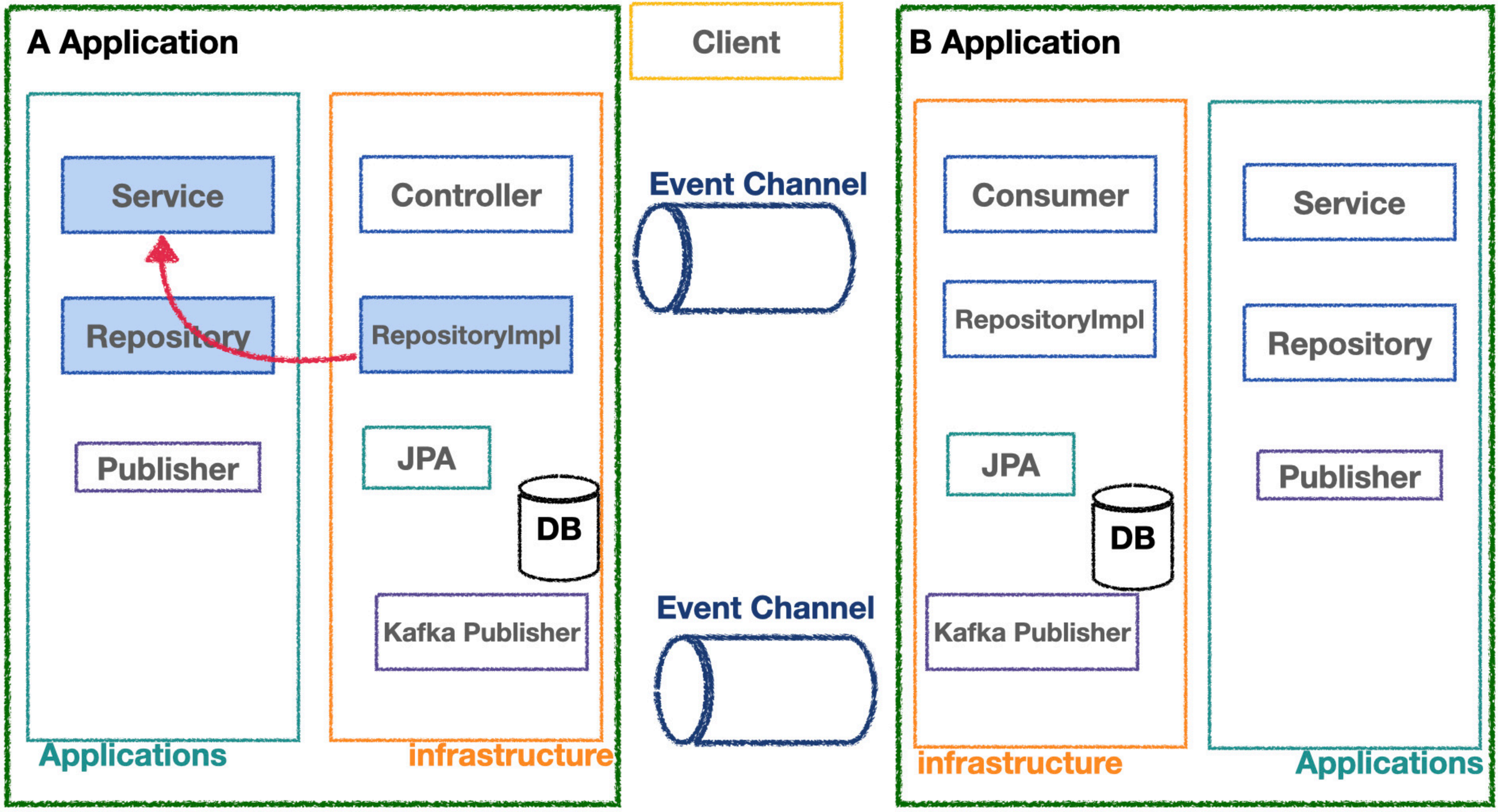

A Application
Client
B Application
Service
Controller
Event Channel
Consumer
Service
Repository
RepositoryImpl
RepositoryImpl
Repository
Publisher
JPA
JPA
Publisher
DB
DB
Kafka Publisher
Event Channel
Kafka Publisher
Applications
infrastructure
infrastructure
Applications

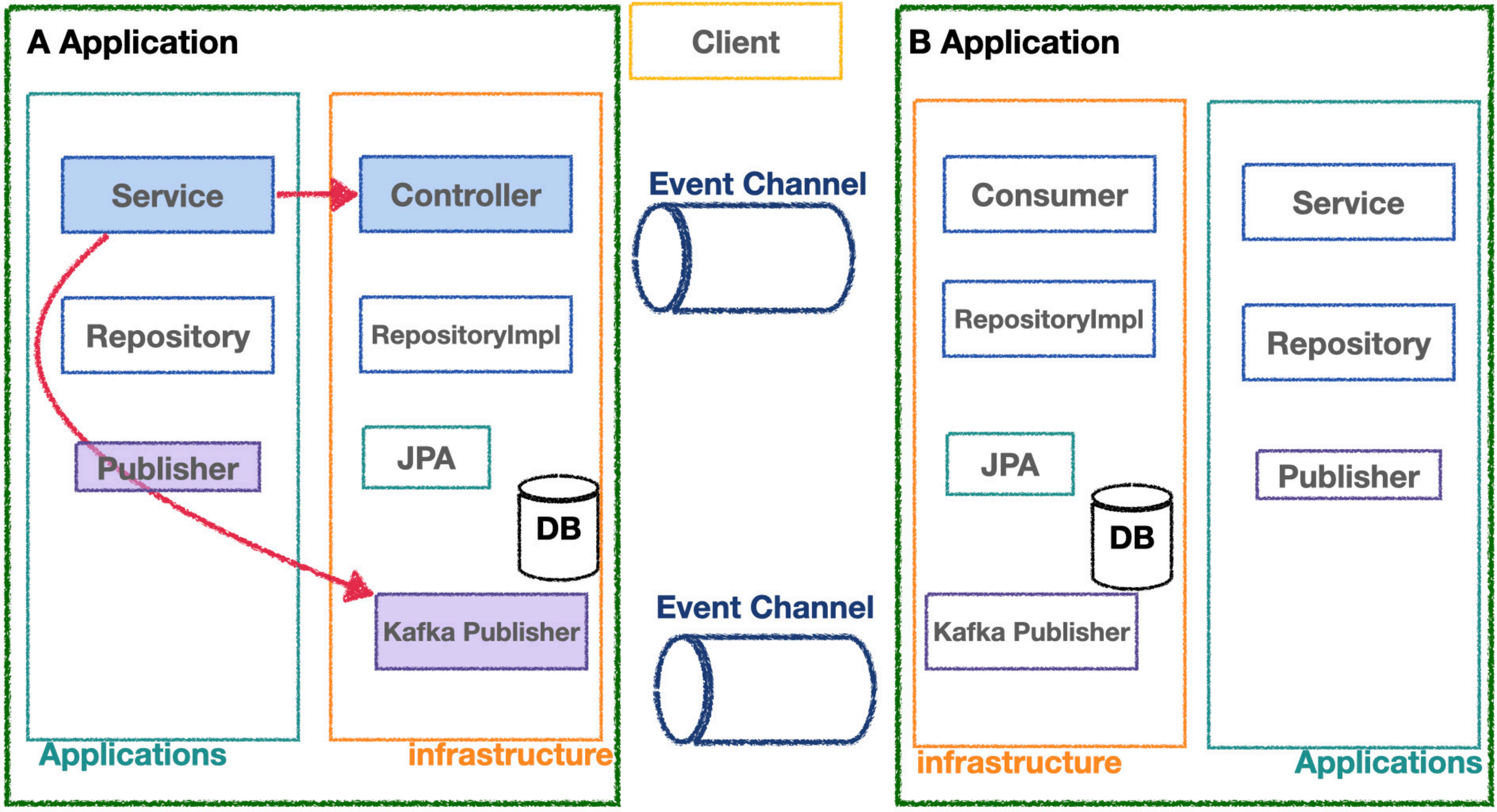

A Application
Client
B Application
Service
Controller
Event Channel
Consumer
Service
Repository
RepositoryImpl
RepositoryImpl
Repository
Publisher
JPA
JPA
Publisher
DB
DB
Kafka Publisher
Event Channel
Kafka Publisher
Applications
infrastructure
infrastructure
Applications

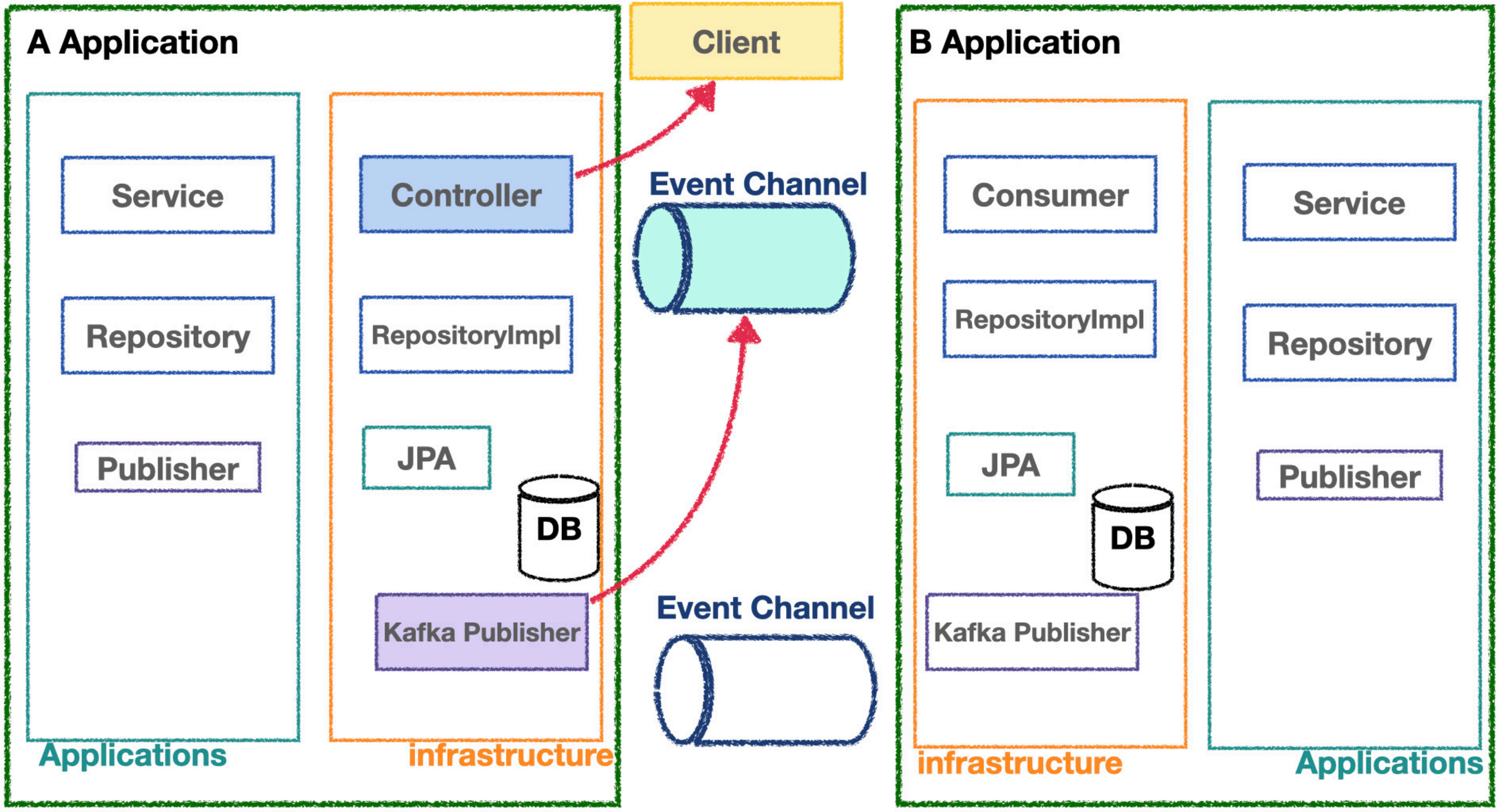

A Application
Service
Repository
Publisher
Controller
RepositoryImpl
JPA
DB
Kafka Publisher
Applications
infrastructure
Client
Event Channel
Event Channel
B Application
Consumer
RepositoryImpl
JPA
DB
Kafka Publisher
Service
Repository
Publisher
infrastructure
Applications

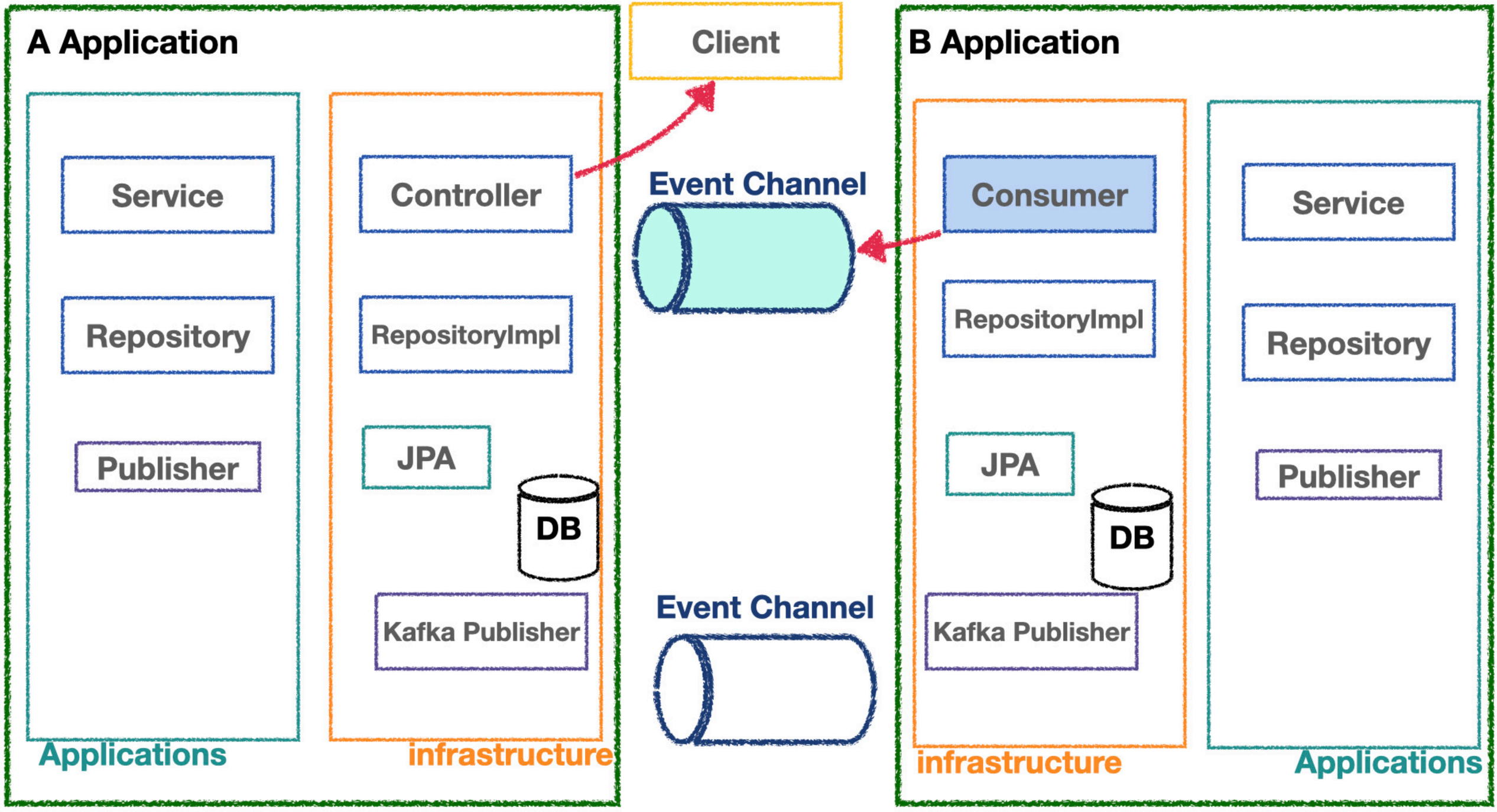

A Application
Service
Repository
Publisher
Applications
Controller
RepositoryImpl
JPA
DB
Kafka Publisher
infrastructure
Client
Event Channel
Event Channel
B Application
Consumer
RepositoryImpl
JPA
DB
Kafka Publisher
infrastructure
Service
Repository
Publisher
Applications

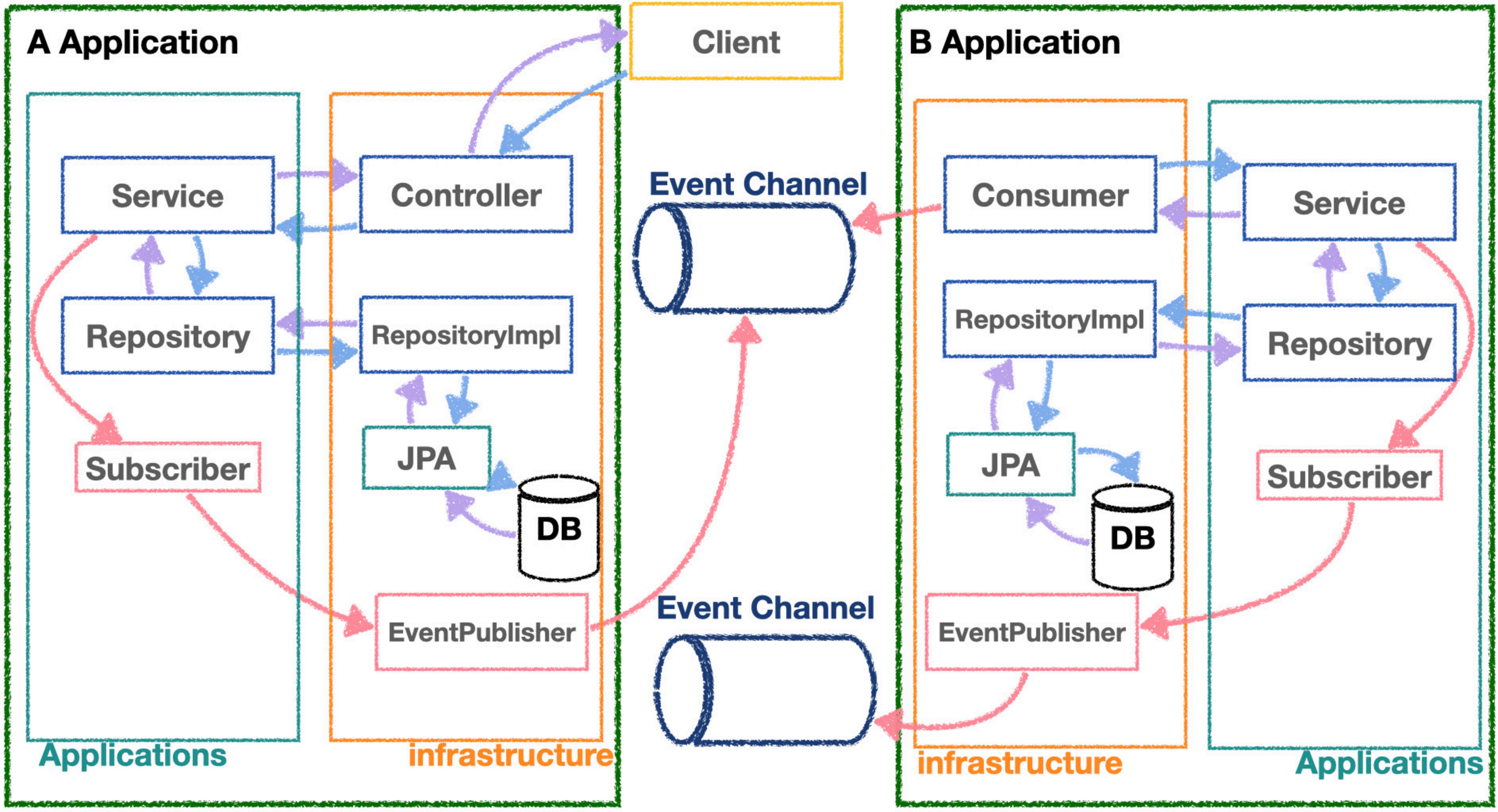

A Application
Service
Repository
Publisher
Applications
Controller
RepositoryImpl
JPA
DB
Kafka Publisher
infrastructure
Client
Event Channel
Event Channel
B Application
Consumer
RepositoryImpl
JPA
DB
Kafka Publisher
infrastructure
Service
Repository
Publisher
Applications


A Application
Client
B Application
Service
Controller
Event Channel
Consumer
Service
Repository
RepositoryImpl
RepositoryImpl
Repository
Publisher
JPA
JPA
Publisher
DB
DB
Kafka Publisher
Event Channel
Kafka Publisher
Applications
infrastructure
infrastructure
Applications


A Application
Client
B Application
Service
Controller
Repository
RepositoryImpl
Publisher
JPA
DB
Kafka Publisher
Event Channel
Event Channel
Consumer
RepositoryImpl
JPA
DB
Kafka Publisher
Service
Repository
Publisher
Applications
infrastructure
infrastructure
Applications


A Application
Client
B Application
Service
Repository
Publisher
Controller
RepositoryImpl
JPA
DB
Kafka Publisher
Event Channel
Event Channel
Consumer
RepositoryImpl
JPA
DB
Kafka Publisher
Service
Repository
Publisher
Applications
infrastructure
infrastructure
Applications


A Application
Client
B Application
Service
Repository
Publisher
Controller
RepositoryImpl
JPA
DB
Kafka Publisher
Event Channel
Event Channel
Consumer
RepositoryImpl
JPA
DB
Kafka Publisher
Service
Repository
Publisher
Applications
infrastructure
infrastructure
Applications


A Application
Client
B Application
Service
Controller
Event Channel
Consumer
Service
Repository
RepositoryImpl
RepositoryImpl
Repository
JPA
JPA
Subscriber
Subscriber
DB
DB
EventPublisher
Event Channel
EventPublisher
Applications
infrastructure
infrastructure
Applications

무화과 나무 키우기

무화과 나무 키우기

만들긴 했는데 언제 전환하지?

요구사항

- 회원의 새로운 담당자 타입 추가


명수는 12살
공격
VS
수비
MBCHD
희망나눔 성금
060-700-1212
이 때 다 !

무화과 나무 키우기

요구사항

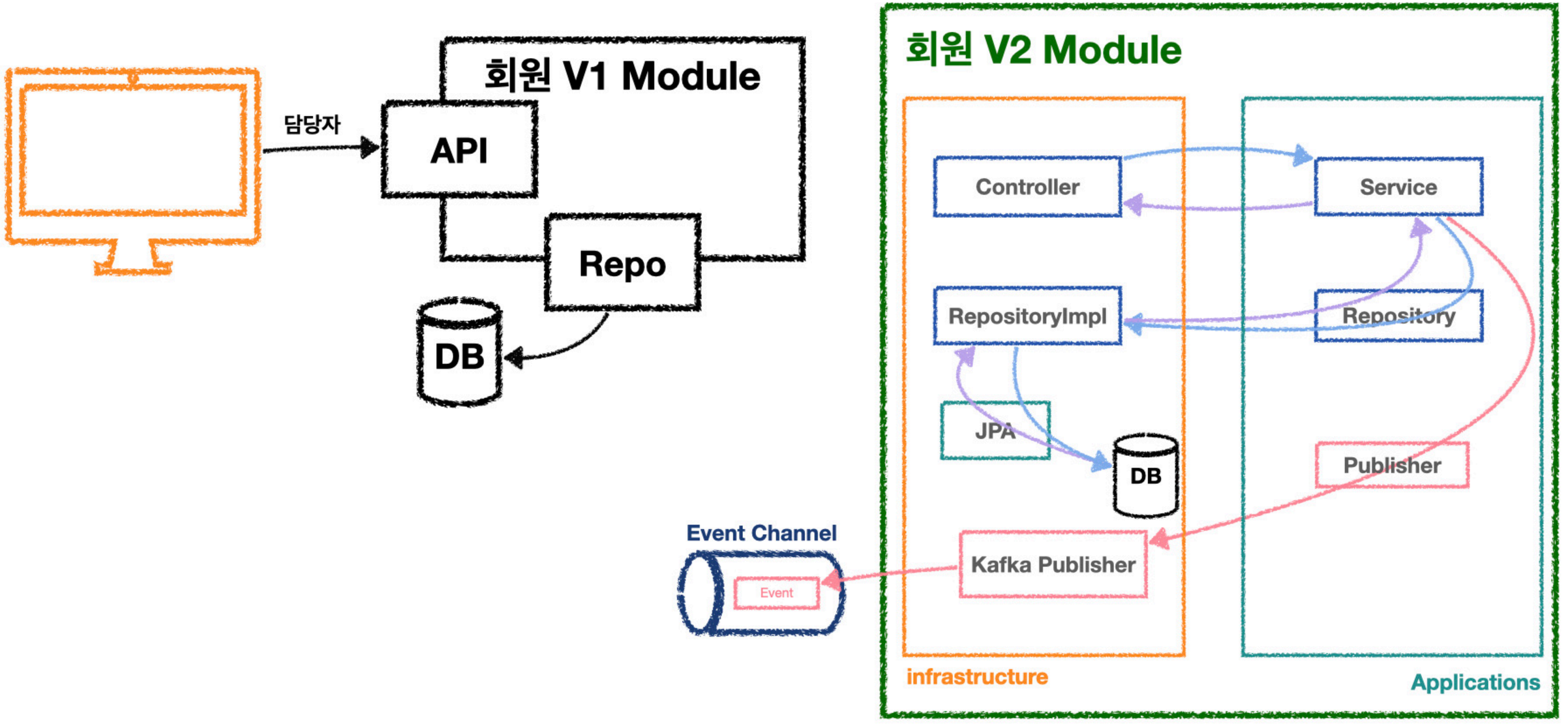
중요한 건 새로운 요구사항의 발생

- 회원의 새로운 담당자 타입 추가

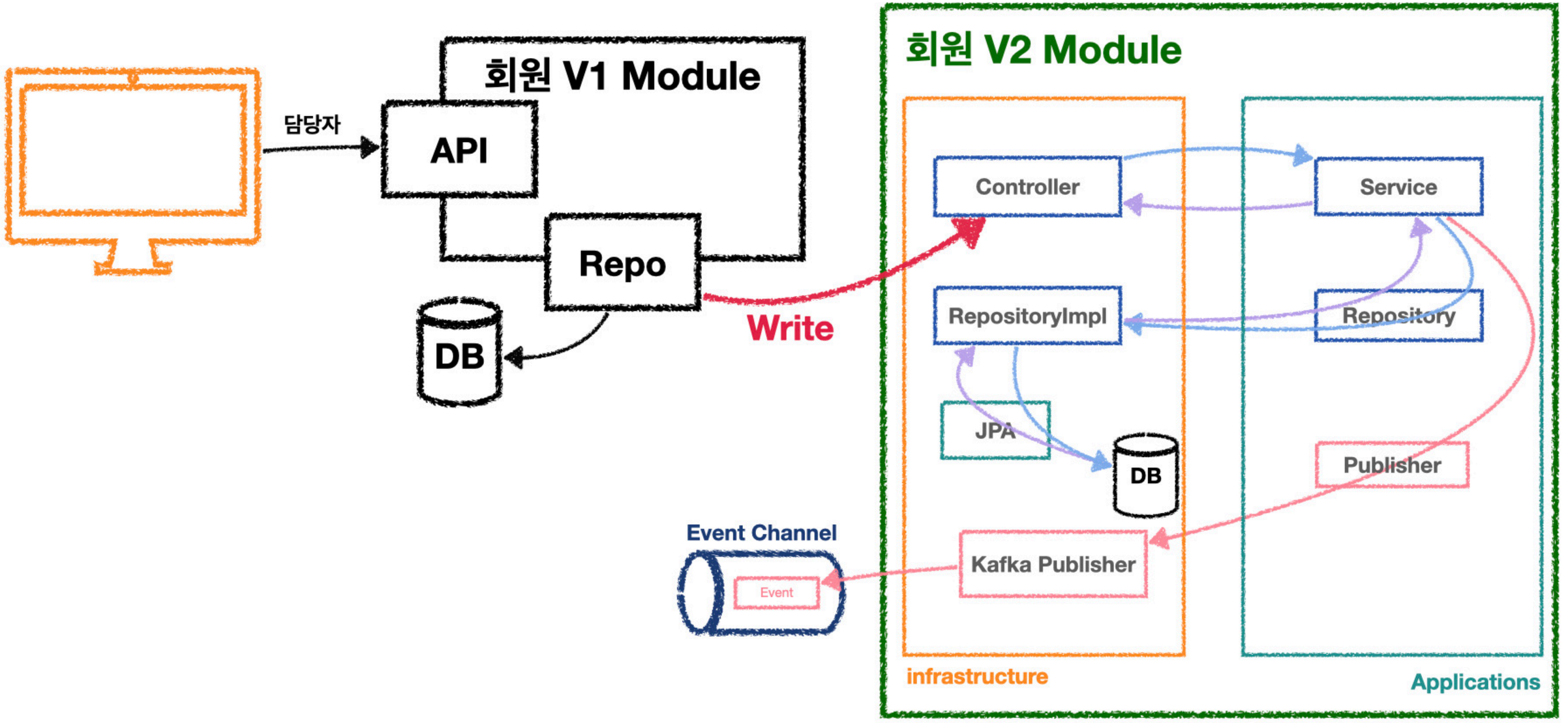
요구사항

서비스 요구사항에 대한 충족
동시에 시스템 전환 기회

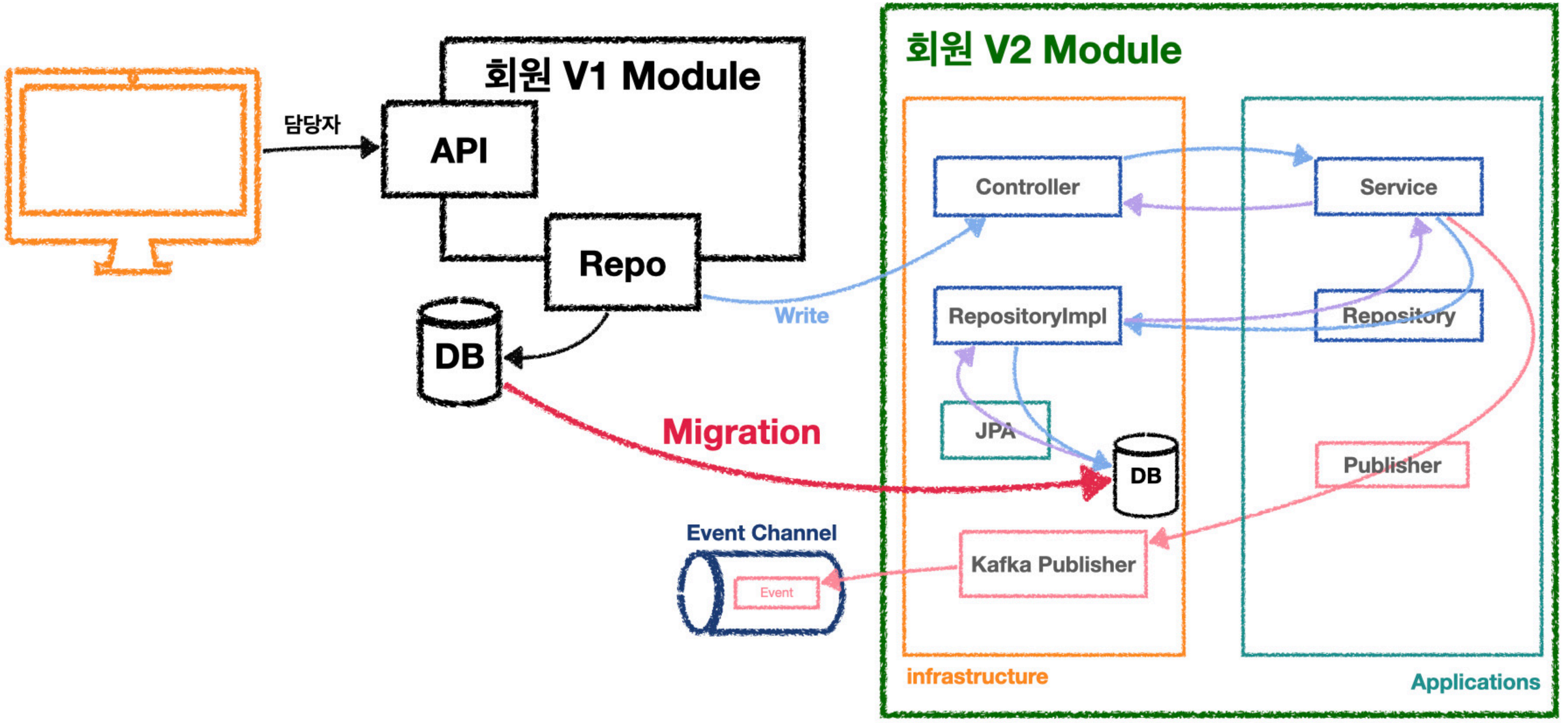
무화과 나무 키우기


회원 V1 Module
담당자
API
Repo
DB
회원 V2 Module
Controller
Service
RepositoryImpl
Repository
JPA
DB
Publisher
Event Channel
Event
Kafka Publisher
infrastructure
Applications

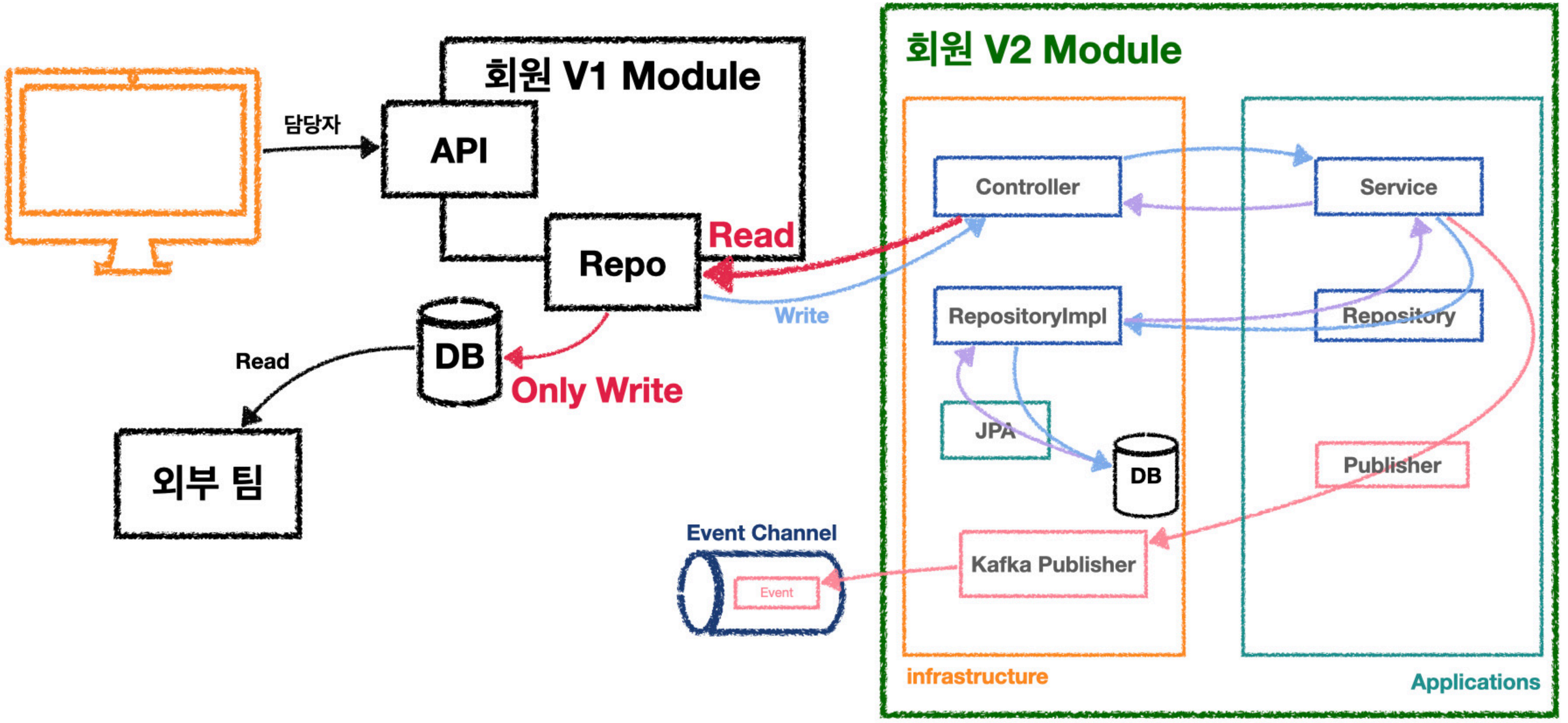
무화과 나무 키우기


회원 V1 Module
담당자
API
Repo
DB
Write
회원 V2 Module
Controller
Service
RepositoryImpl
Repository
JPA
DB
Publisher
Event Channel
Event
Kafka Publisher
infrastructure
Applications

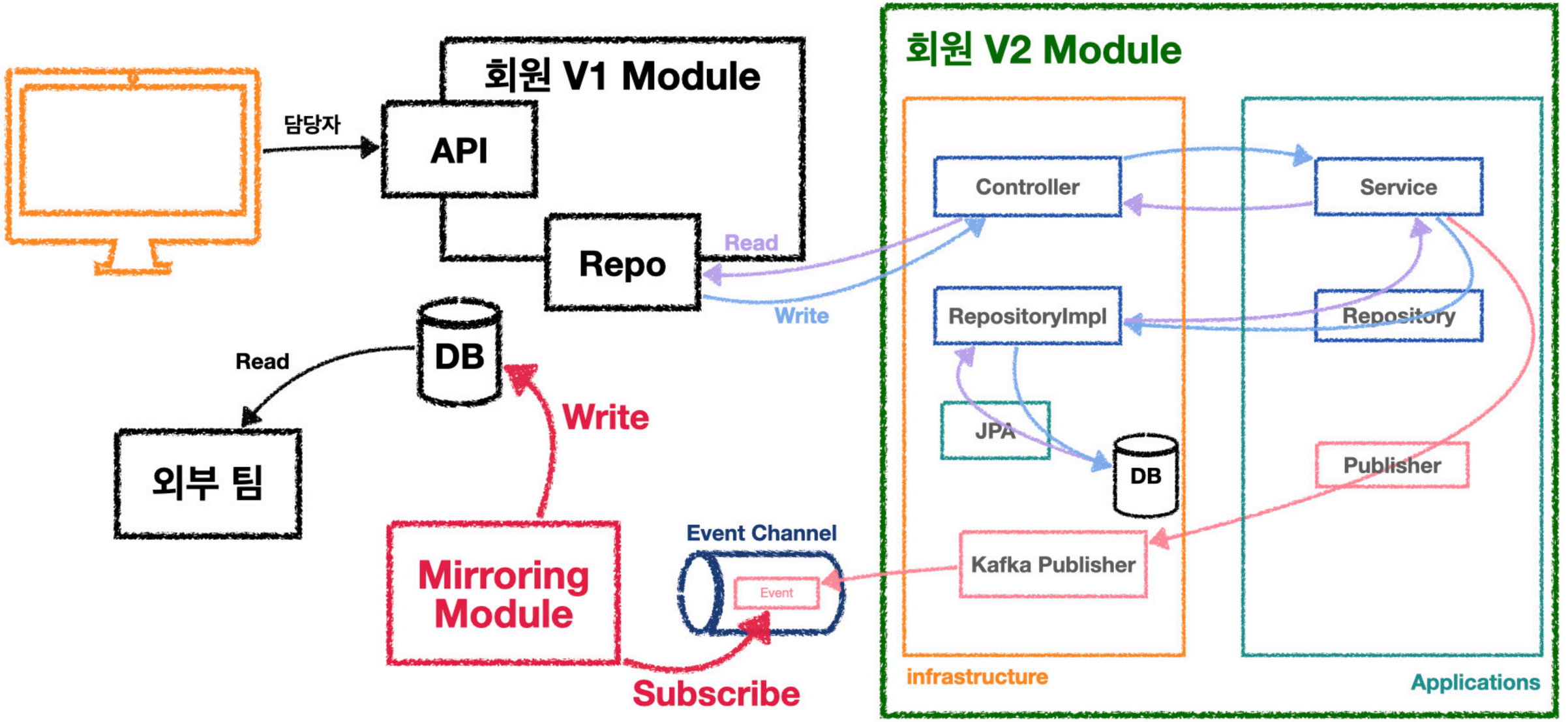
무화과 나무 키우기


회원 V1 Module
담당자
API
Repo
DB
Write
Migration
Event Channel
Event
회원 V2 Module
Controller
Service
RepositoryImpl
Repository
JPA
DB
Publisher
Kafka Publisher
infrastructure
Applications

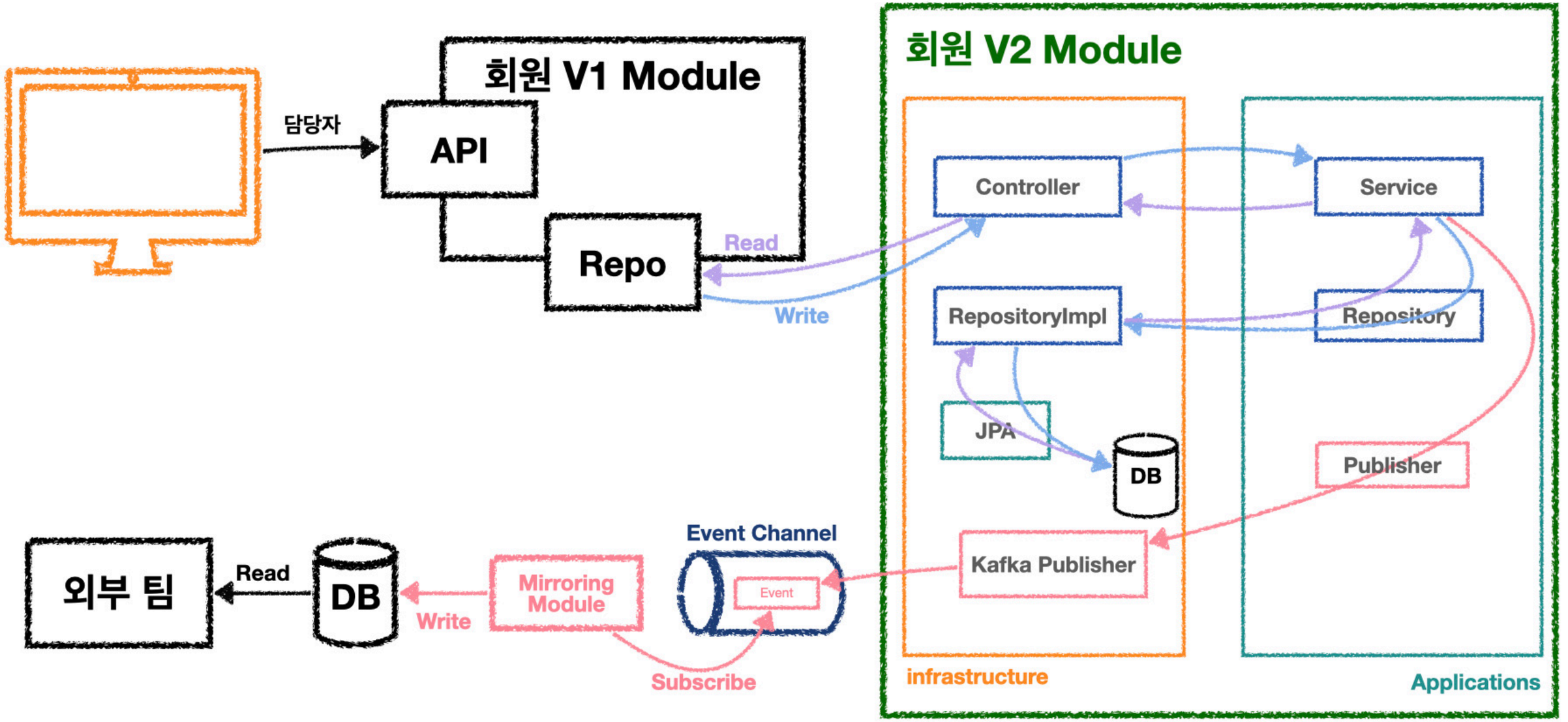
무화과 나무 키우기


회원 V1 Module
API
Repo
Read
Write
DB
Only Write
담당자
Read
외부 팀
Event Channel
Event
회원 V2 Module
Controller
Service
RepositoryImpl
Repository
JPA
DB
Publisher
Kafka Publisher
infrastructure
Applications

무화과 나무 키우기


회원 V1 Module
API
Repo
담당자
Read
Write
DB
Read
외부 팀
Write
Mirroring Module
Event Channel
Event
Subscribe
회원 V2 Module
Controller
Service
RepositoryImpl
Repository
JPA
DB
Publisher
Kafka Publisher
infrastructure
Applications

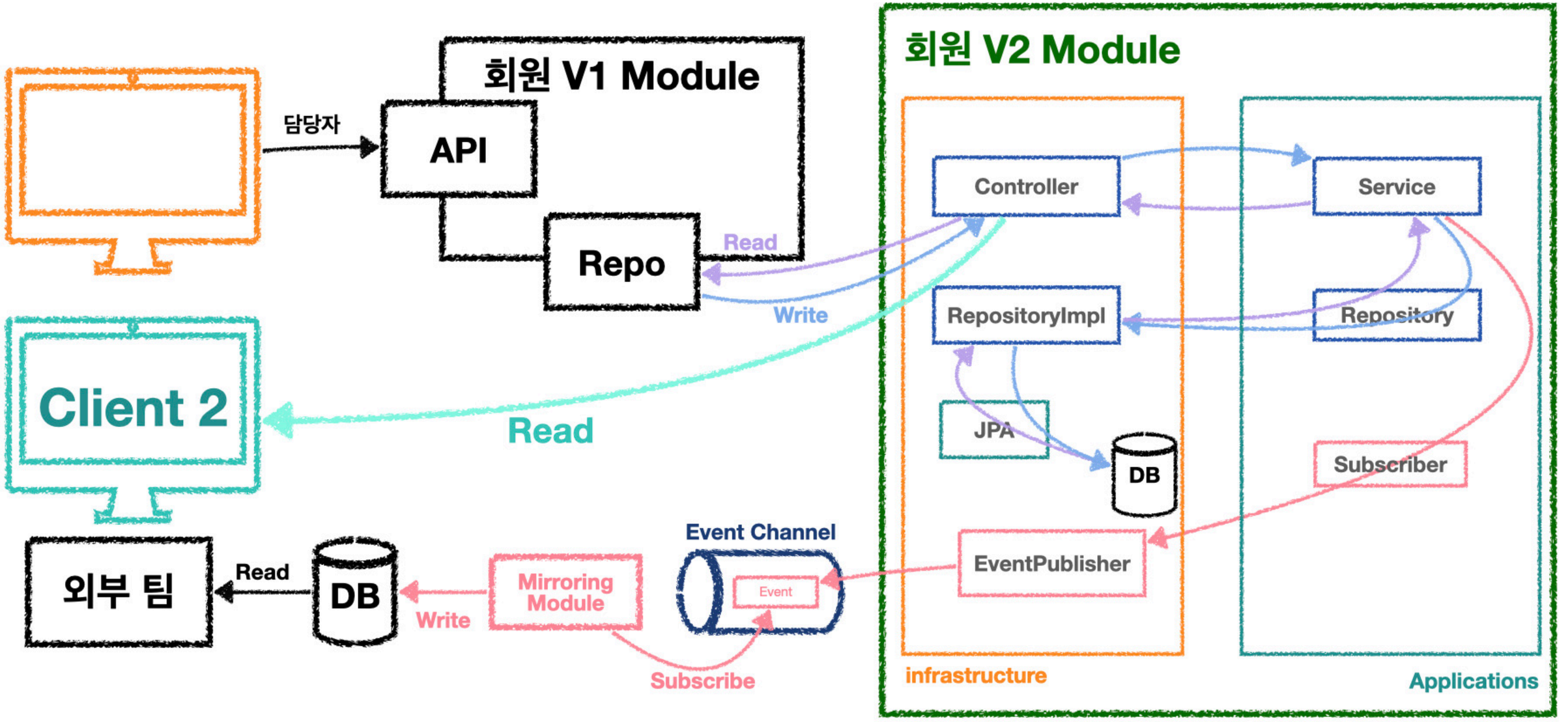
무화과 나무 키우기


회원 V1 Module
API
Repo
담당자
Read
Write
회원 V2 Module
Controller
Service
RepositoryImpl
Repository
JPA
DB
Publisher
Kafka Publisher
infrastructure
Applications
Event Channel
Event
Mirroring Module
Write
Subscribe
DB
Read
외부 팀

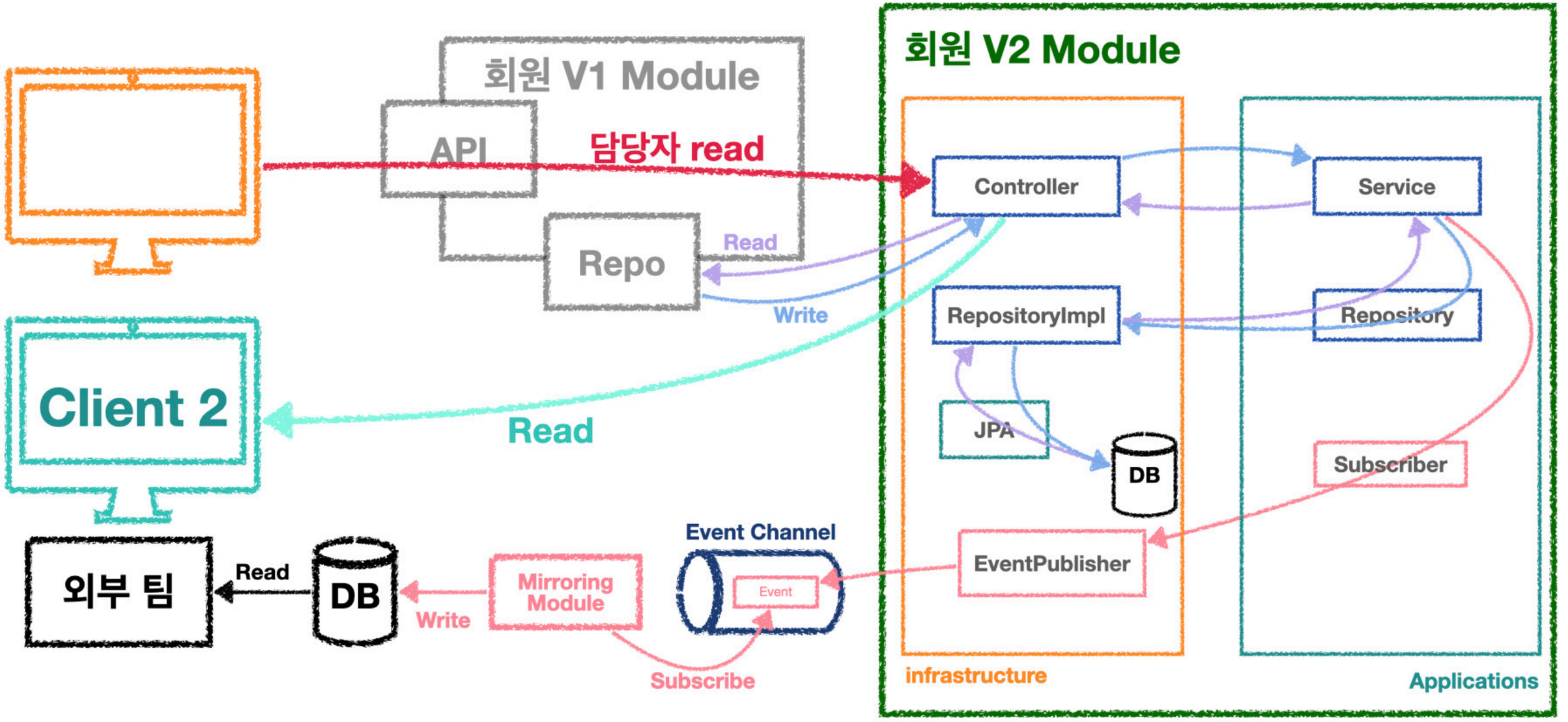
새로운 요구사항

- 회원의 담당자를 조회하고 싶은 새로운 클라이언트 추가

무화과 나무 키우기


회원 V1 Module
API
Repo
담당자
Read
Write
Client 2
Read
외부 팀
Read
DB
Mirroring Module
Write
Event Channel
Event
Subscribe
회원 V2 Module
Controller
Service
RepositoryImpl
Repository
JPA
DB
Subscriber
EventPublisher
infrastructure
Applications

무화과 나무 키우기


회원 V1 Module
API
담당자 read
Repo
Read
Write
회원 V2 Module
Controller
Service
RepositoryImpl
Repository
JPA
DB
Subscriber
EventPublisher
Client 2
Read
외부 팀
Read
DB
Mirroring Module
Write
Event Channel
Event
Subscribe
infrastructure
Applications

무화과 나무 키우기

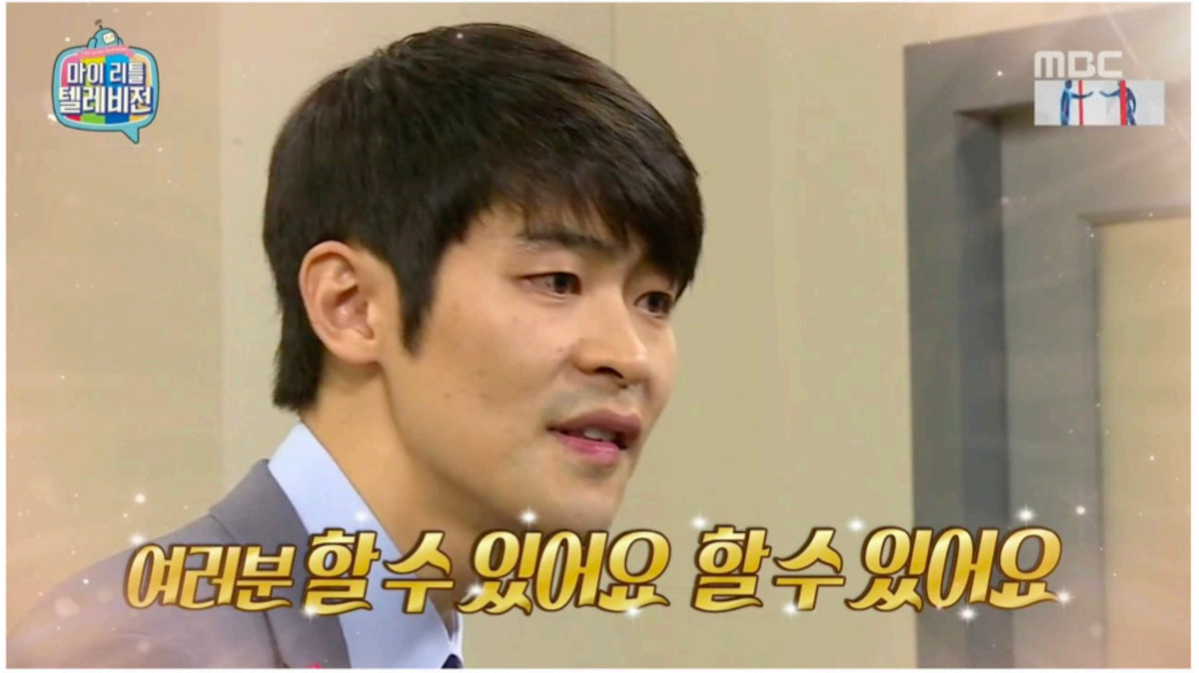
점점 야위어 가는 레거시

점점 풍성해지는 신규 시스템

신규 시스템

두려움 없이 수정 할 수 있어요

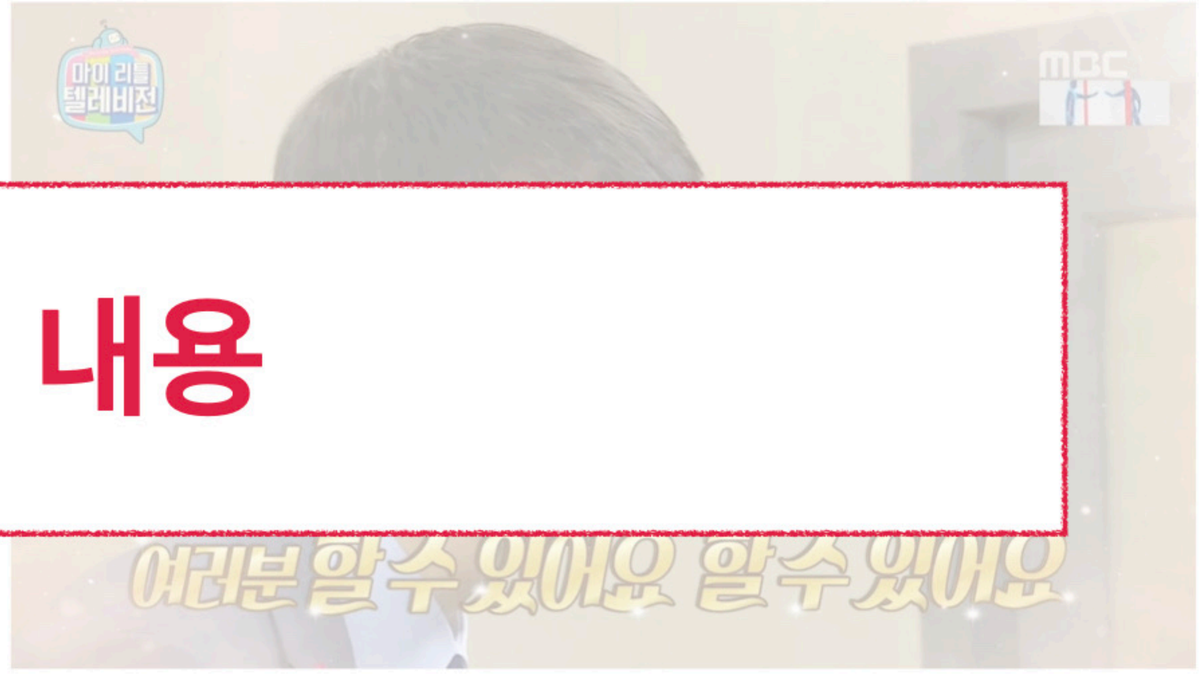
- 역할 별로 레이어가 나누어져 수정에 대한 영향 범위 예측 가능
- 시스템 전반적으로 일관된 구성으로 이루어져 기능 추가나 수정에 대한 비용 감소
- 객체들에게 행위를 위임하는 구조로 되어 코드 가독성 증가
- 로직간 낮은 결합도로 쉬운 변경



신규 시스템

두려움 없이 수정 할 수 있어요

- 역
  영
- 시
  어
- 객
  코드 가독성 증가
- 로직간 낮은 결합도로 쉬운 변경



뻔한 내용

어려웠던 부분들


오늘따라 왜이리 원망스러워 보이는지

책임, 역할 할당에 대한 어려움
(도메인 모델의 모델링 어려움)

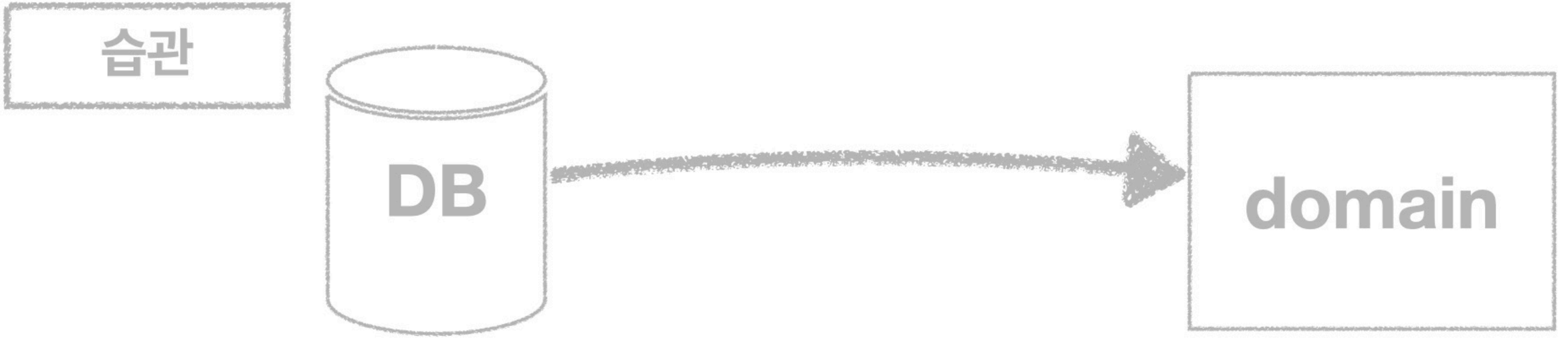
모델링의 어려움

모델링의 어려움

prod_nm	prod_id	prod_url	category_id
Skecher	123456789	http://test.com	A123456
긁힘방지의자캡(사각/투명)소 0572 (4개입)	12345788	http://test.com	A234567
코데즈컴바인 이너웨어 [코데즈이너웨어]	123456787	http://test.com	A345678


```
{
    "prod":{
        "prodId":"123456789",
        "catId":"A123456",
        "prodNm":"Skecher",
        "prodUrl":"http://test.com"
    }
}
```

모델링의 어려움

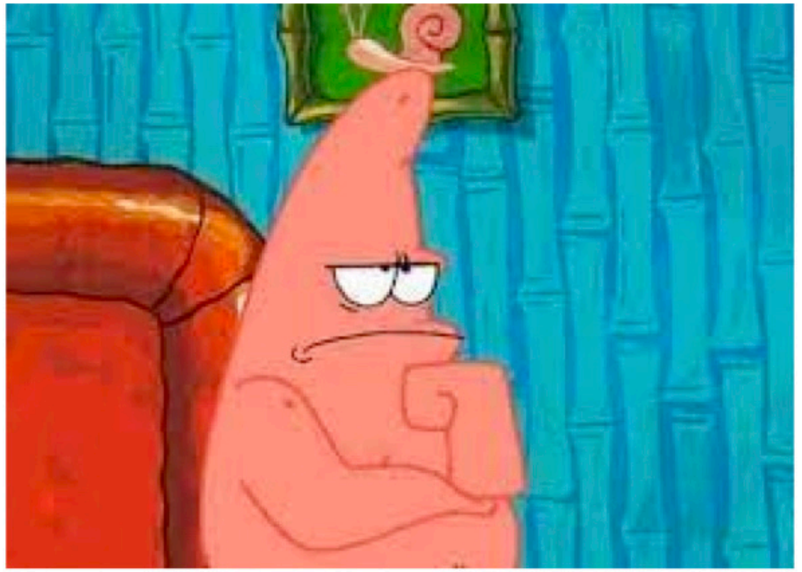


DB 구조를 도메인 구조에 반영하는 습관

prod_nm			
Skecher	123456789	http://test.com	A123456
긁힘방지의자캡(사각/투명)소 0572 (4개입)	12345788	http://test.com	A234567
코데즈컴바인 이너웨어 [코데즈이너웨어]	123456787	http://test.com	A345678

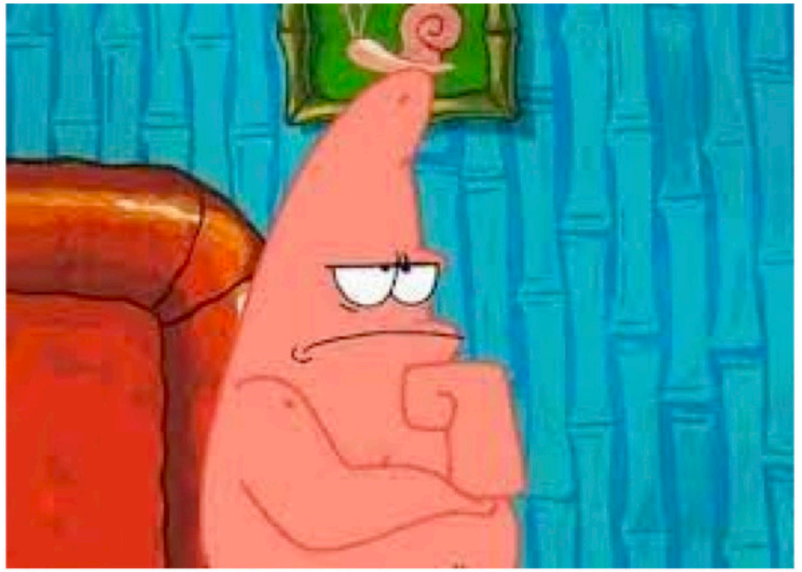

```
"prod":{
    "prodId":"123456789",
    "catId":"A123456",
    "prodNm":"Skecher",
    "prodUrl":"http://test.com"
}
```

역할에 대한 고민



‘상품’의 의미 고민

우리 서비스에서 ‘상품’은 어떤 의미일까?

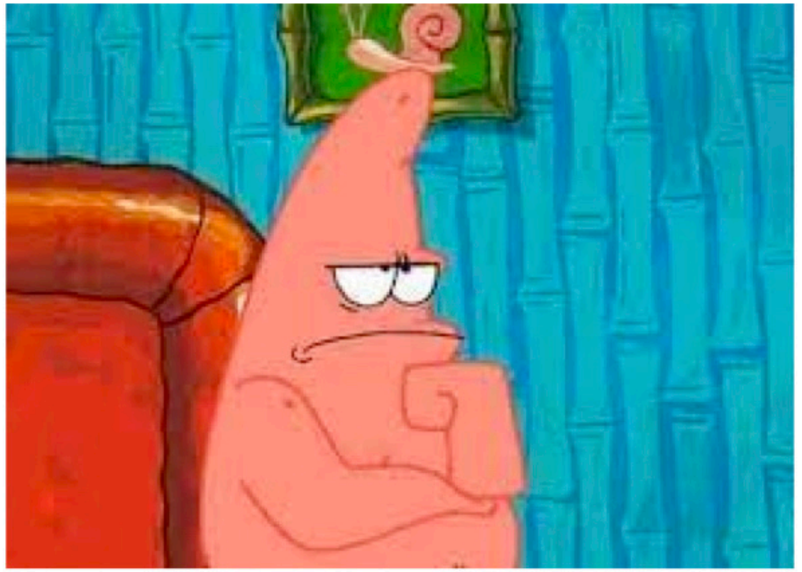


모델링의 어려움

‘상품’의 의미 고민

재화나 서비스 그 자체로는 의미가 와 닿지 않음

우리 서비스에서 ‘상품’은 어떤 의미일까?



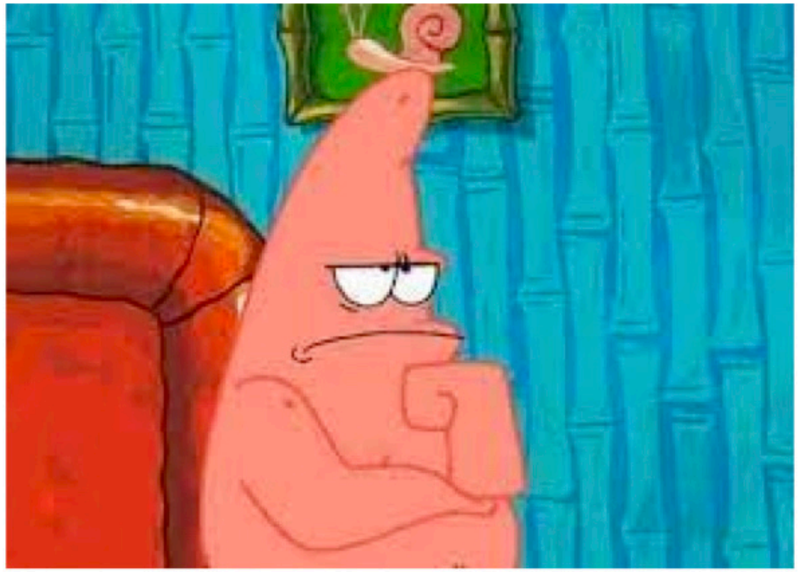
모델링의 어려움

‘상품’의 의미 고민

재화나 서비스의 의미는 와 닿지 않음

우리 서비스에서 ‘상품’은 어떤 의미일까?

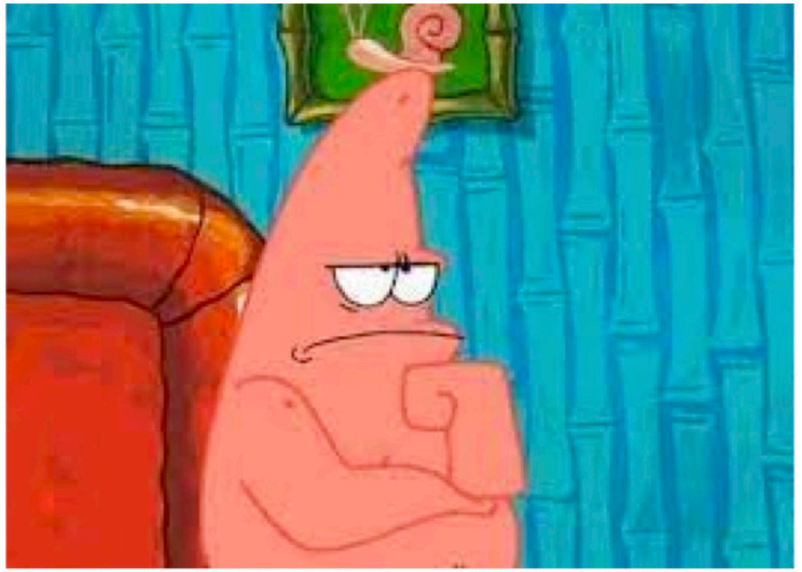
판매자가 구매자에게 제공하는
재화나 서비스에 대한 구매 제안서



모델링의 어려움

‘판매상품’ 정의

‘상품’을 ‘판매 상품’으로 재정의

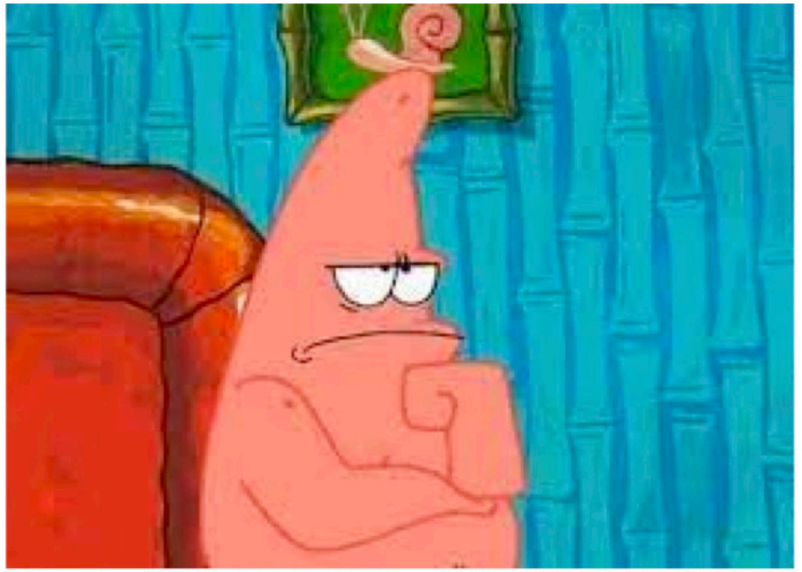


모델링의 어려움

‘판매상품’ 정의

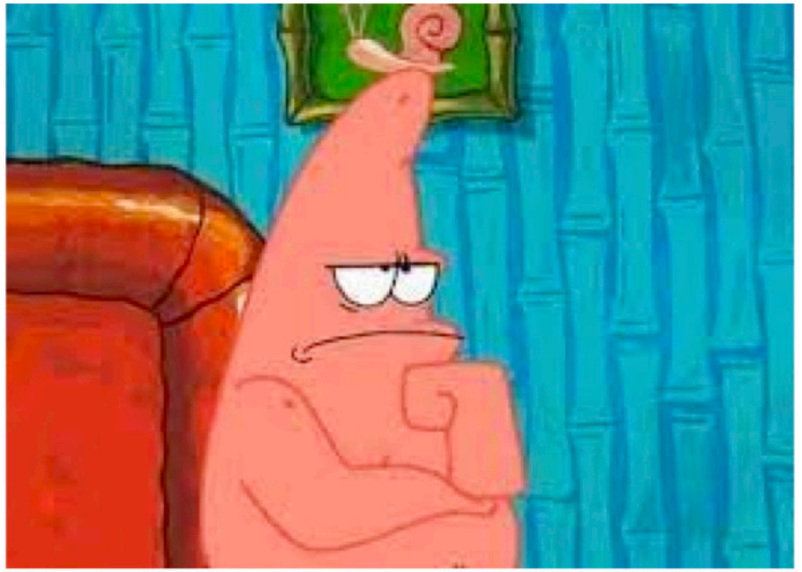
판매자가 상품을 판매하기 위한 판매 정보

‘상품’을 ‘판매 상품’으로 재정의



‘판매상품’ 역할 고민

‘판매 상품’이 수행하는 역할은?



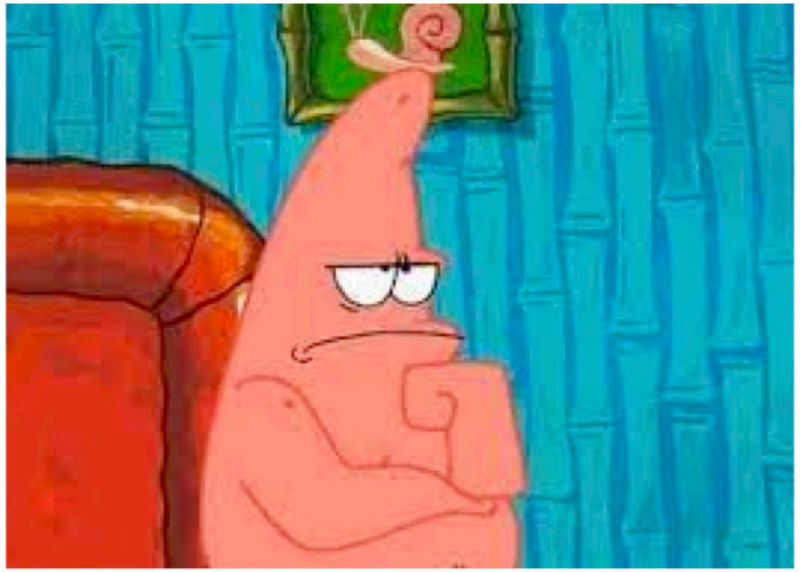
모델링의 어려움

역할에 대한 고민

가격 제시, 상품에 대한 정보, 상품에 대한 이벤트

.....

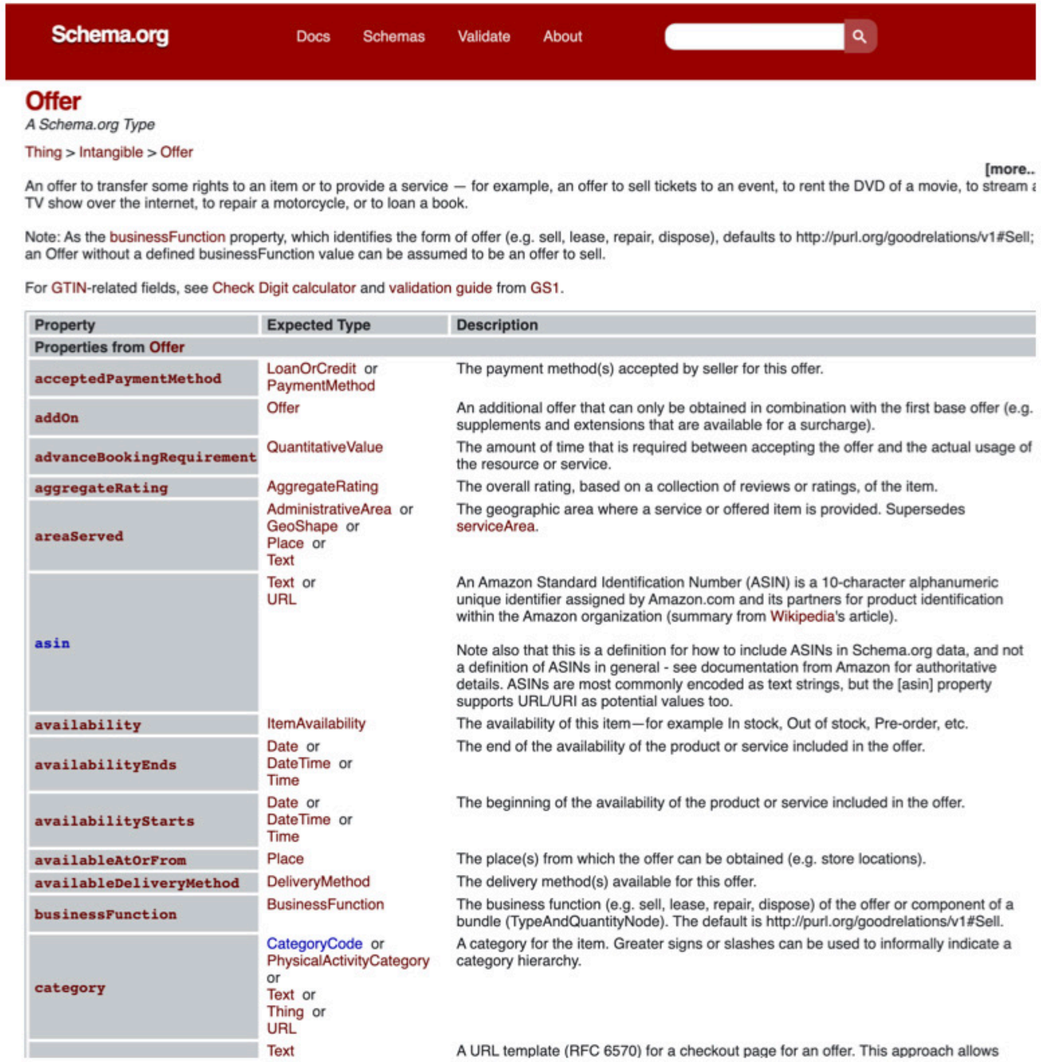
'판매 상품'이 수행하는 역할은?



모델링의 어려움

역할에 맞는 타입 구성

- 구현체에 종속되는 스키마가 아닌 일반적으로 사용 가능한 추상화 된 스키마 구성 필요
  - schema.org
- ‘판매 상품’을 표현할 수 있는 타입 검색
  - **‘Offer’** 타입 선정
  - ‘아이템에 대한 권리를 양도하거나 서비스를 제공 하겠다는 제안.’
- 필요한 속성들 차용해서 구성



Schema.org Docs Schemas Validate About

Offer

A Schema.org Type

Thing > Intangible > Offer

[more..

An offer to transfer some rights to an item or to provide a service — for example, an offer to sell tickets to an event, to rent the DVD of a movie, to stream a TV show over the internet, to repair a motorcycle, or to loan a book.

Note: As the businessFunction property, which identifies the form of offer (e.g. sell, lease, repair, dispose), defaults to http://purl.org/goodrelations/v1#Sell; an Offer without a defined businessFunction value can be assumed to be an offer to sell.

For GTIN-related fields, see Check Digit calculator and validation guide from GS1.

Property	Expected Type	Description
Properties from Offer		
acceptedPaymentMethod	LoanOrCredit or PaymentMethod	The payment method(s) accepted by seller for this offer.
addOn	Offer	An additional offer that can only be obtained in combination with the first base offer (e.g. supplements and extensions that are available for a surcharge).
advanceBookingRequirement	QuantitativeValue	The amount of time that is required between accepting the offer and the actual usage of the resource or service.
aggregateRating	AggregateRating	The overall rating, based on a collection of reviews or ratings, of the item.
areaServed	AdministrativeArea or GeoShape or Place or Text	The geographic area where a service or offered item is provided. Supersedes serviceArea.
asin	Text or URL	An Amazon Standard Identification Number (ASIN) is a 10-character alphanumeric unique identifier assigned by Amazon.com and its partners for product identification within the Amazon organization (summary from Wikipedia's article). Note also that this is a definition for how to include ASINs in Schema.org data, and not a definition of ASINs in general - see documentation from Amazon for authoritative details. ASINs are most commonly encoded as text strings, but the [asin] property supports URL/URI as potential values too.
availability	ItemAvailability	The availability of this item—for example In stock, Out of stock, Pre-order, etc.
availabilityEnds	Date or DateTime or Time	The end of the availability of the product or service included in the offer.
availabilityStarts	Date or DateTime or Time	The beginning of the availability of the product or service included in the offer.
availableAtOrFrom	Place	The place(s) from which the offer can be obtained (e.g. store locations).
availableDeliveryMethod	DeliveryMethod	The delivery method(s) available for this offer.
businessFunction	BusinessFunction	The business function (e.g. sell, lease, repair, dispose) of the offer or component of a bundle (TypeAndQuantityNode). The default is http://purl.org/goodrelations/v1#Sell.
category	CategoryCode or PhysicalActivityCategory or Text or Thing or URL	A category for the item. Greater signs or slashes can be used to informally indicate a category hierarchy.
	Text	A URL template (RFC 6570) for a checkout page for an offer. This approach allows

모델링의 어려움

여기서 잠깐!! schema.org가 무엇인가요?

Schema.org is a collaborative, community activity with a mission to create, maintain, and promote schemas for structured data on the Internet, on web pages, in email messages, and beyond. Schema.org vocabulary can be used with many different encodings, including RDFa, Microdata and JSON-LD. These vocabularies cover entities, relationships between entities and actions, and can easily be extended through a well-documented extension model. Over 10 million sites use Schema.org to markup their web pages and email messages. Many applications from Google, Microsoft, Pinterest, Yandex and others already use these vocabularies to power rich, extensible experiences.
Founded by Google, Microsoft, Yahoo and Yandex, Schema.org vocabularies are developed by an open community process, using the public-schemaorg@w3.org mailing list and through GitHub. A shared vocabulary makes it easier for webmasters and developers to decide on a schema and get the maximum benefit for their efforts. It is in this spirit that the founders, together with the larger community have come together - to provide a shared collection of schemas.

- 필요한 속성들 차용해서 구성

모델링의 어려움

세 줄 요약

인터넷, 웹 페이지, 이메일 메시지 등에서 구조화된 데이터에 대한 스키마를 만들고, 유지 관리하고, 홍보하는 것을 사명으로 하는 협업 커뮤니티 활동입니다.

어휘를 공유하면 웹마스터와 개발자가 스키마를 쉽게 결정하고 노력에 대한 최대한의 이점을 얻을 수 있습니다

Google, Microsoft, Yahoo, Yandex가 설립한 Schema.org

모델링의 어려움

역할에 맞는 타입 구성

- 구현체에 종속되는 스키마가 아닌 일반적으로 사용 가능한 추상화 된 스키마 구성 필요
  - schema.org
- ‘판매 상품’을 표현할 수 있는 타입 검색
  - ‘아이템에 대한 권리를 양도하거나 서비스를 제공 하겠다는 제안.’
  - **‘Offer’** 타입 선정
- 필요한 속성들 차용해서 구성

Schema.org Docs Schemas Validate About

Offer

A Schema.org Type

Thing > Intangible > Offer

[more..

An offer to transfer some rights to an item or to provide a service — for example, an offer to sell tickets to an event, to rent the DVD of a movie, to stream a TV show over the internet, to repair a motorcycle, or to loan a book.

Note: As the businessFunction property, which identifies the form of offer (e.g. sell, lease, repair, dispose), defaults to http://purl.org/goodrelations/v1#Sell; an Offer without a defined businessFunction value can be assumed to be an offer to sell.

For GTIN-related fields, see Check Digit calculator and validation guide from GS1.

Property	Expected Type	Description
Properties from Offer		
acceptedPaymentMethod	LoanOrCredit or PaymentMethod	The payment method(s) accepted by seller for this offer.
addOn	Offer	An additional offer that can only be obtained in combination with the first base offer (e.g. supplements and extensions that are available for a surcharge).
advanceBookingRequirement	QuantitativeValue	The amount of time that is required between accepting the offer and the actual usage of the resource or service.
aggregateRating	AggregateRating	The overall rating, based on a collection of reviews or ratings, of the item.
areaServed	AdministrativeArea or GeoShape or Place or Text	The geographic area where a service or offered item is provided. Supersedes serviceArea.
asin	Text or URL	An Amazon Standard Identification Number (ASIN) is a 10-character alphanumeric unique identifier assigned by Amazon.com and its partners for product identification within the Amazon organization (summary from Wikipedia's article). Note also that this is a definition for how to include ASINs in Schema.org data, and not a definition of ASINs in general - see documentation from Amazon for authoritative details. ASINs are most commonly encoded as text strings, but the [asin] property supports URL/URI as potential values too.
availability	ItemAvailability	The availability of this item—for example In stock, Out of stock, Pre-order, etc.
availabilityEnds	Date or DateTime or Time	The end of the availability of the product or service included in the offer.
availabilityStarts	Date or DateTime or Time	The beginning of the availability of the product or service included in the offer.
availableAtOrFrom	Place	The place(s) from which the offer can be obtained (e.g. store locations).
availableDeliveryMethod	DeliveryMethod	The delivery method(s) available for this offer.
businessFunction	BusinessFunction	The business function (e.g. sell, lease, repair, dispose) of the offer or component of a bundle (TypeAndQuantityNode). The default is http://purl.org/goodrelations/v1#Sell.
category	CategoryCode or PhysicalActivityCategory or Text or Thing or URL	A category for the item. Greater signs or slashes can be used to informally indicate a category hierarchy.
	Text	A URL template (RFC 6570) for a checkout page for an offer. This approach allows

모델링의 어려움

역할(

- 구현체에 종
  추상화 된 스
  - schema
- '판매 상품'
  - 'Offer'
  - '아이템(
    다는 제안
- 필요한 속성

```
{
    "offer":{
        "identifier":"123456789",
        "itemOffered":{
            "category":{
                "identifier":"A123456"
            }
        },
        "name":"Skecher",
        "url":"http://test.com"
    }
}
```

category
Text or Thing or URL
Text
A URL template (RFC 6570) for a checkout page for an offer. This approach allows


오분순삭
0:58
뇌가 뽑혀 나갈듯한 고통

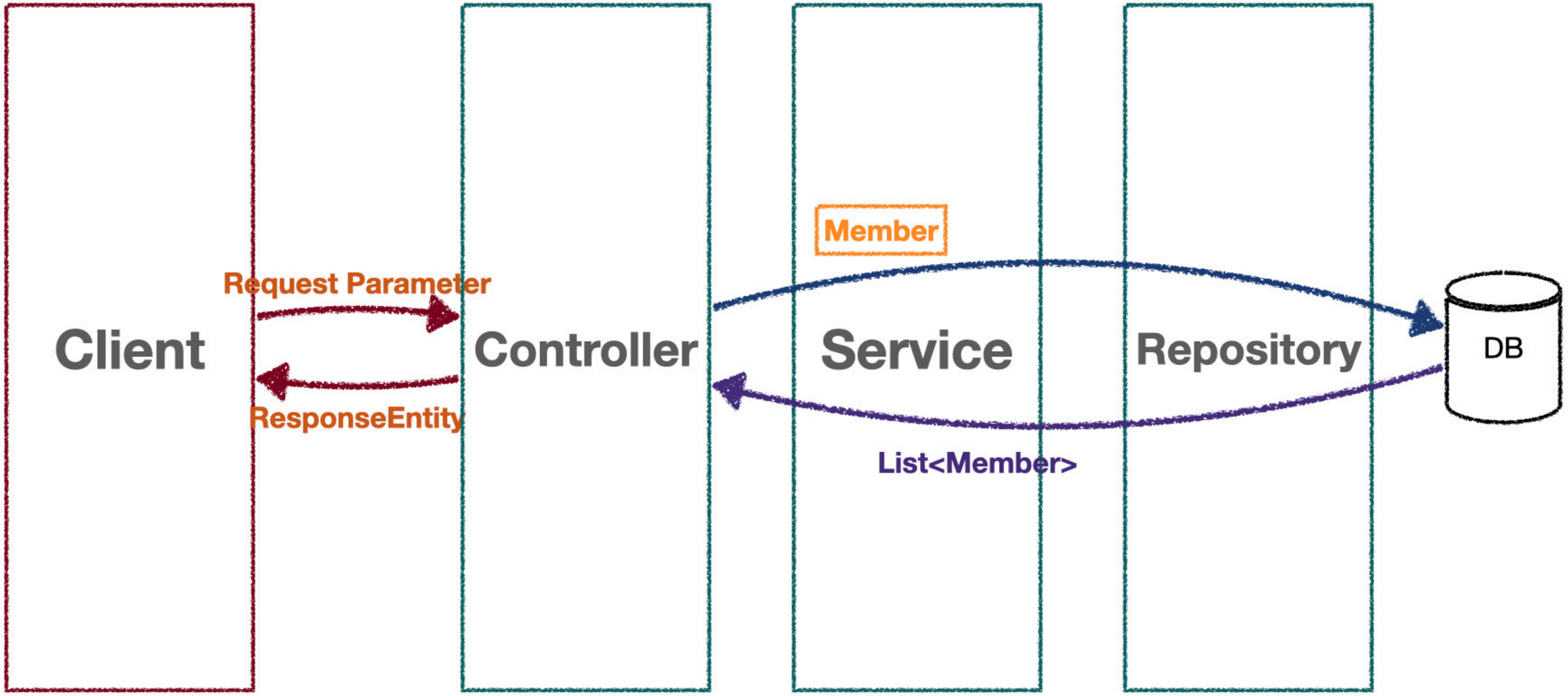
의존성 방향 정리에 대한 어려움

- 인터페이스가 잘 정의되지 않은 경우
- 역할이 잘 할당되지 않은 경우

요구사항

- 회원 리스트를 여러가지 조건으로 조회

의존성 정리 어려움


Client
Request Parameter
ResponseEntity
Controller
Member
Service
Repository
DB
List<Member>

요구사항 추가

- 회원 리스트를 여러가지 조건으로 조회
- 페이징을 통해 개수 조정

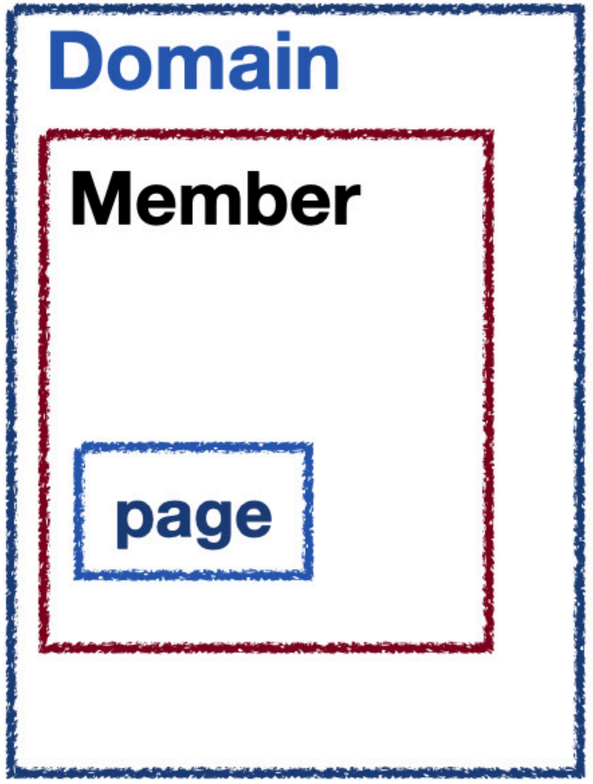
페이징을 어떻게 구현?

- 기존 레이어 간 메세지 전달을 담당하던 Member 객체를 통해 구현

의존성 정리 어려움

Member 타입의 역할 확대

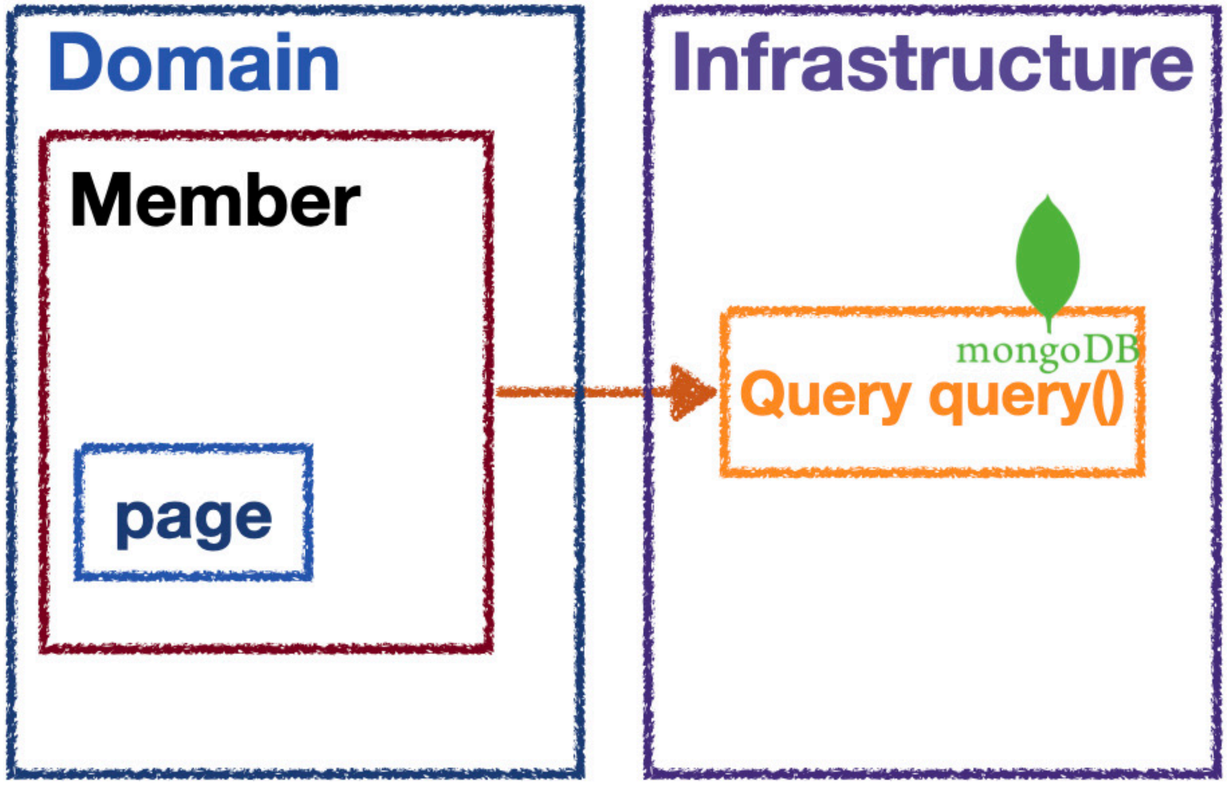
- 페이징에 대한 요구사항 확대 지원
  - Member 내부에 Page 데이터 추가



의존성 정리 어려움

Member 타입의 역할 확대

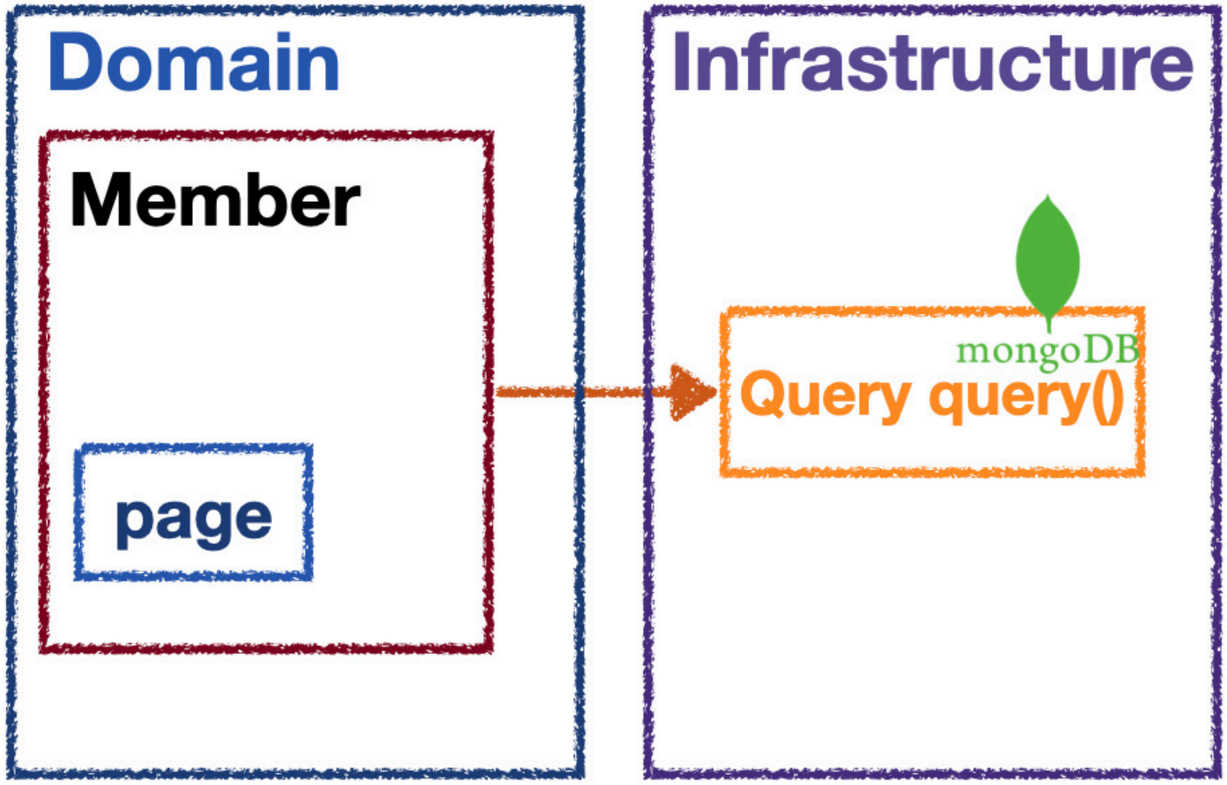
- 페이징에 대한 요구사항 확대 지원
  - Member 내부에 Page 데이터 추가
- DB 쿼리 생성에 대한 책임을 Member에게 부여



의존성 정리 어려움

Member 타입의 역할 확대

- 페이징에 대한 요구사항 확대 지원
  - Member 내부에 Page 데이터 추가
- DB 쿼리 생성에 대한 책임을 Member에게 부여
- 본래의 역할을 넘어서는 책임을 부여하고 싶은 유혹
- 불필요한 역할의 확대로 도메인 객체가 인프라에 대한 의존성을 가져가고 싶은 유혹



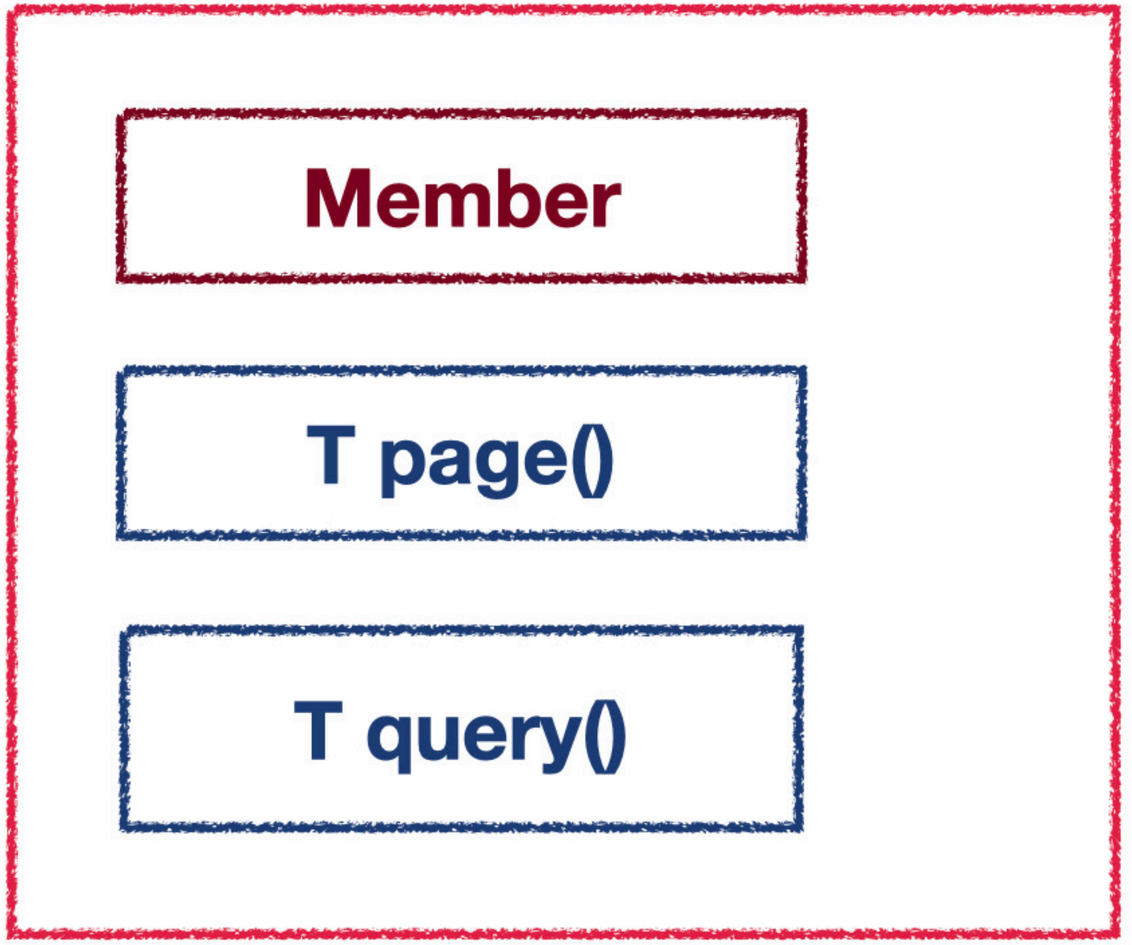
역할의 재조정

- Member에서 요청에 관련한 역할 분할
- 요청에 관련된 역할 Request 객체에게 할당

의존성 정리 어려움

역할 분할

- 요청에 대한 책임 담당할 인터페이스 생성
- 행동 정의
  - 요청에 대한 쿼리 생성
  - 요청에 대한 페이징 쿼리 생성
- MemberRequest 인터페이스로 레이어 간 메시지 전송
- 메소드 구현은 구현체에서 DB에 알맞는 타입으로 구현



의존성 정리 어려움

역할 분할

- 요청에 대한 책
- 행동 정의
  - 요청에 대
  - 요청에 대
- MemberRequest 타입으로 레이어 간 메시지 전송
- 메소드 구현은 구현체에서 DB에 알맞는 타입으로 구현

MemberRequest<T>

인터페이스들에 알맞는 역할들 부여

규약에 어긋나는 의존성 방향 정리 가능

T query()

의존성 정리 어려움

역할 분할

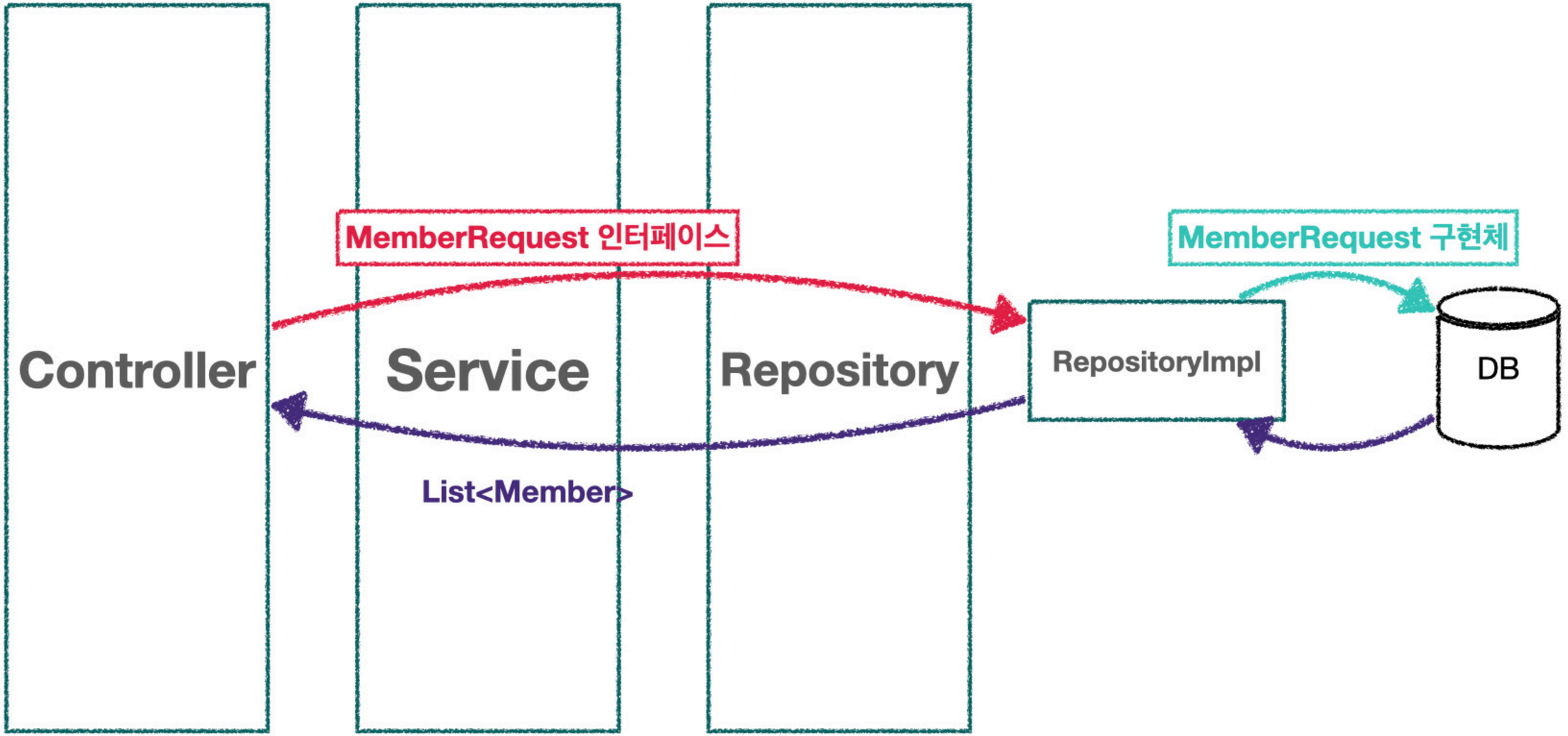
MemberRequest<T>

중요한 건 역할

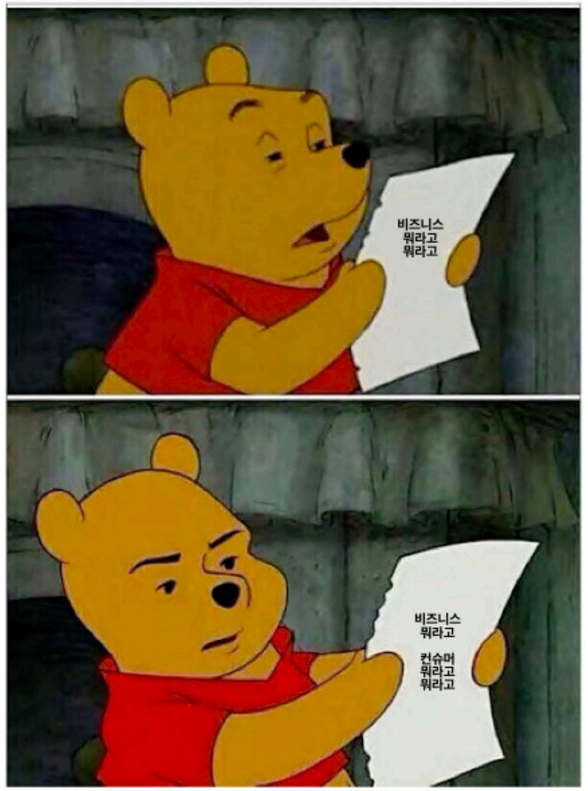
- 요청
- 행동
  - 요
  - 요
- MemberRequest 타입으
- 메소드 구현은 구현체에서 구현

query()

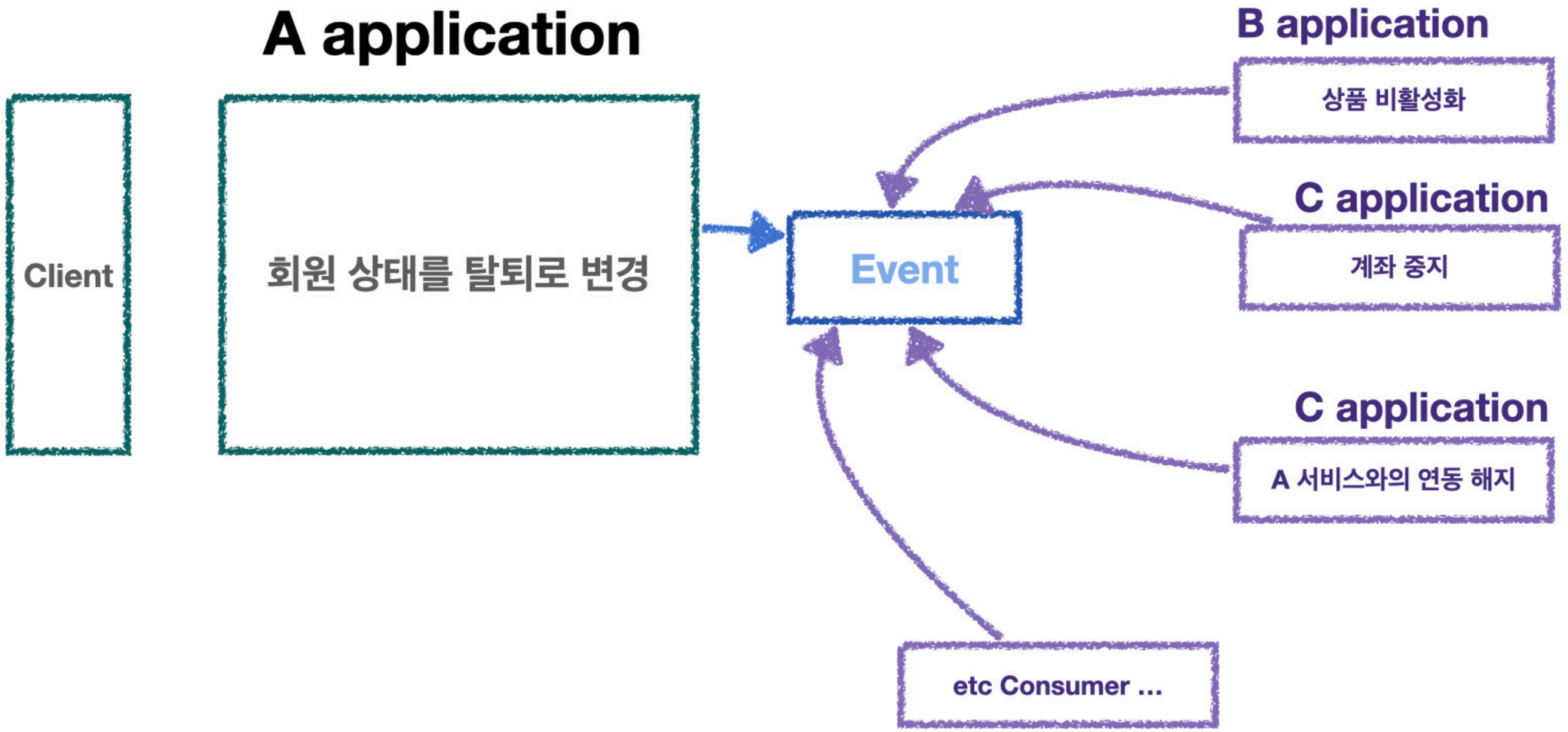
의존성 정리 어려움


Controller
Service
Repository
MemberRequest 인터페이스
MemberRequest 구현체
RepositoryImpl
DB
List<Member>

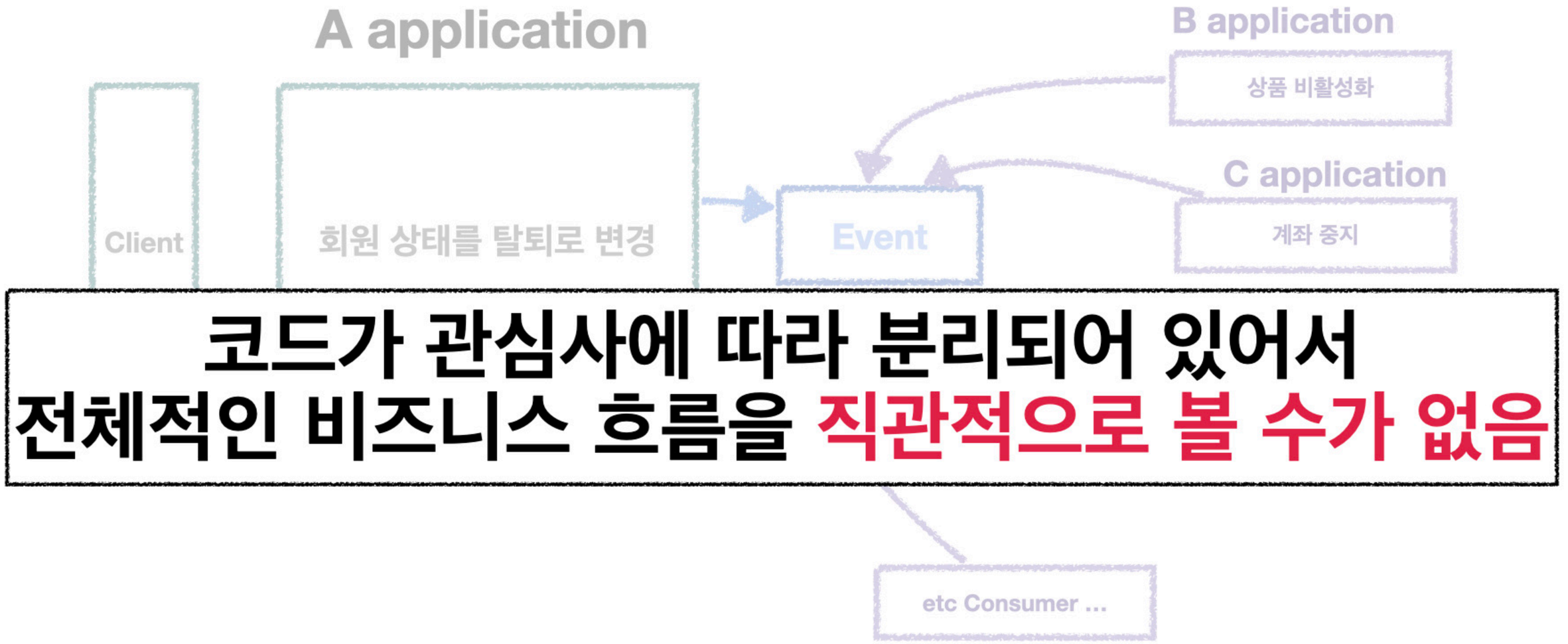
눈에 보이지 않는 비즈니스 흐름



비직관적인 비즈니스 흐름



비직관적인 비즈니스 흐름



코드가 관심사에 따라 분리되어 있어서
전체적인 비즈니스 흐름을 직관적으로 볼 수가 없음

비직관적인 비즈니스 흐름

코드의 직관성과 유연성은 trade off 되는 가치

비직관적인 비즈니스 흐름

코드의 직관성과 유연성은 trade off 되는 가치

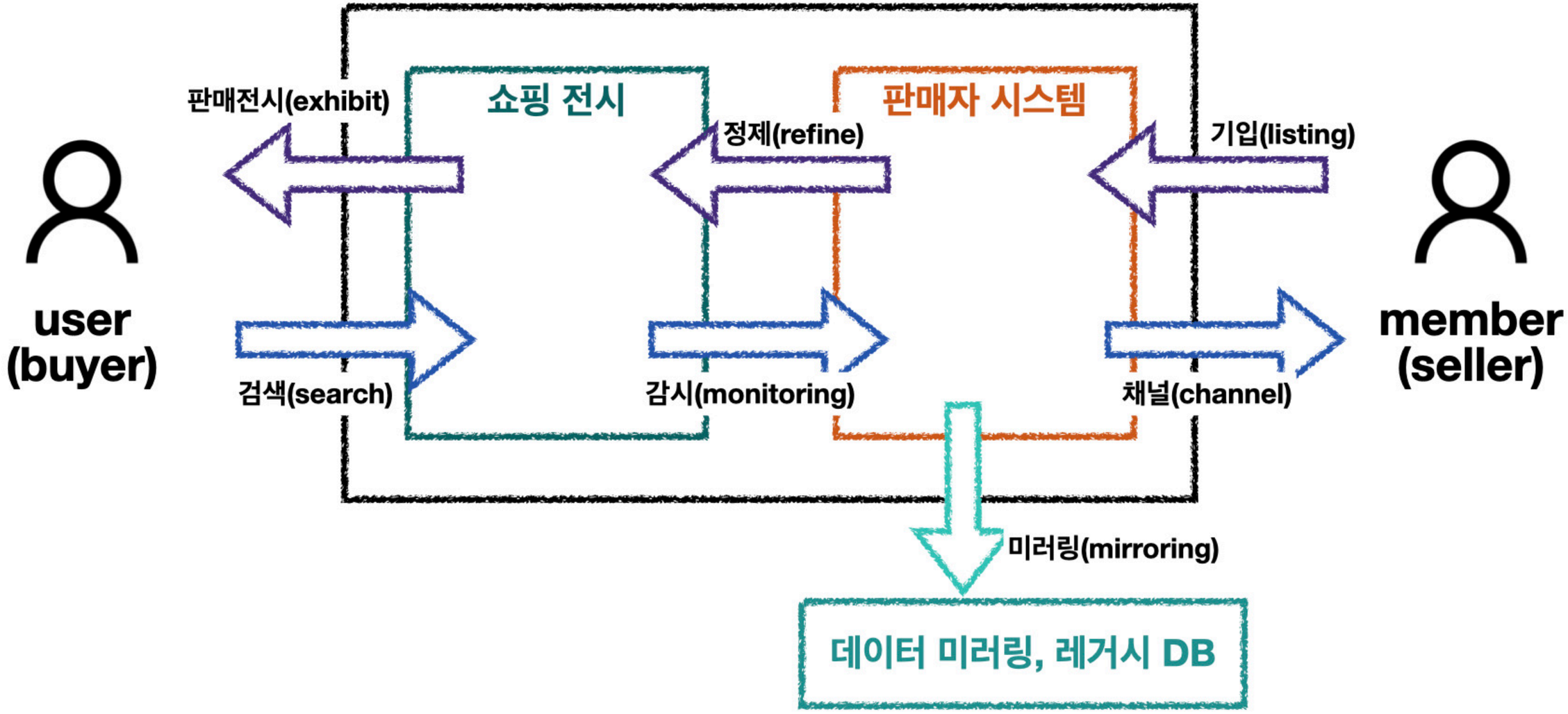
> 시스템의 유연성을 더 높은 가치로 평가

비직관적인 비즈니스 흐름

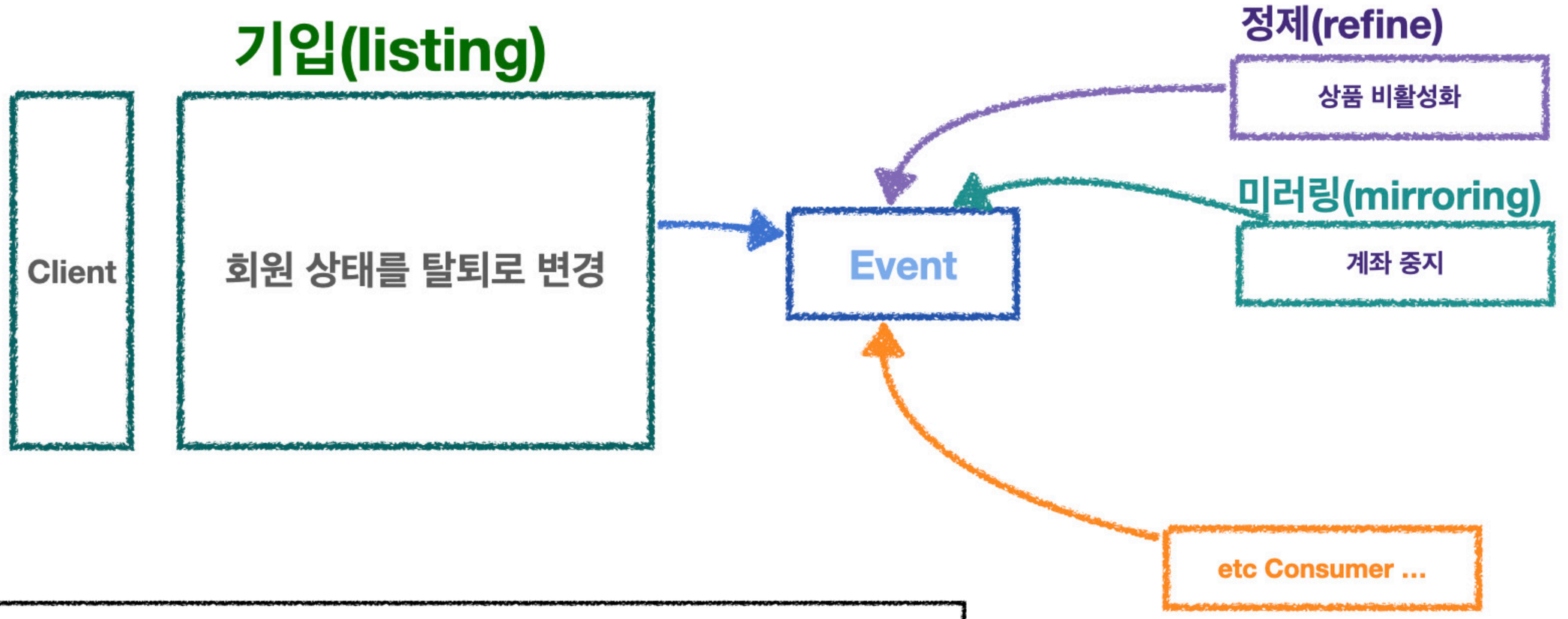
패키지, 애플리케이션은 역할과 책임으로 구분

> 코드의 위치는 코드가 어떤 책임을 가지냐에 따라 유추 가능

비직관적인 비즈니스 흐름



비직관적인 비즈니스 흐름



책임을 바탕으로 구현 위치 확인

마지막 소감

마지막 소감

여전히 진행중인 시스템 전환

마지막 소감

역할을 할당하는 고통
또한 현재 진행형

마지막 소감

옳은 방향으로 가고 있다는 믿음

마지막 소감

팀 내부에서의
합의와 공감대 형성


릴레이툰
MBC
웹툰 VS 현실의 만남
힘들지만 싸워볼게!

들어주셔서 감사합니다.