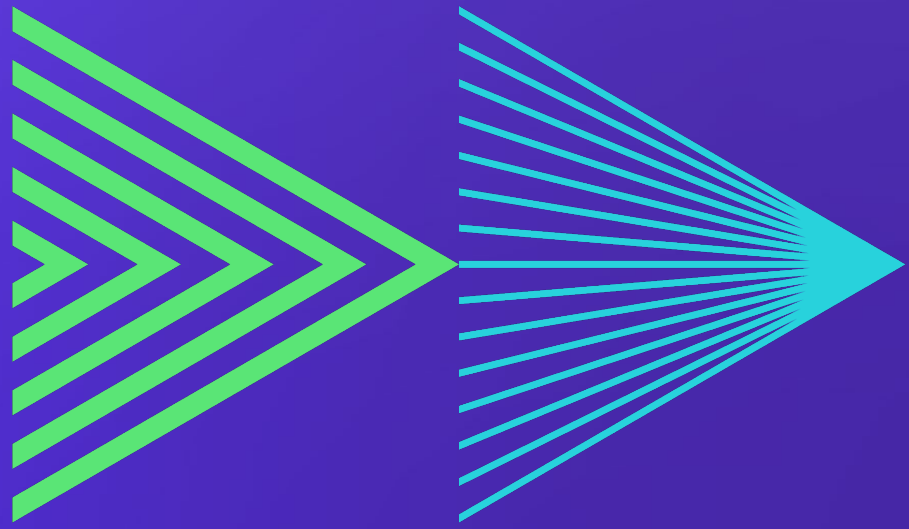
NHN FORWARD

Dooray! 모바일 앱의 클린 아키텍처 적용기

NHN Dooray 모바일서비스개발팀

이우람



두레이 소개

NHN FORWARD


프로젝트


메신저


홈/게시판


메일


화상 회의


근무 관리


드라이브


위키


결재


캘린더


주소록

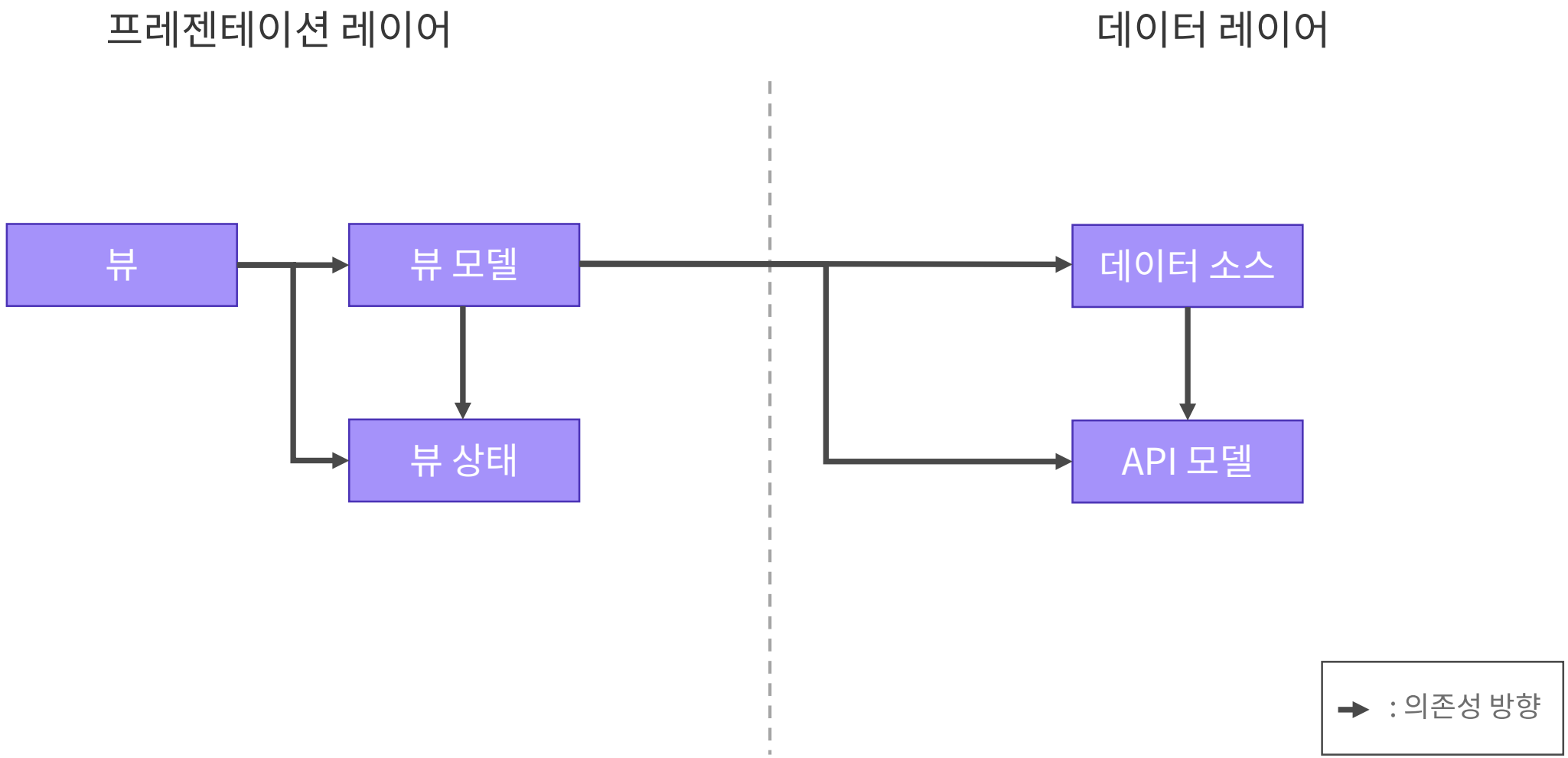

자원 예약

다룰 내용

1. 클린 아키텍처를 도입한 배경
2. 클린 아키텍처 소개
3. 클린 아키텍처 구현
4. 현재 두레이의 모습
5. 앞으로의 계획
6. QnA

클린 아키텍처를 도입한 배경

기존 구조



기존 뷰 모델의 문제점

복잡한 뷰 모델

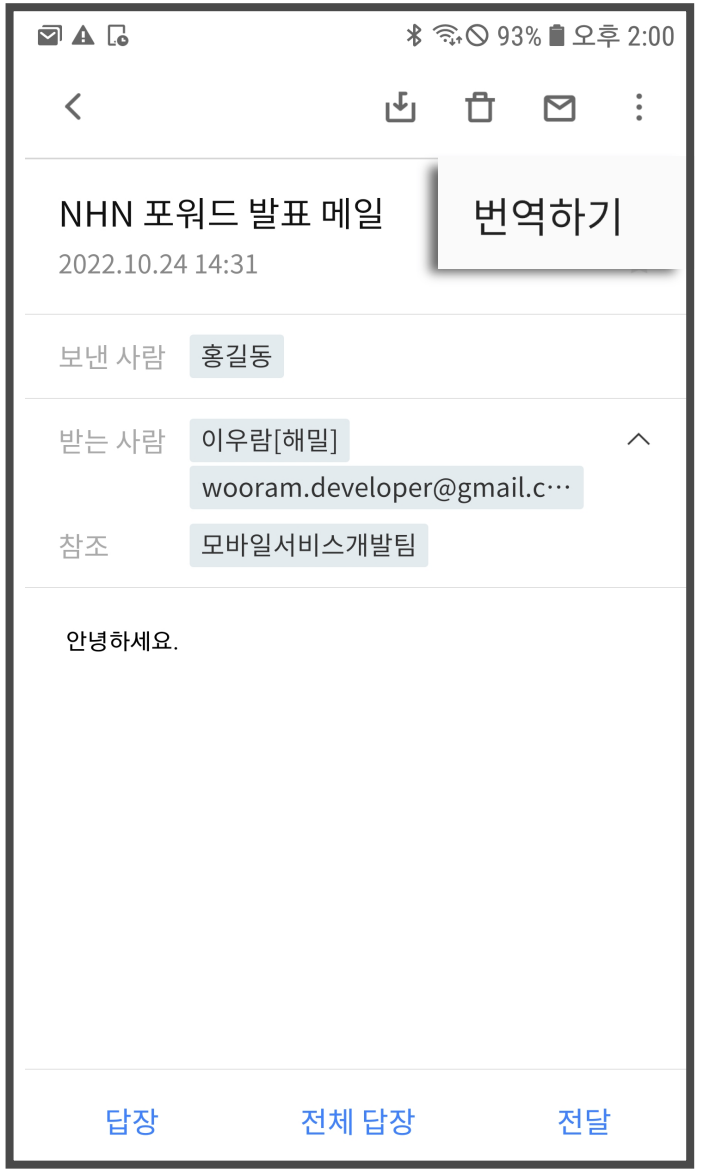
- 프레젠테이션 로직과 비즈니스 로직을 포함

뷰 모델이 외부 변화에 취약함

메일 읽기 화면 소개

NHN FORWARD

- 메일 제목, 본문, 수신자
- 언어 번역 기능



복잡한 뷰 모델

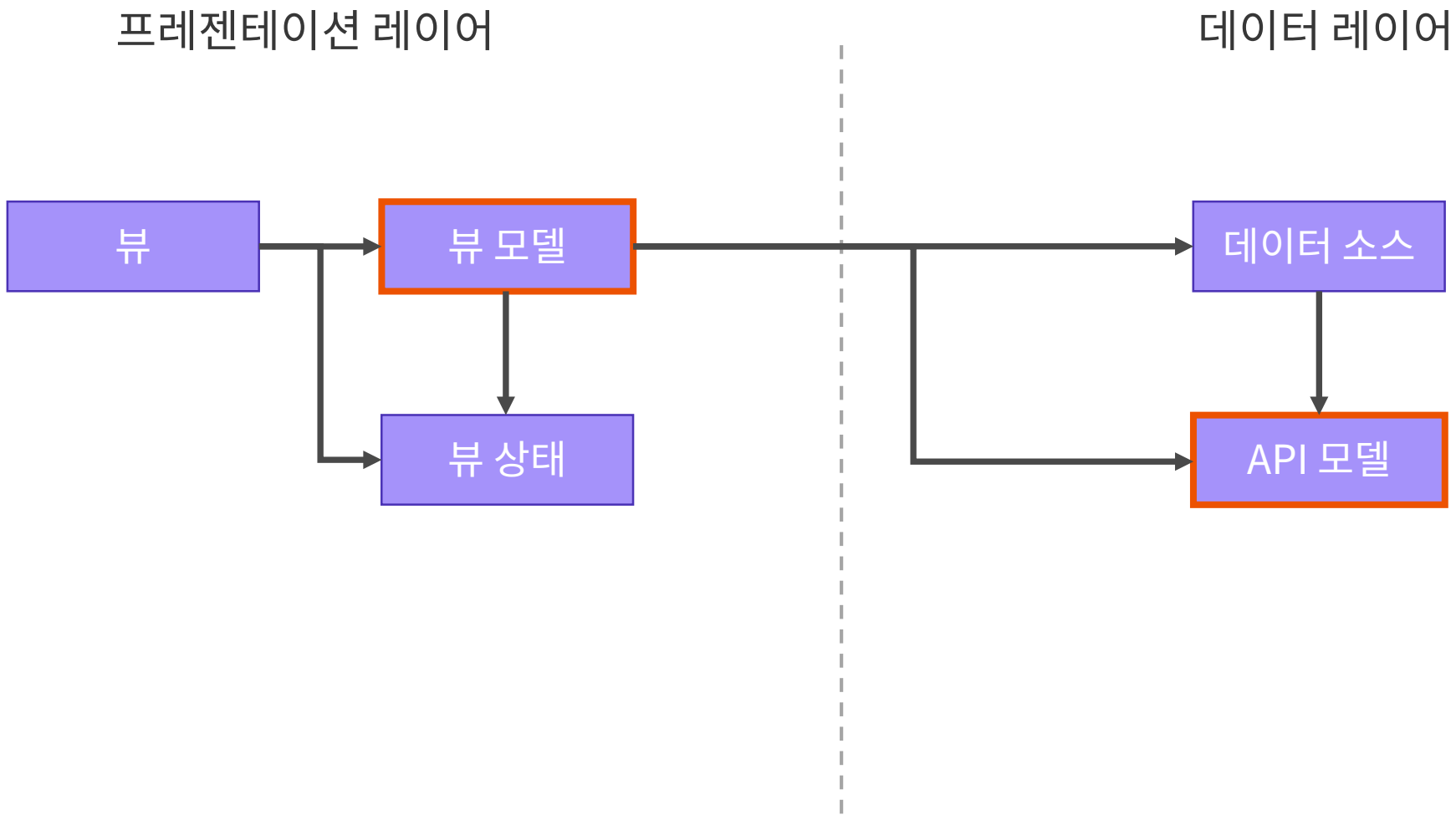
1. 요금제 유형 따라 번역 기능 활성화

2. 메일 정보 조회 API 호출

3. 수신자 유형에 따라 추가 정보 조회 API 호출

4. 뷰 상태 생성

외부 요인에 취약한 프레젠테이션



클린 아키텍처 소개

클린 아키텍처

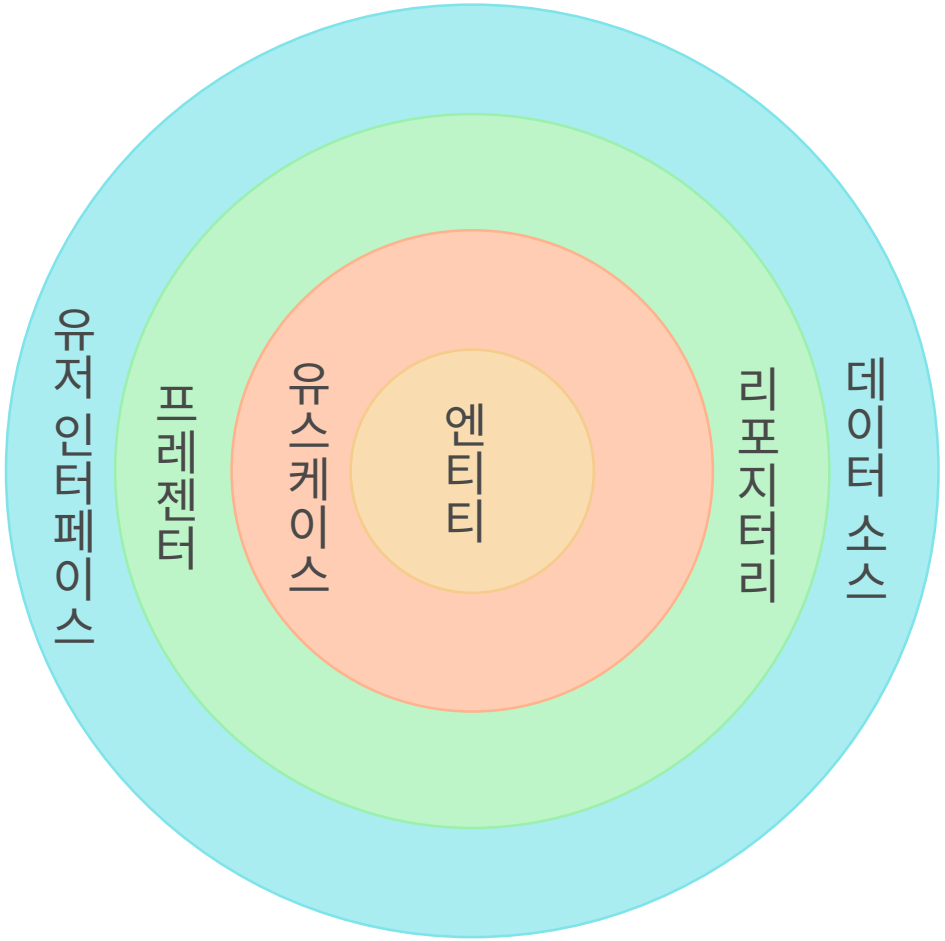
아키텍처란?

- 소프트웨어의 구성 요소들의 유기적 관계를 설명
- 소프트웨어의 설계와 업그레이드를 통제하는 원칙

클린 아키텍처란?

- 유연하게 변경하며 견고한 구조를 만들기 위한 소프트웨어 디자인 철학

클린 아키텍처 다이어그램



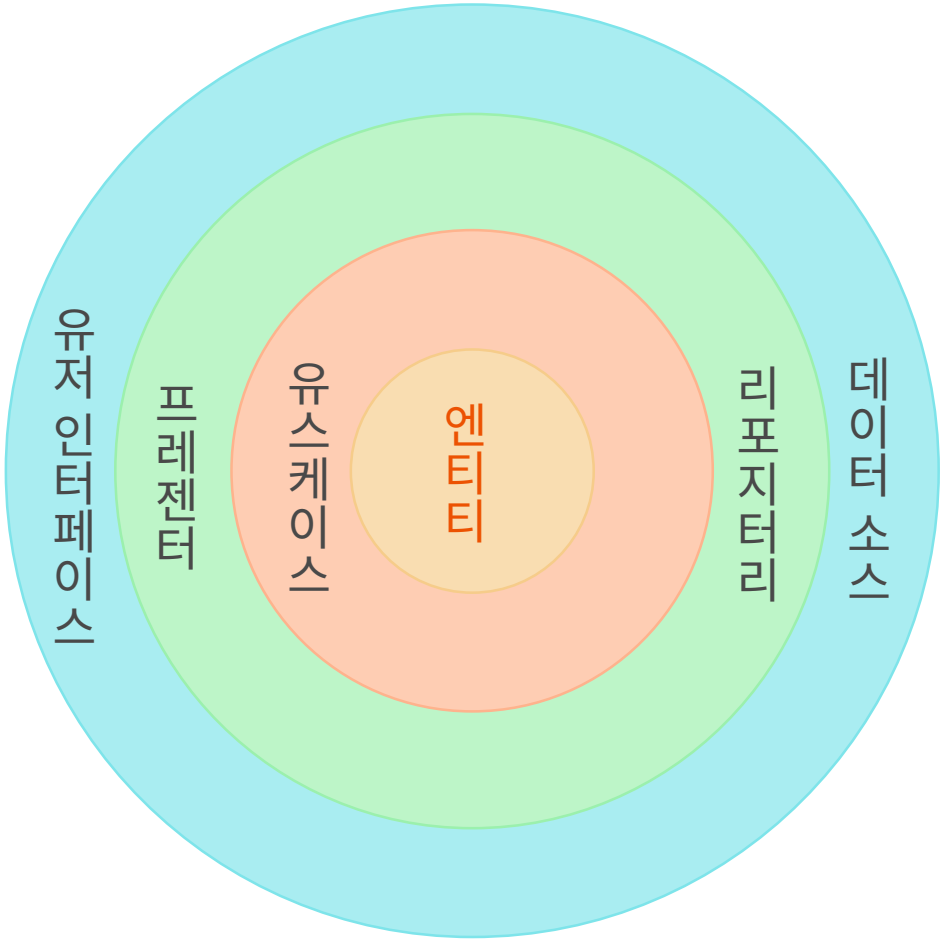
엔티티와 도메인 지식

엔티티란?

- 도메인 지식을 기반으로 설계된 클래스

도메인 지식이란?

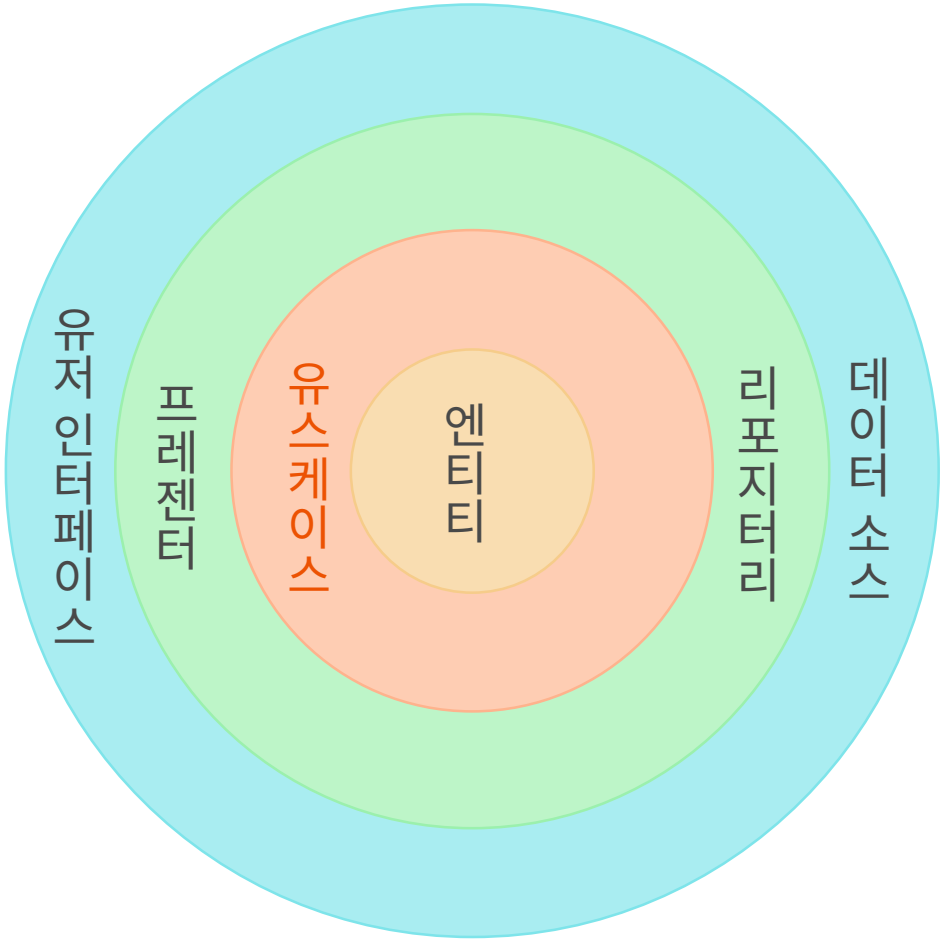
- 서비스에 대한 전문적인 지식



유스케이스

유스케이스란?

- 사용자와 시스템 간 주요 활동을 기술
- 엔티티 간 데이터 흐름을 조정



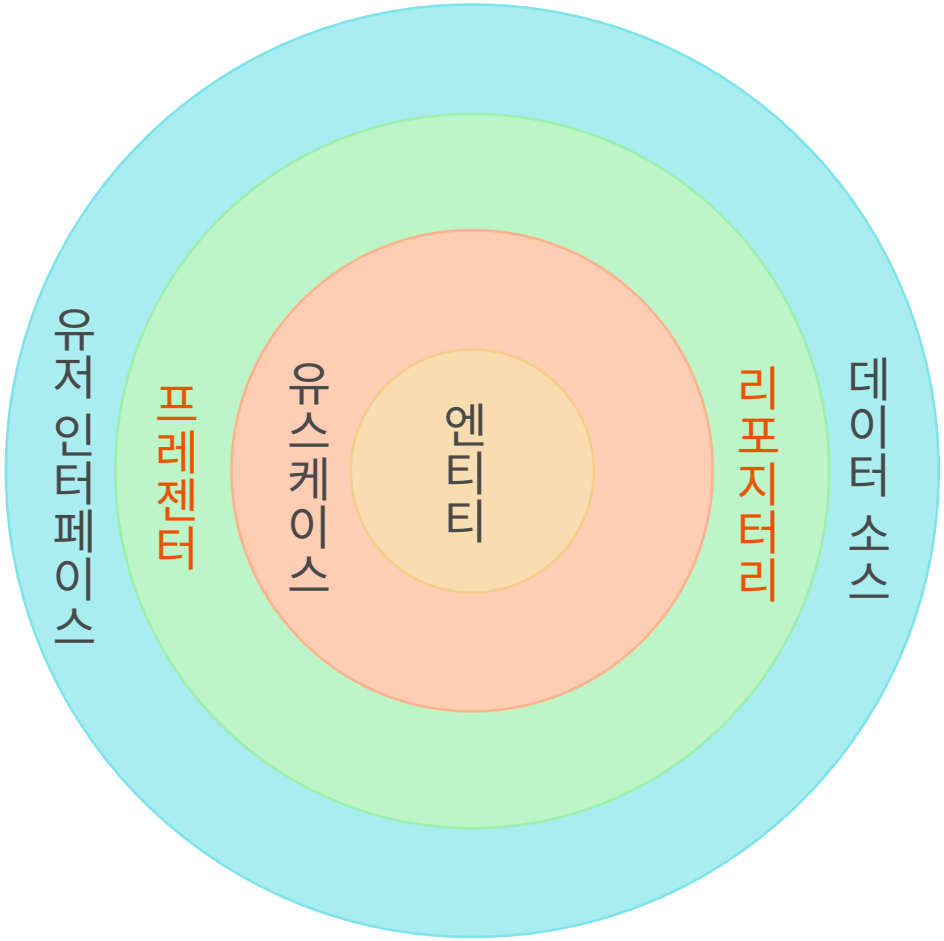
리포지터리와 프레젠터

리포지터리란?

- 데이터를 {생성, 저장, 변경} 방식을 결정
- 엔티티를 전달

프레젠터란?

- 뷰 상태를 생성



데이터 소스와 유저 인터페이스

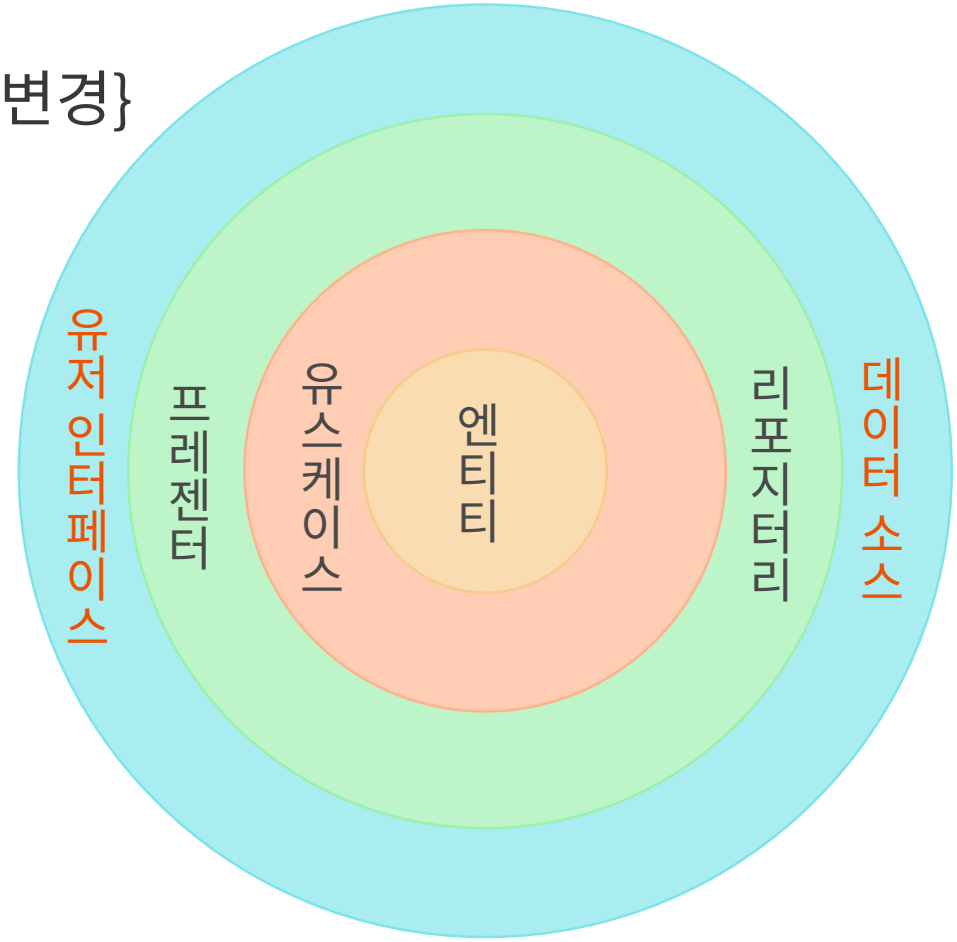
NHN FORWARD

데이터 소스란?

- 네트워크 통신 등을 통해 데이터를 {생성, 저장, 변경}

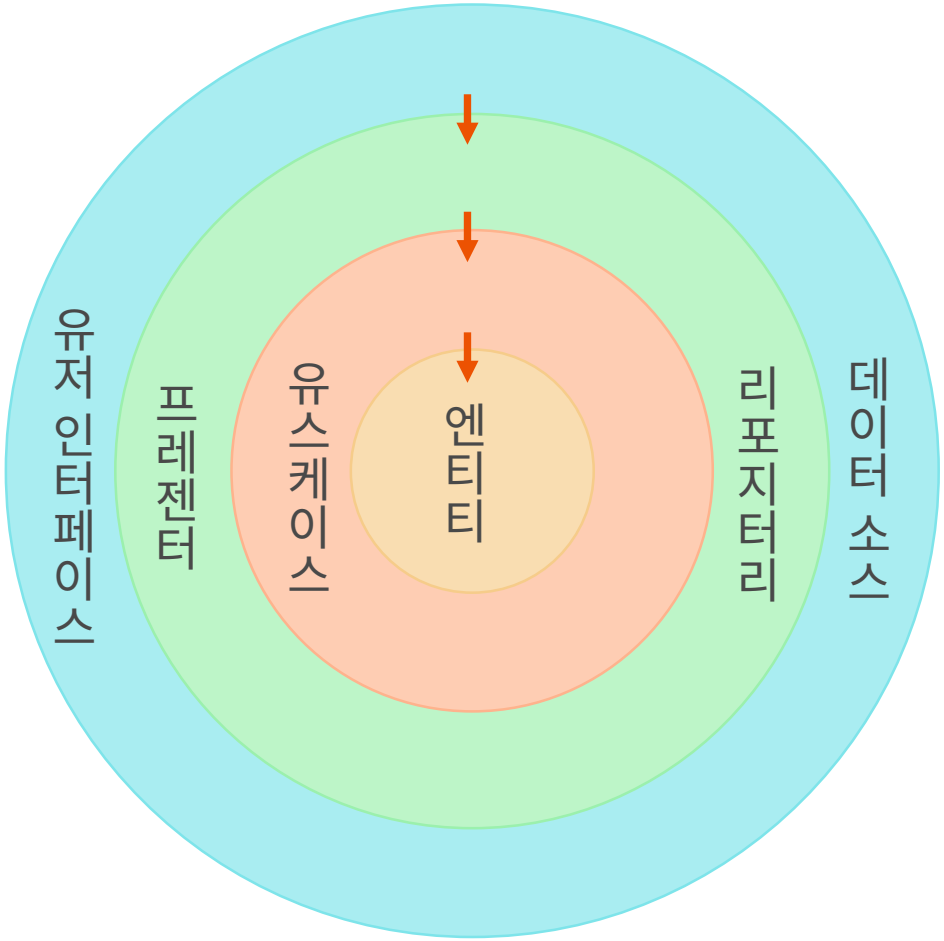
유저 인터페이스란?

- 사용자와 상호작용 및 데이터를 화면에 표시

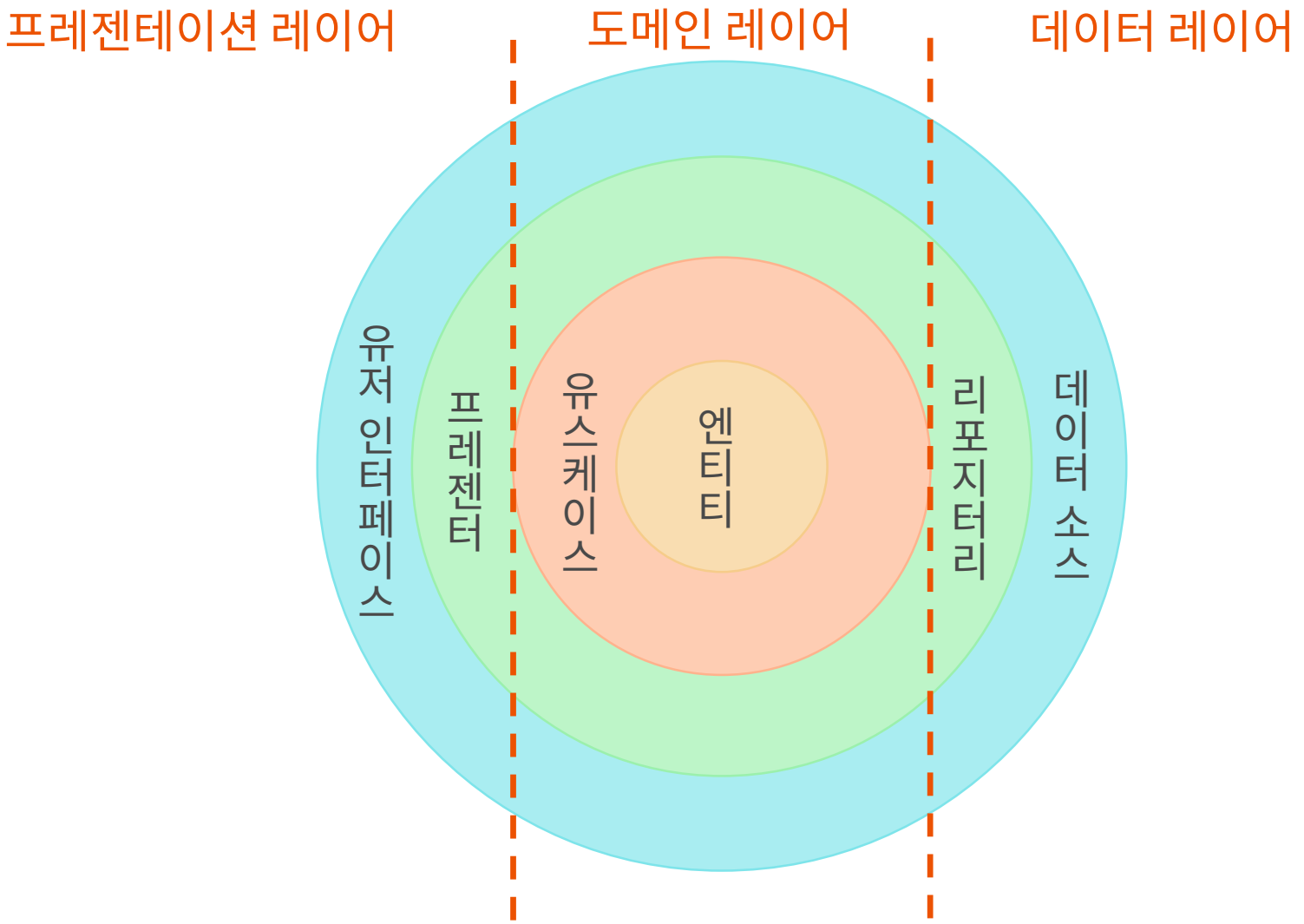


클린 아키텍처 설계 원칙

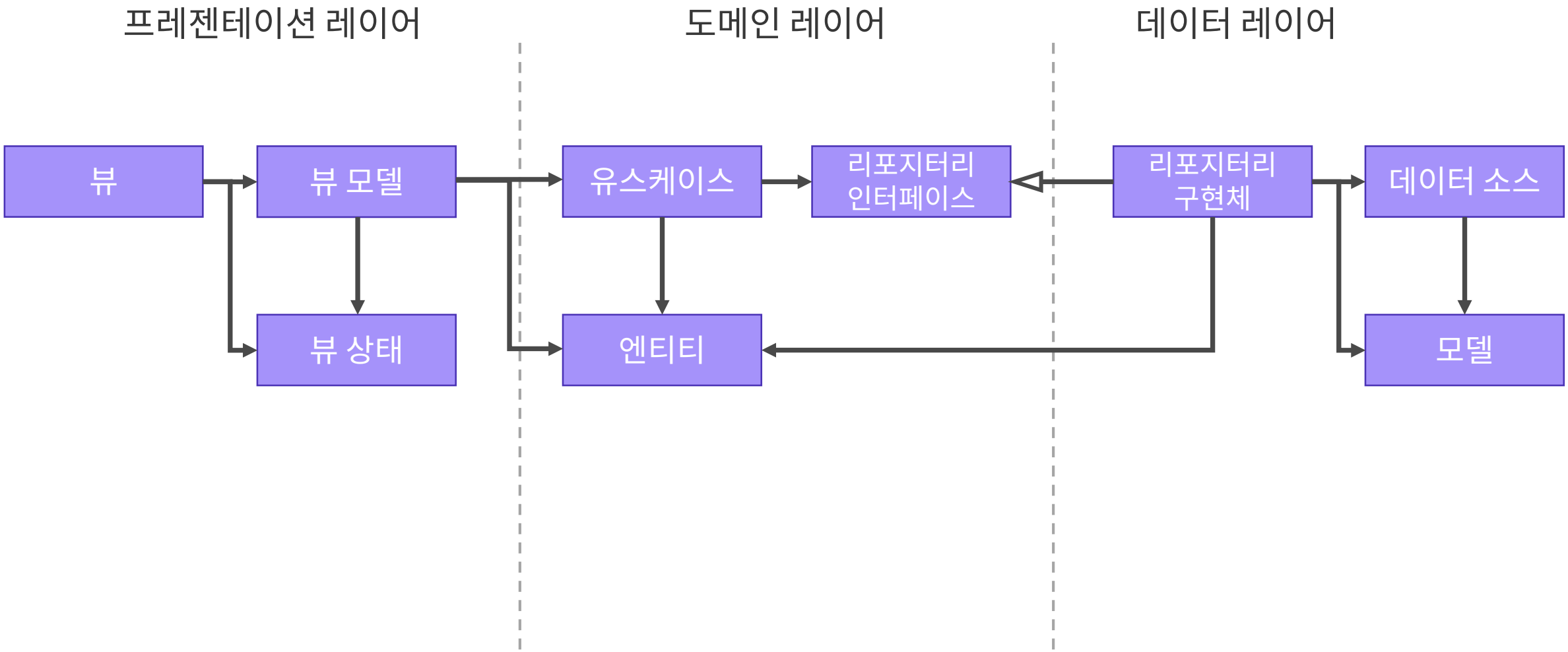
- 구성 요소를 계층적으로 분리
- 코드 변동성은 원 바깥으로 향할수록 큼
- 의존성은 원 바깥쪽에서 안쪽으로 향함



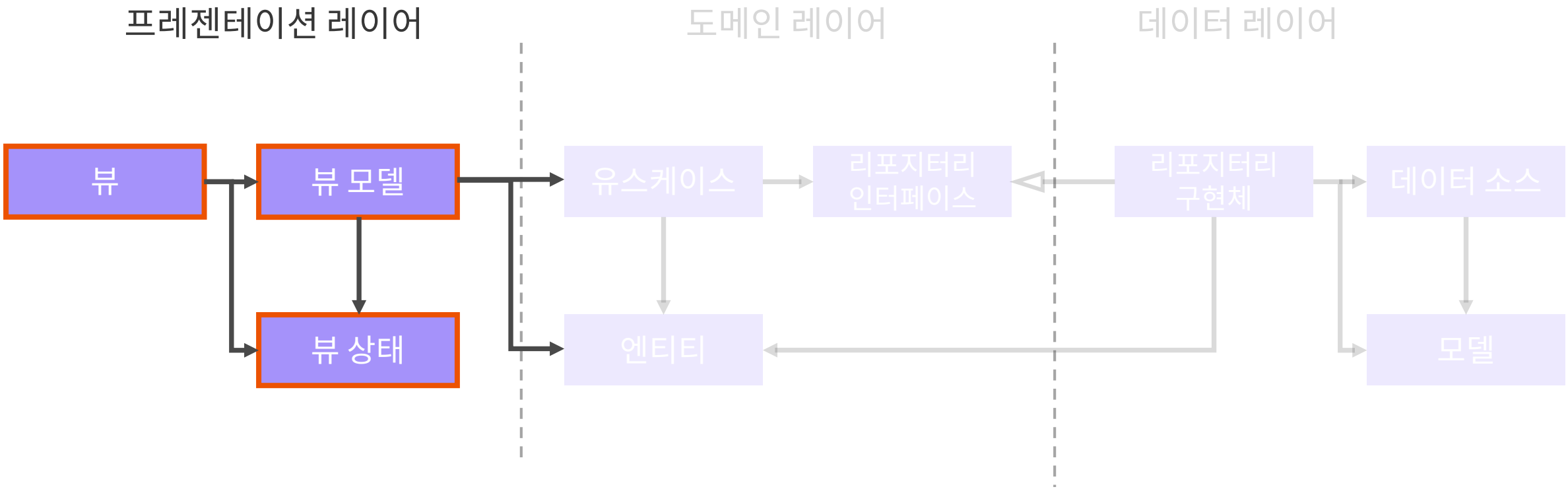
클린 아키텍처 레이어


프레젠테이션 레이어
도메인 레이어
데이터 레이어
유저 인터페이스
프레젠터
유스케이스
엔티티
리포지터리
데이터 소스

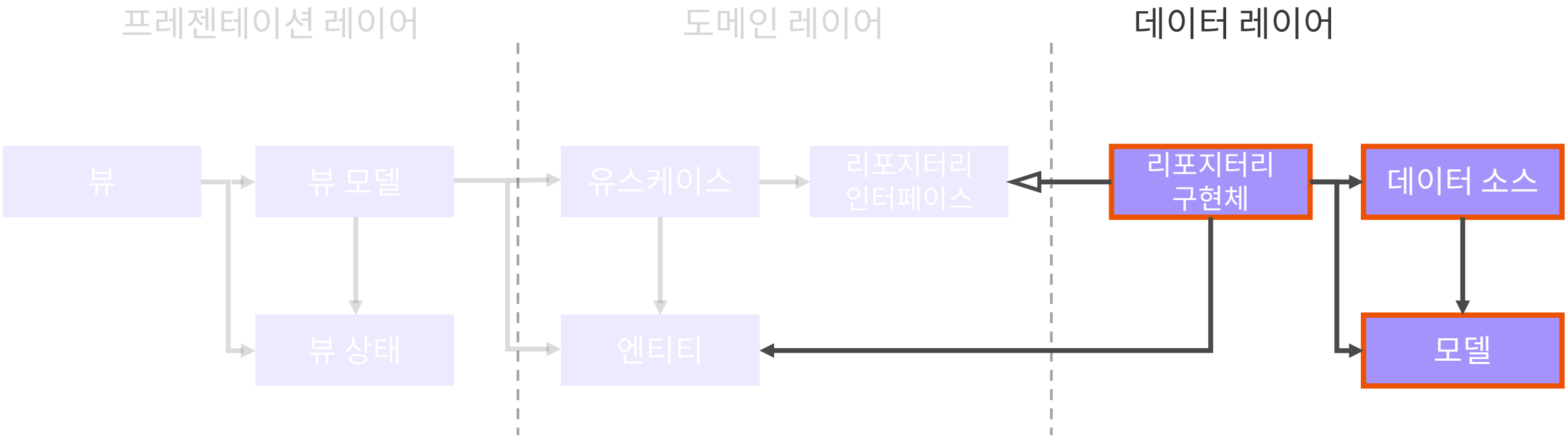
클린 아키텍처 레이어


프레젠테이션 레이어
도메인 레이어
데이터 레이어
뷰
뷰 모델
뷰 상태
유스케이스
엔티티
리포지터리
인터페이스
리포지터리
구현체
데이터 소스
모델

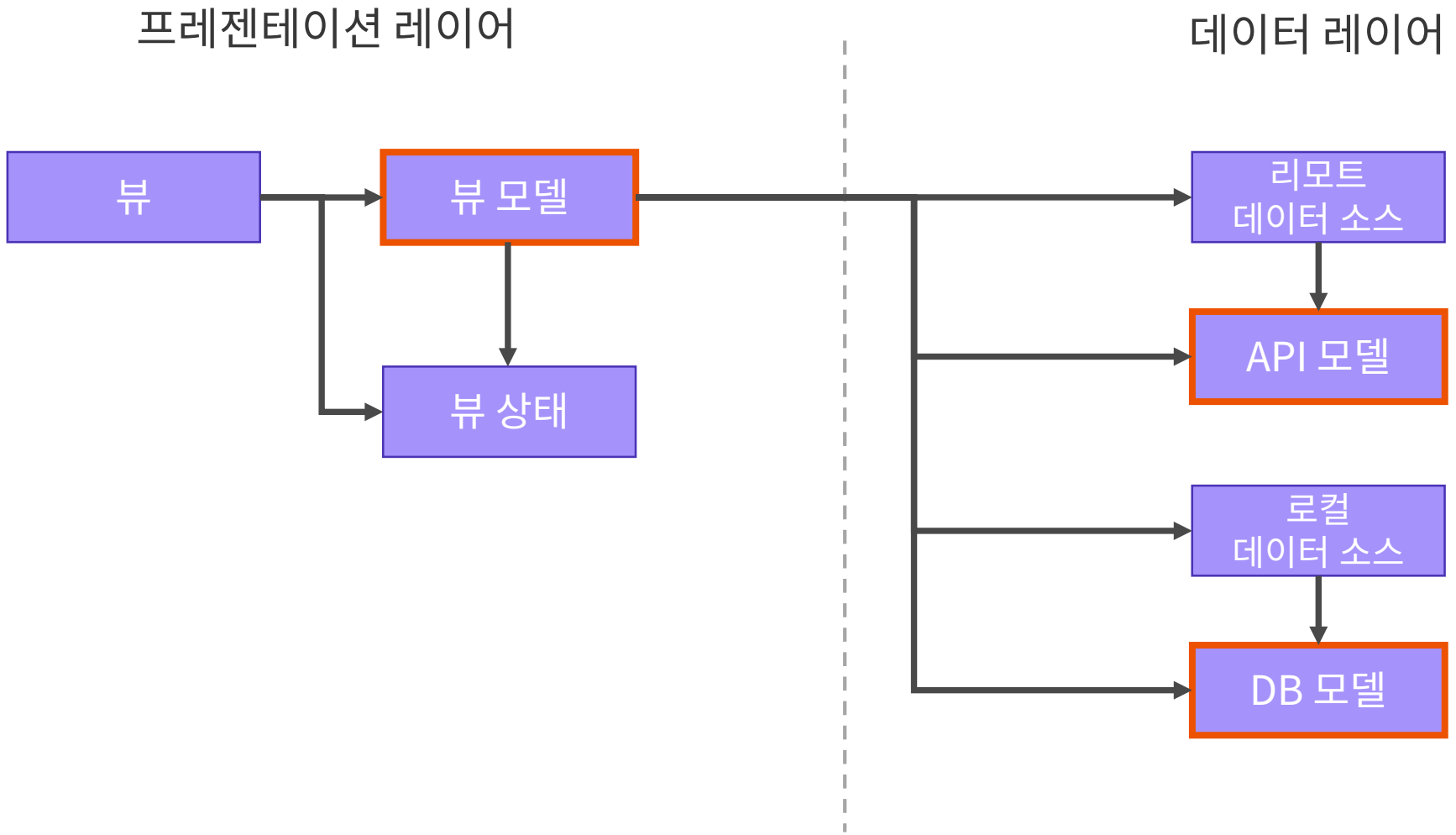
프레젠테이션 레이어


프레젠테이션 레이어
도메인 레이어
데이터 레이어
뷰
뷰 모델
뷰 상태
유스케이스
리포지터리 인터페이스
엔티티
리포지터리 구현체
데이터 소스
모델

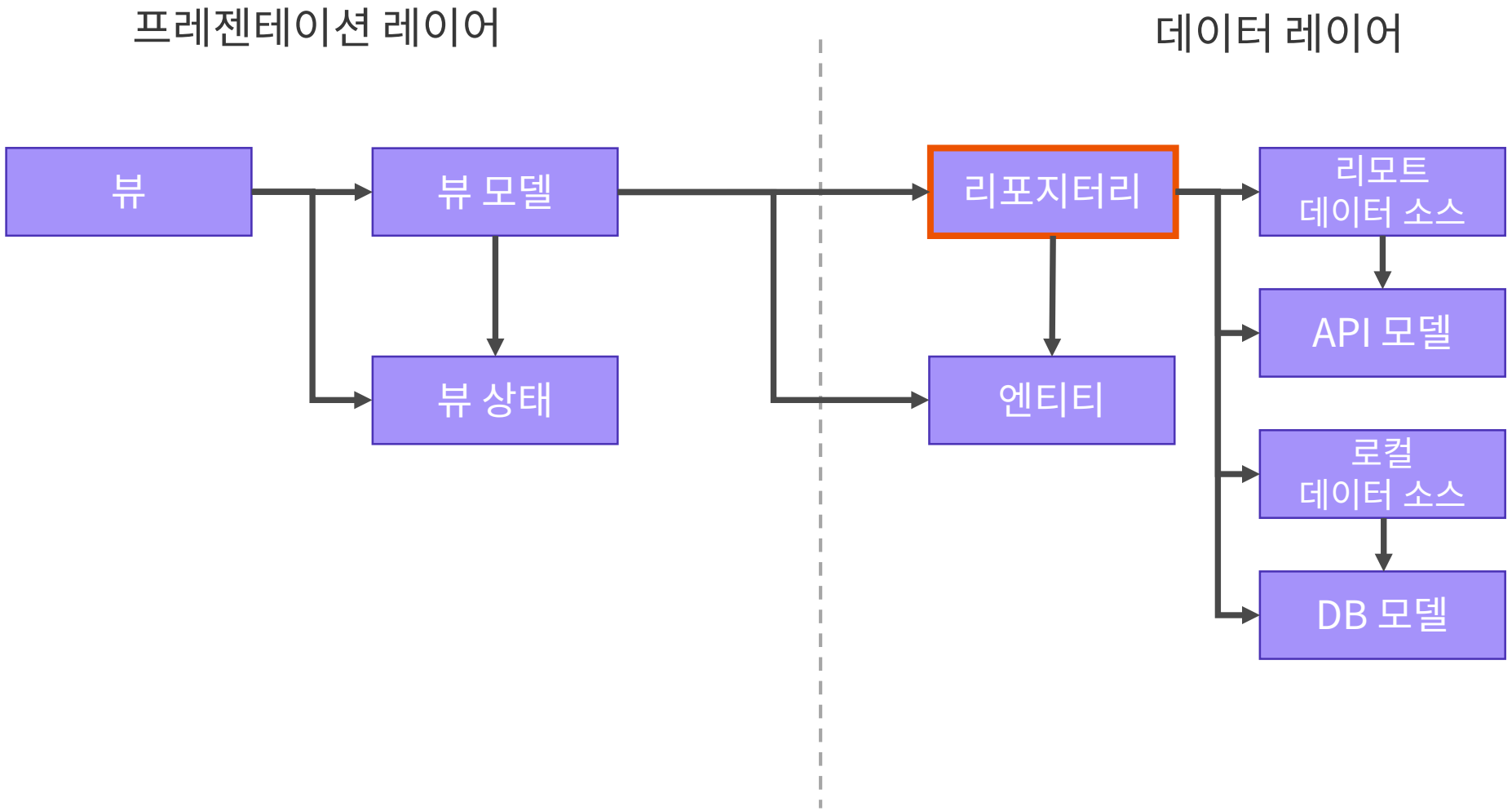
데이터 레이어


프레젠테이션 레이어
도메인 레이어
데이터 레이어
뷰
뷰 모델
뷰 상태
유스케이스
엔티티
리포지터리
인터페이스
리포지터리
구현체
데이터 소스
모델

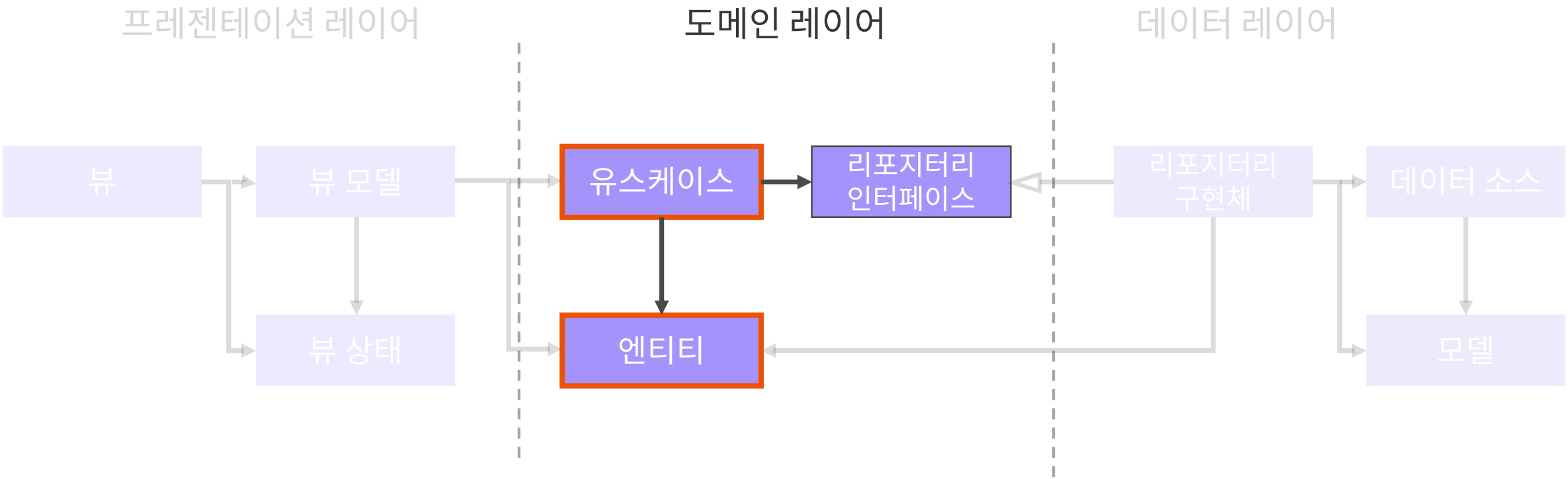
리포지터리가 필요한 이유



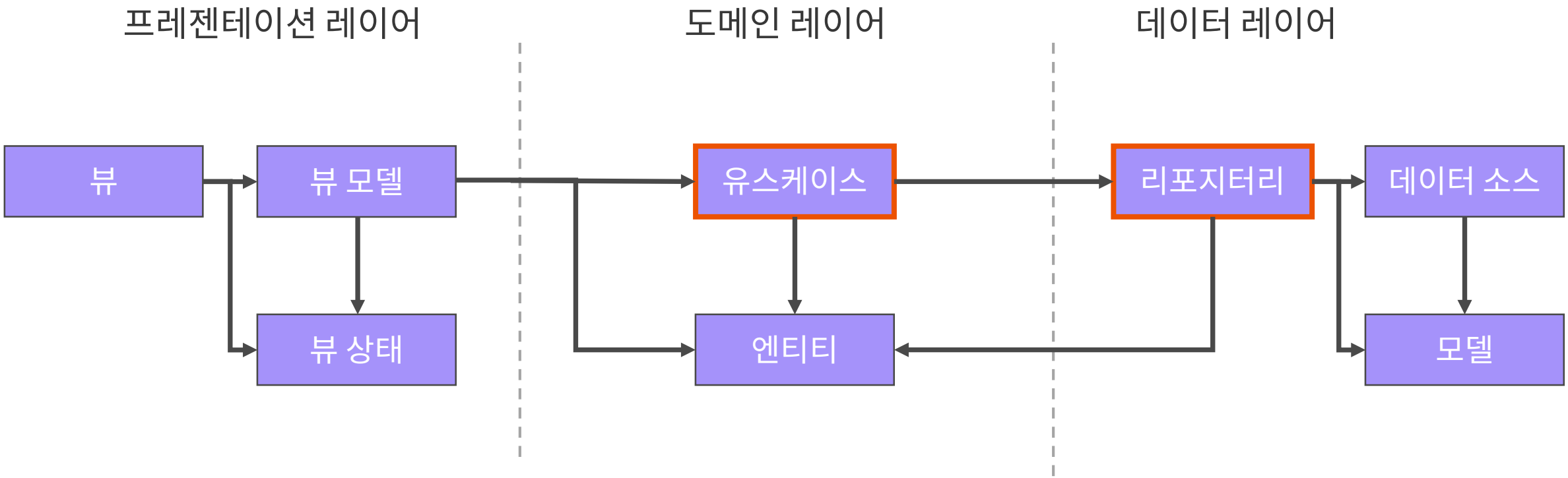
리포지터리가 필요한 이유



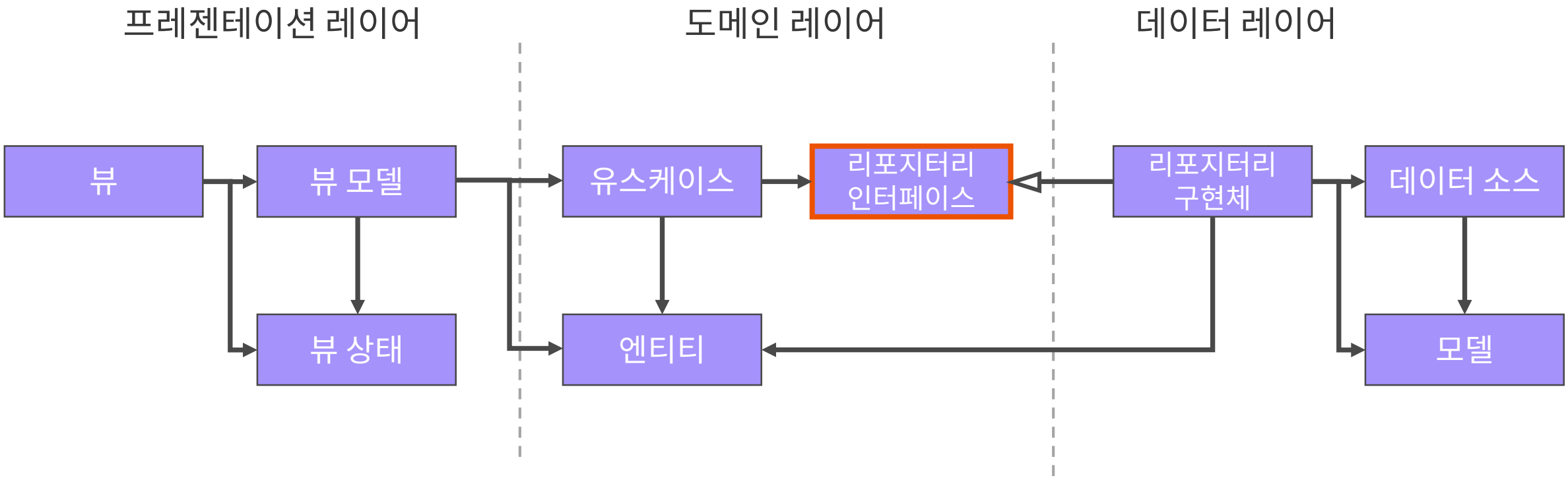
도메인 레이어


프레젠테이션 레이어
도메인 레이어
데이터 레이어
뷰
뷰 모델
뷰 상태
유스케이스
엔티티
리포지터리
인터페이스
리포지터리
구현체
데이터 소스
모델

리포지터리 인터페이스가 필요한 이유

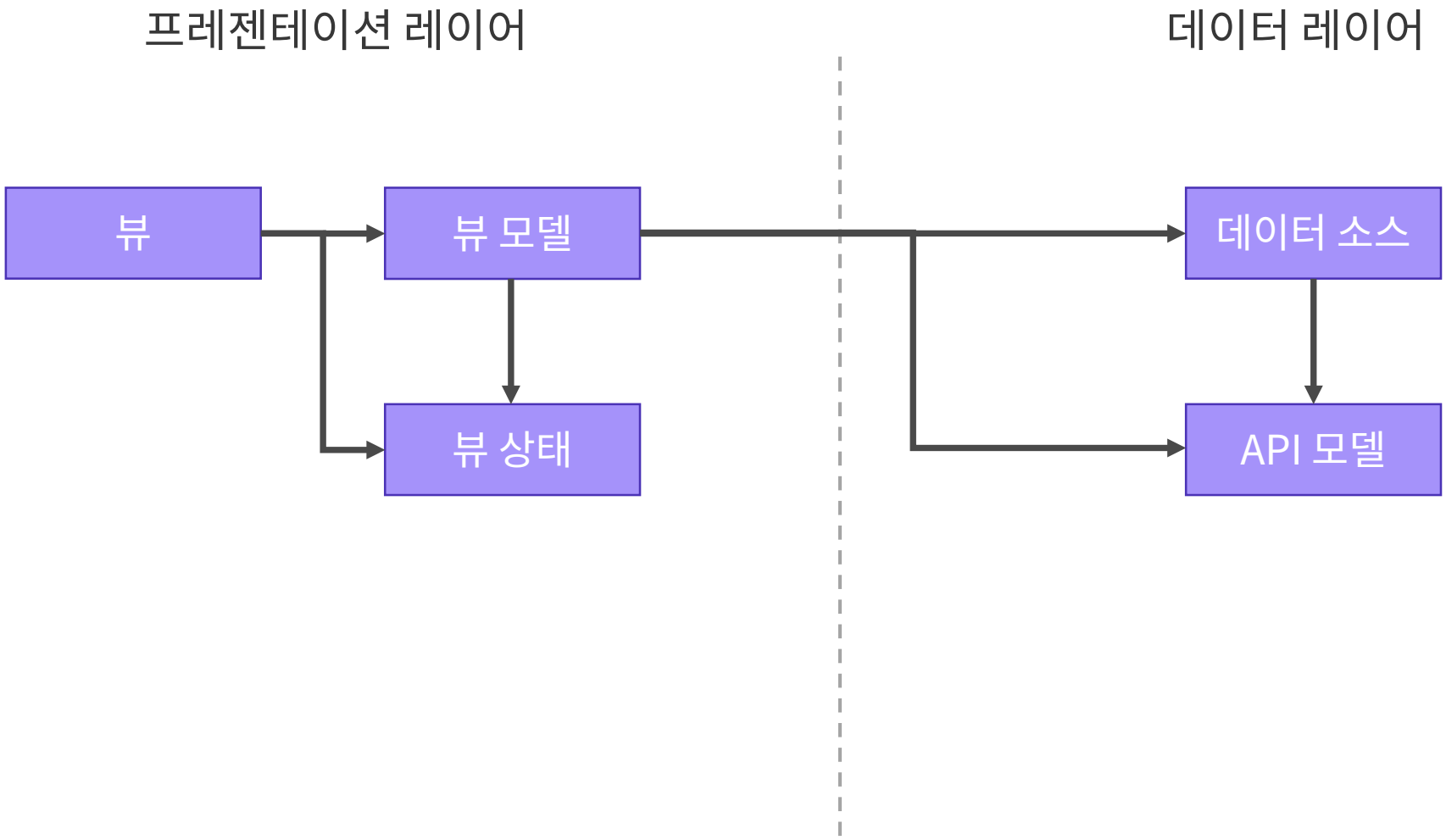


리포지터리 인터페이스가 필요한 이유

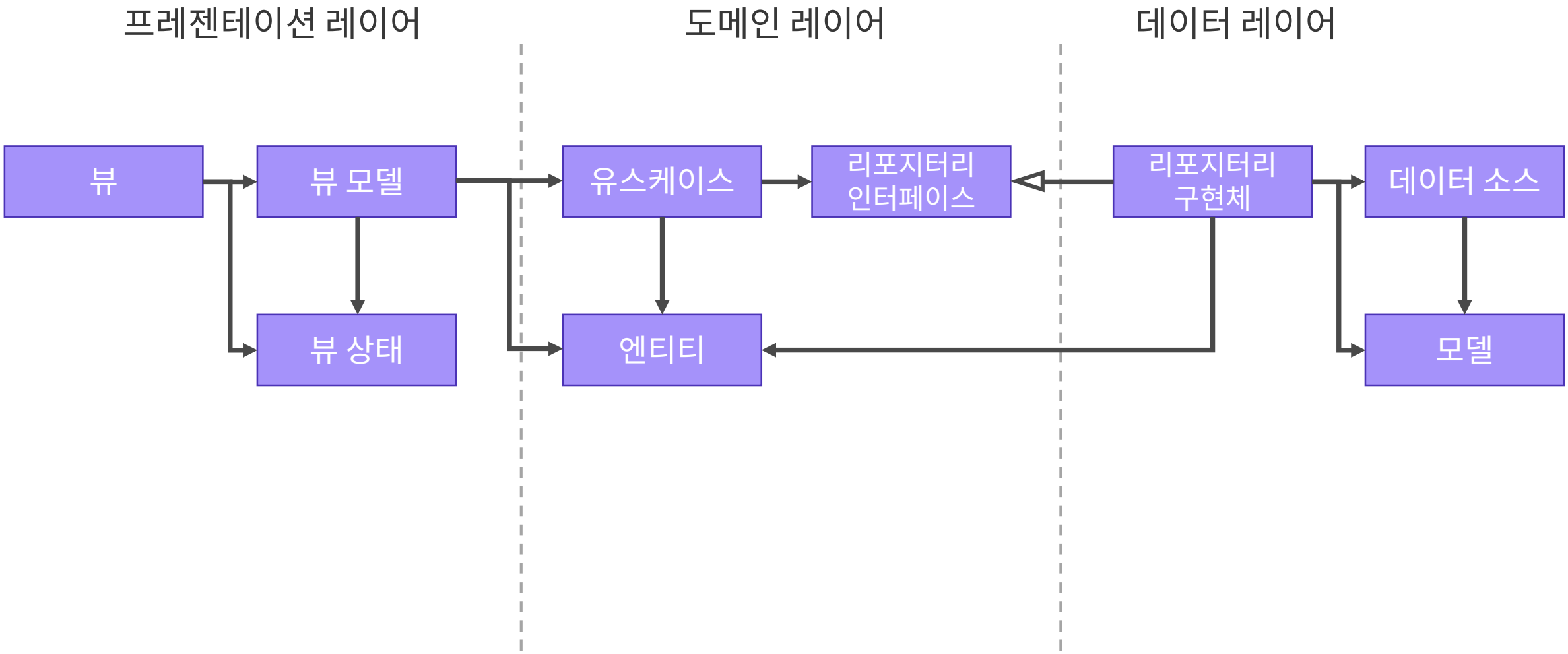


클린 아키텍처 구현

메일 읽기 화면 변경 전



메일 읽기 화면 변경 후


프레젠테이션 레이어
도메인 레이어
데이터 레이어
뷰
뷰 모델
뷰 상태
유스케이스
엔티티
리포지터리
인터페이스
리포지터리
구현체
데이터 소스
모델

도메인 지식 파악

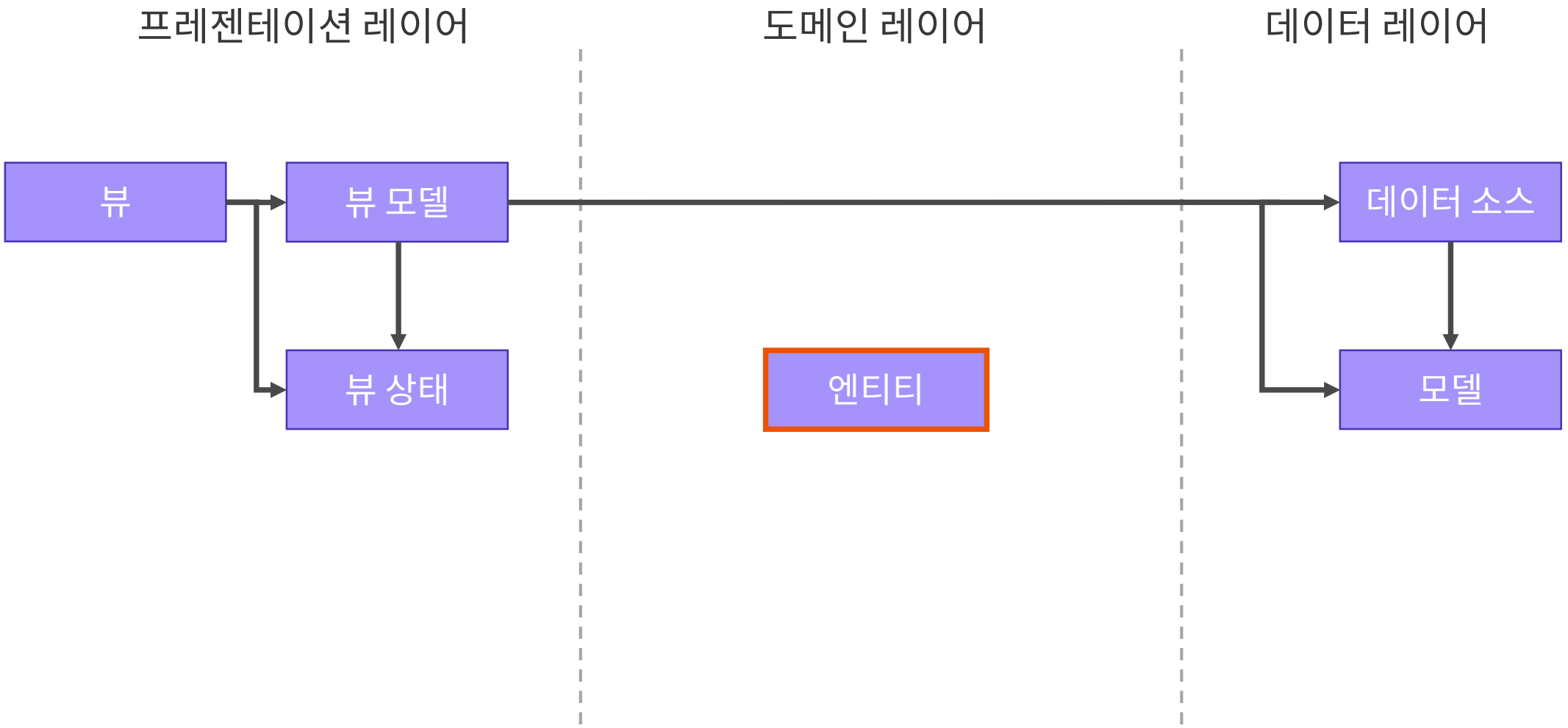
메일 도메인 지식

- 메일은 {제목, 본문, 수신자} 정보 포함
- 수신자 유형은 {직원, 부서, 외부 메일}

번역 상품 도메인 지식

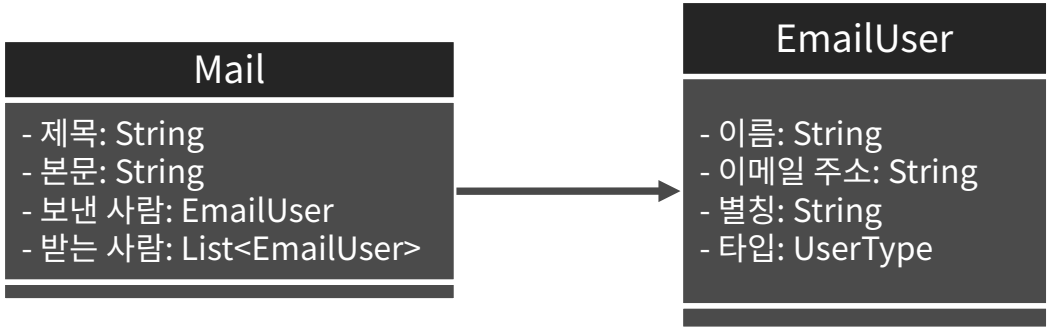
- 요금제 유형은 {프리, 베이식, 비즈니스, 엔터프라이즈}
- {프리, 베이식}은 번역 기능 비활성화

엔티티 구현

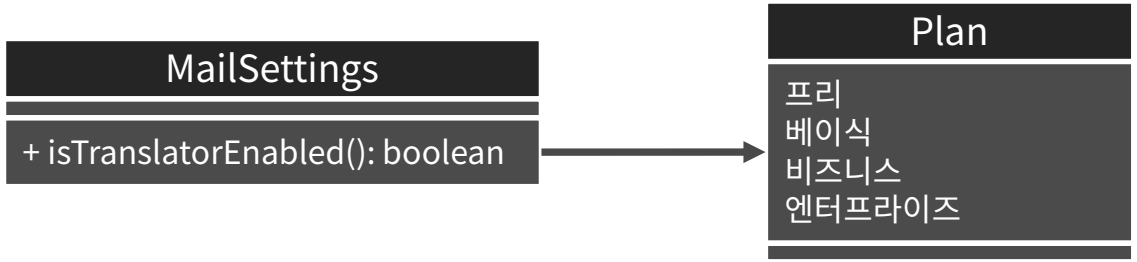


엔티티 구현 예시

메일 엔티티



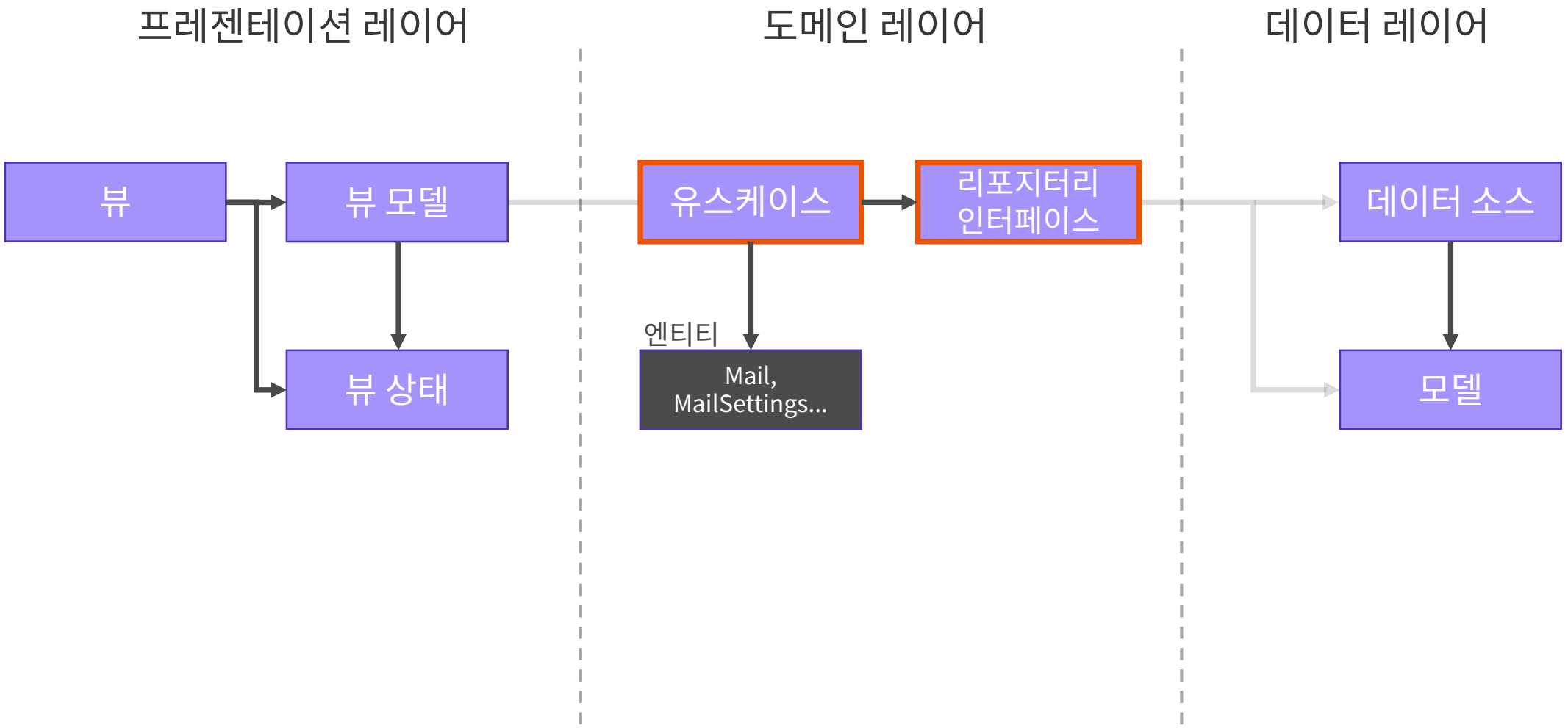
번역 상품 엔티티



번역 상품 엔티티 구현(Pseudo Code)


```
class MailSettings {
   // 요금제 유형
   Plan plan

   public boolean isTranslatorEnabled() {
      // 요금제 유형이 {비즈니스, 엔터프라이즈}일 경우 번역 기능 활성화
      return (Plan.비즈니스 is plan || Plan.엔터프라이즈 is plan)
   }
}
```

유스케이스 구현

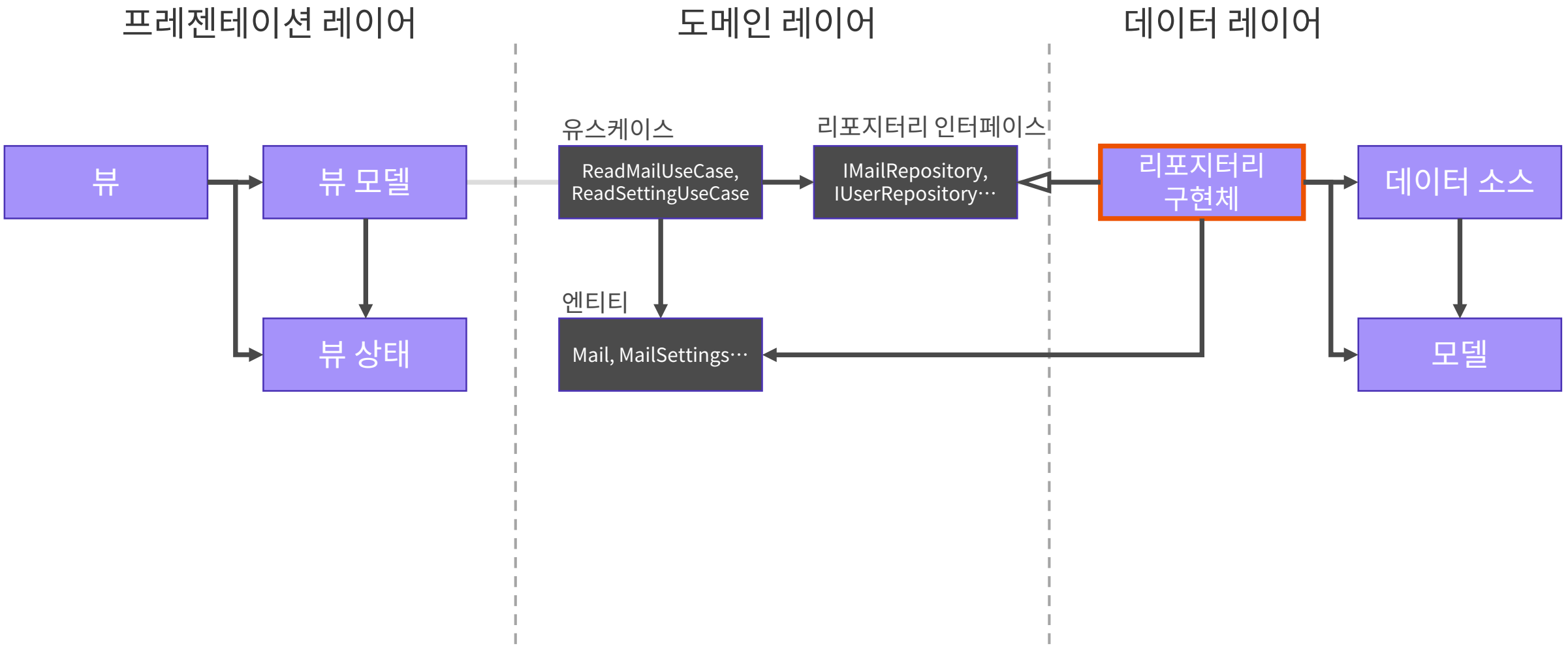

프레젠테이션 레이어
도메인 레이어
데이터 레이어
뷰
뷰 모델
뷰 상태
유스케이스
리포지터리 인터페이스
엔티티
Mail, MailSettings...
데이터 소스
모델

메일 조회 유스케이스 구현(Pseudo Code)

NHN FORWARD

```
class ReadMailUseCase {
   IMailRepository mailRepository
   IUserRepository userRepository

   public Mail getMail() {
      Mail mail = mailRepository.getMail() // 메일 정보를 조회

      if (“사내 직원” is mail.getTo()) { // 수신자가 사내 직원이면

         Member member = userRepository.getMember() // 직원 정보 추가 조회

         mail.setTo(member) // 수신자 정보 업데이트
      }
      return mail
   }
}
```

리포지터리 구현

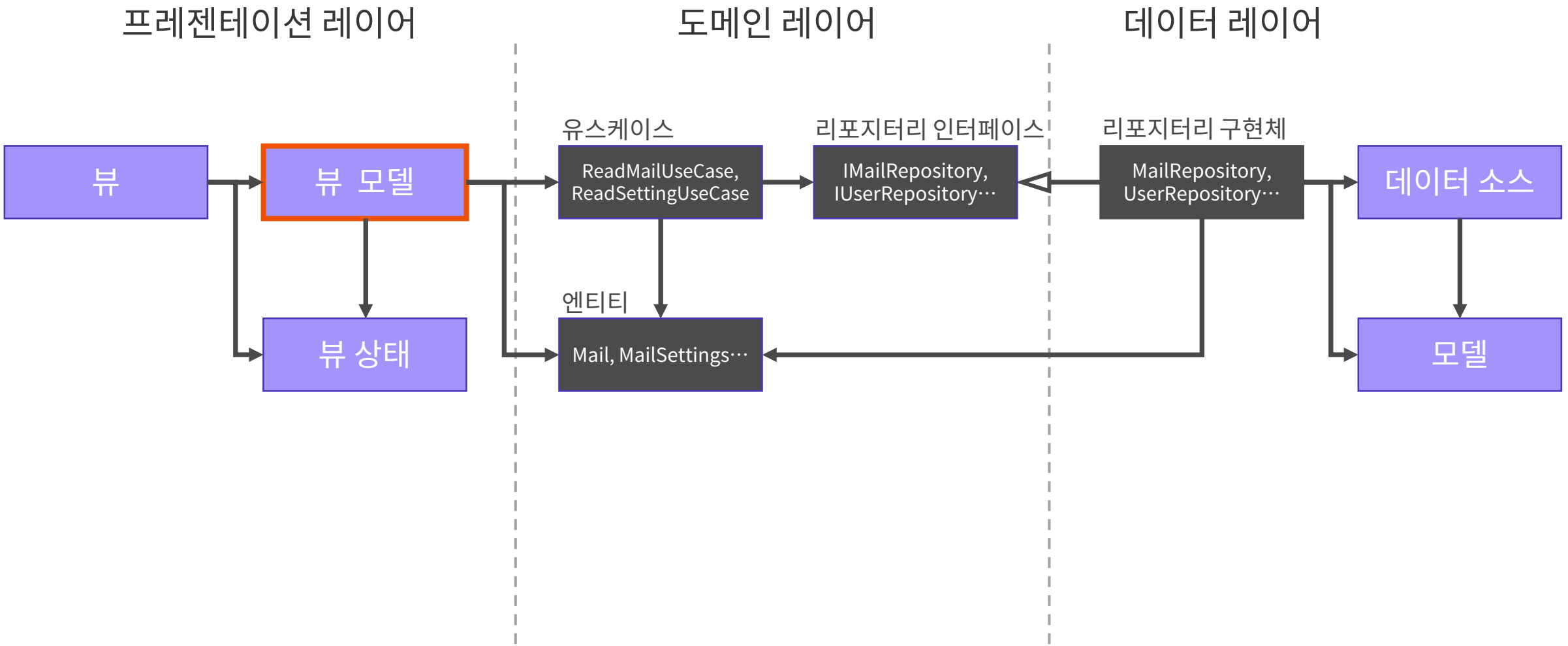

프레젠테이션 레이어
도메인 레이어
데이터 레이어
뷰
뷰 모델
뷰 상태
유스케이스
ReadMailUseCase,
ReadSettingUseCase
엔티티
Mail, MailSettings…
리포지터리 인터페이스
IMailRepository,
IUserRepository…
리포지터리
구현체
데이터 소스
모델

메일 리포지터리 구현(Pseudo Code)

NHN FORWARD

```
class MailRepository implements IMailRepository {
   MailRemoteDataSource mailRemoteDataSource

   public Mail getMail() {
      // 데이터 소스에서 메일 정보 조회
      MailApiModel mailApiModel = mailRemoteDataSource.fetchMail()

      // MailApiModel을 Mail 엔티티로 변환
      Mail mail = mapToEntity(mailApiModel)

      // Mail 엔티티를 리턴
      return mail
   }
}
```

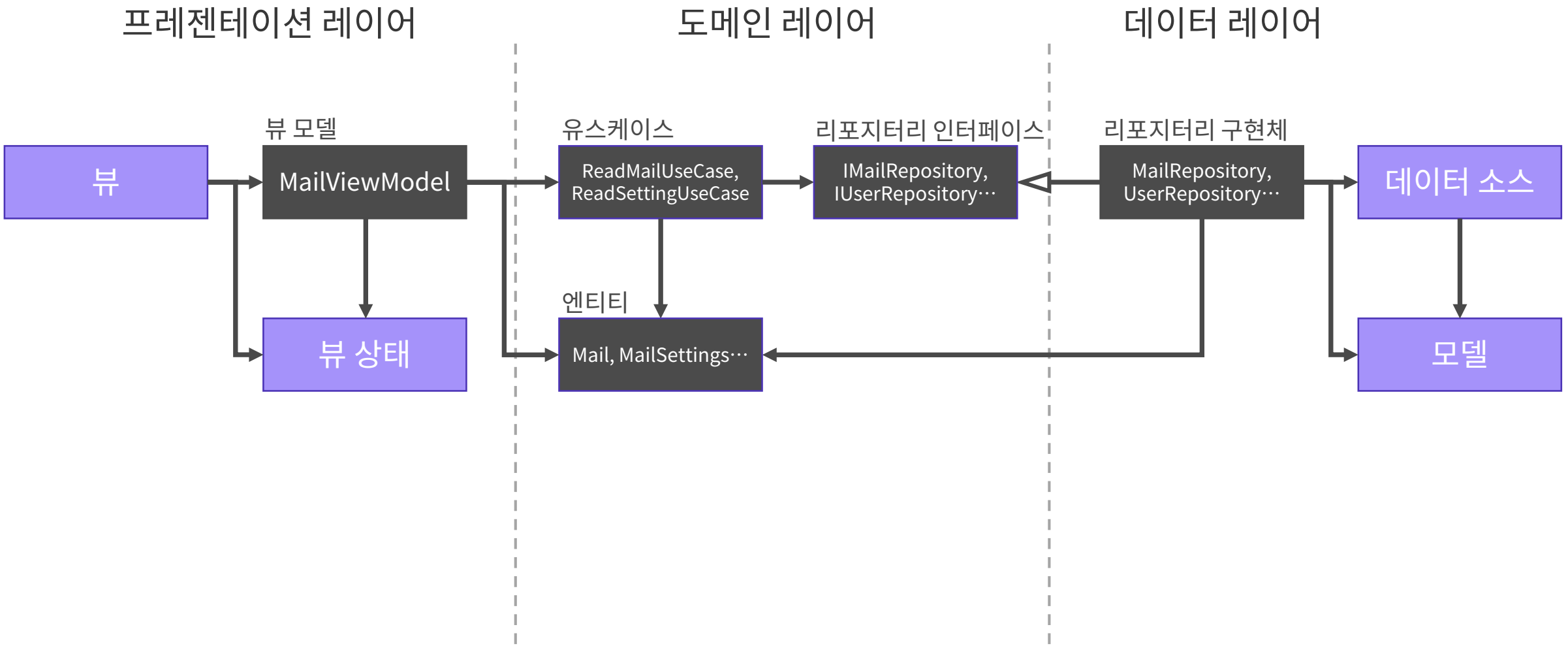
뷰 모델 리팩터링


프레젠테이션 레이어
도메인 레이어
데이터 레이어
뷰
뷰 모델
뷰 상태
유스케이스
ReadMailUseCase,
ReadSettingUseCase
엔티티
Mail, MailSettings…
리포지터리 인터페이스
IMailRepository,
IUserRepository…
리포지터리 구현체
MailRepository,
UserRepository…
데이터 소스
모델

뷰 모델 리팩터링(Pseudo Code)


```
class MailViewModel {
    ReadMailUseCase readMailUseCase
    ReadSettingUseCase readSettingUseCase

    public ViewState createViewState() {
        ViewState mailViewState
        // 번역 기능 활성화 여부 업데이트
        MailSettings settings = readSettingUseCase.getMailSettings()
        mailViewState.isTranslatorEnabled = settings.isTranslatorEnabled()
        // 메일 정보 업데이트
        Mail mail = readMailUseCase.getMail()
        mailViewState.subject = mail.getSubject()
        mailViewState.body = mail.getBody()
        …

        return mailViewState
    }
}
```

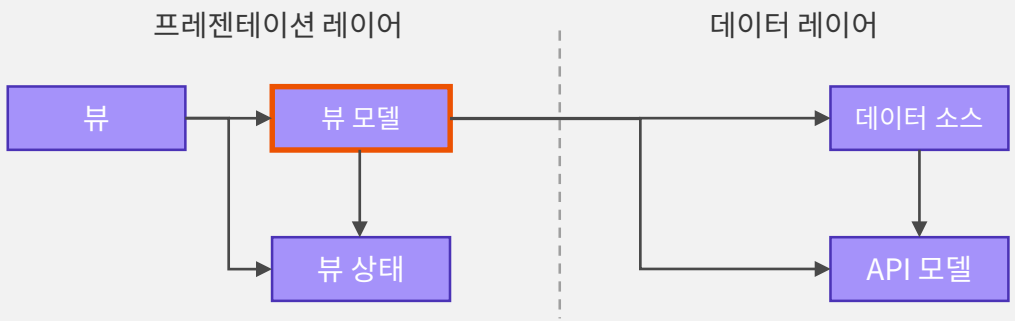
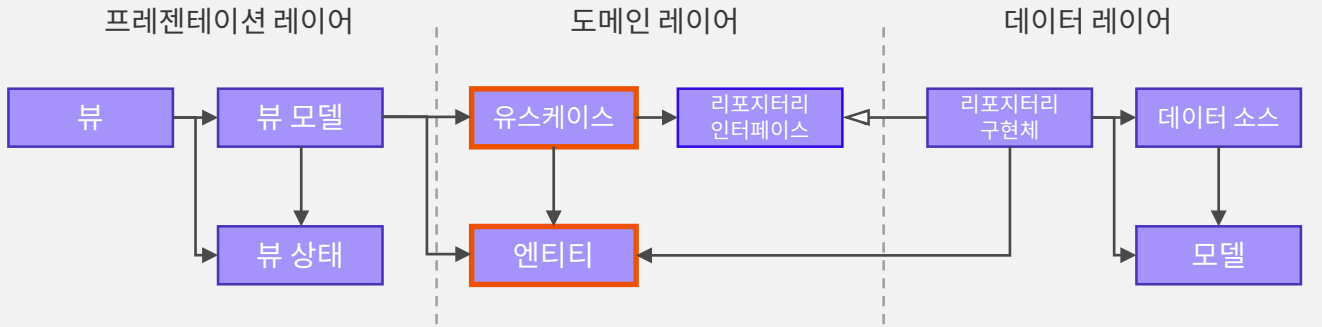
구현 완료


프레젠테이션 레이어
도메인 레이어
데이터 레이어
뷰 모델
유스케이스
리포지터리 인터페이스
리포지터리 구현체
뷰
MailViewModel
ReadMailUseCase,
ReadSettingUseCase
IMailRepository,
IUserRepository…
MailRepository,
UserRepository…
데이터 소스
엔티티
뷰 상태
Mail, MailSettings…
모델

클린 아키텍처 도입 후 이점

NHN FORWARD

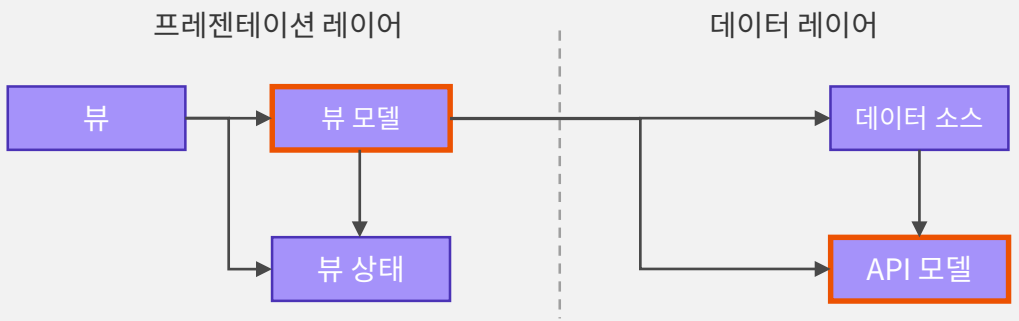
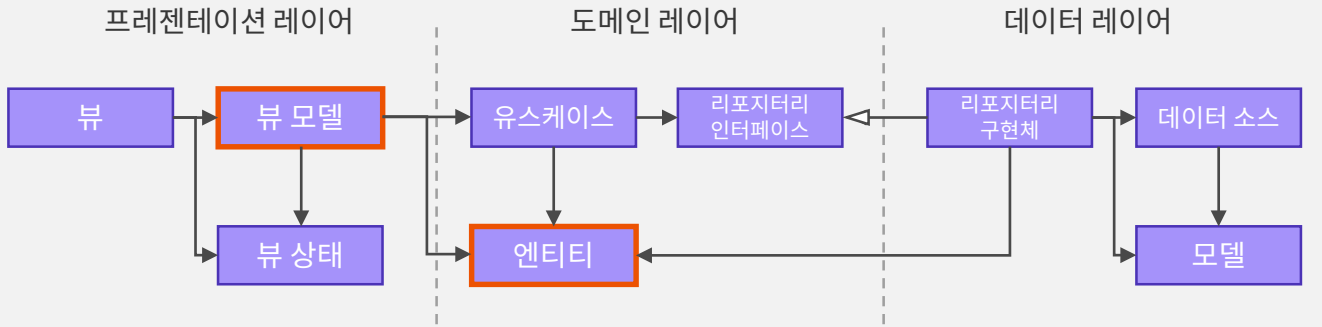
- 뷰 모델에서 비즈니스 로직 분리

메일 읽기 화면 변경 전	메일 읽기 화면 변경 후
	

클린 아키텍처 도입 후 이점

NHN FORWARD

- 외부 변화에 견고한 프레젠테이션 레이어

메일 읽기 화면 변경 전	메일 읽기 화면 변경 후
	

현재 두레이의 모습

두레이 서비스

NHN FORWARD


프로젝트


메신저


홈/게시판


메일


화상 회의


근무 관리


드라이브


위키

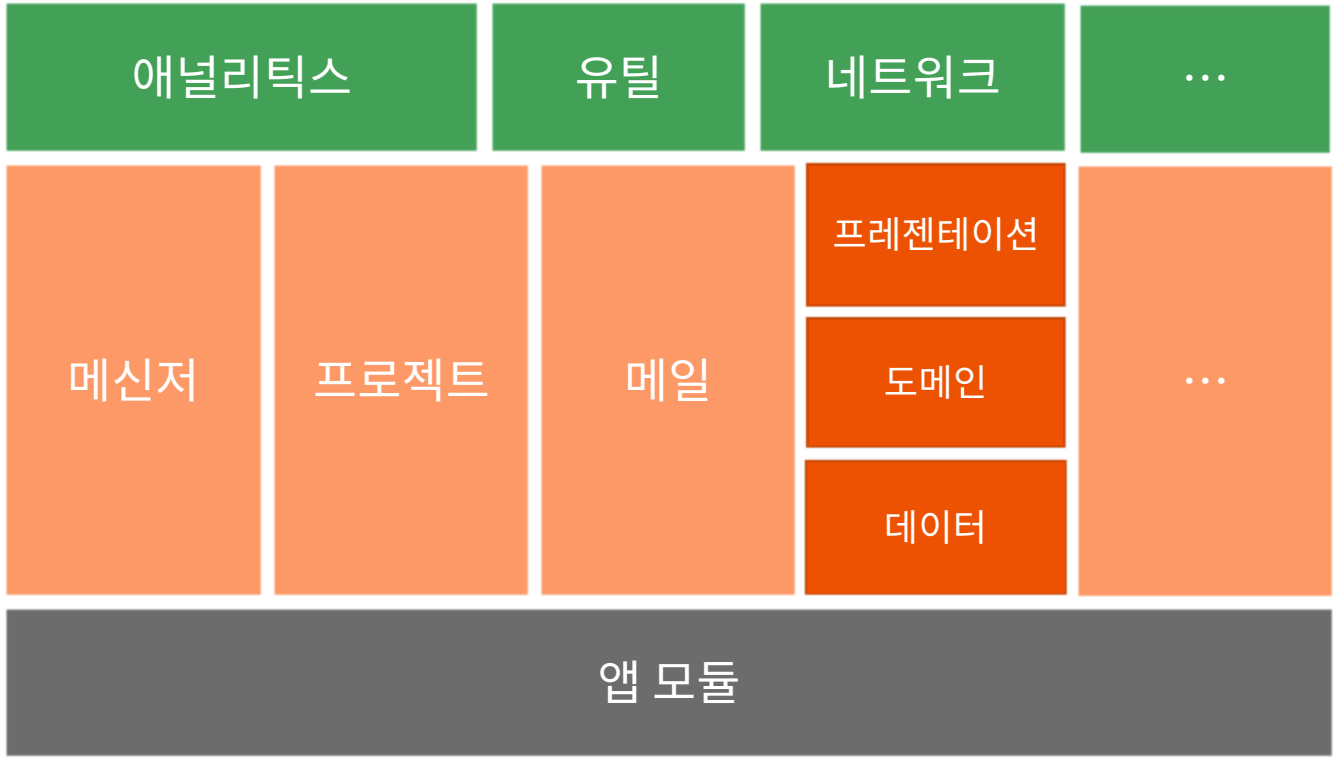

결재


캘린더


주소록


자원 예약

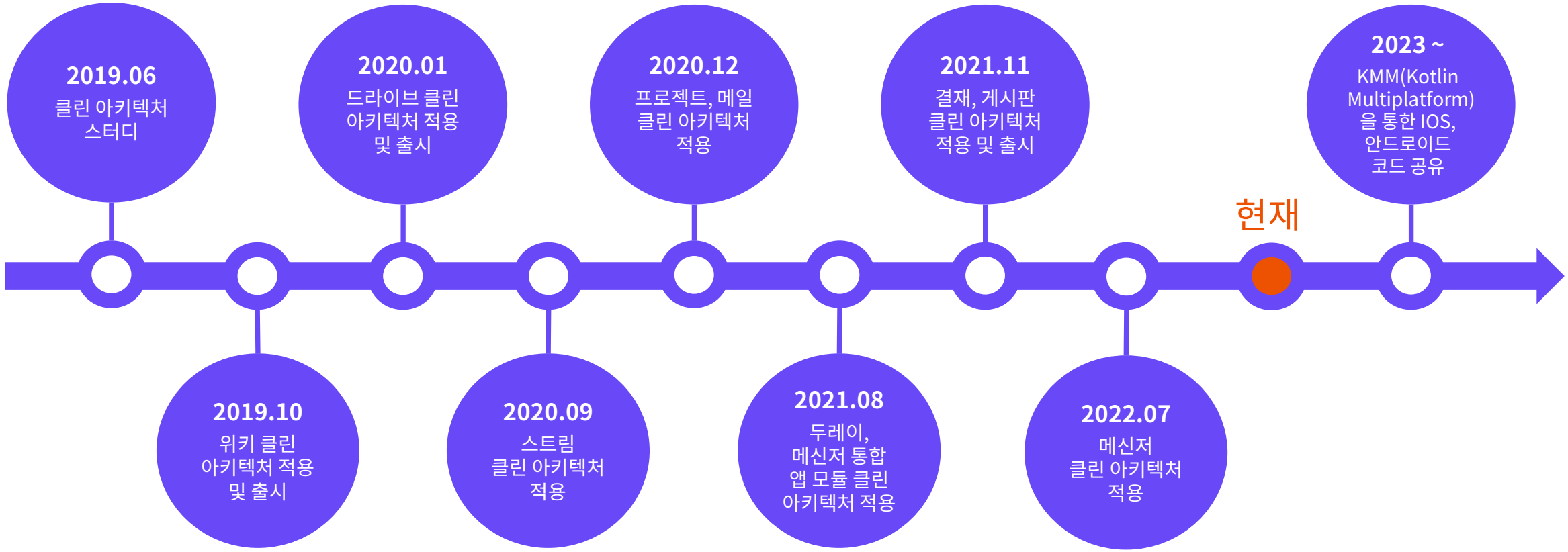
두레이 모듈 구조



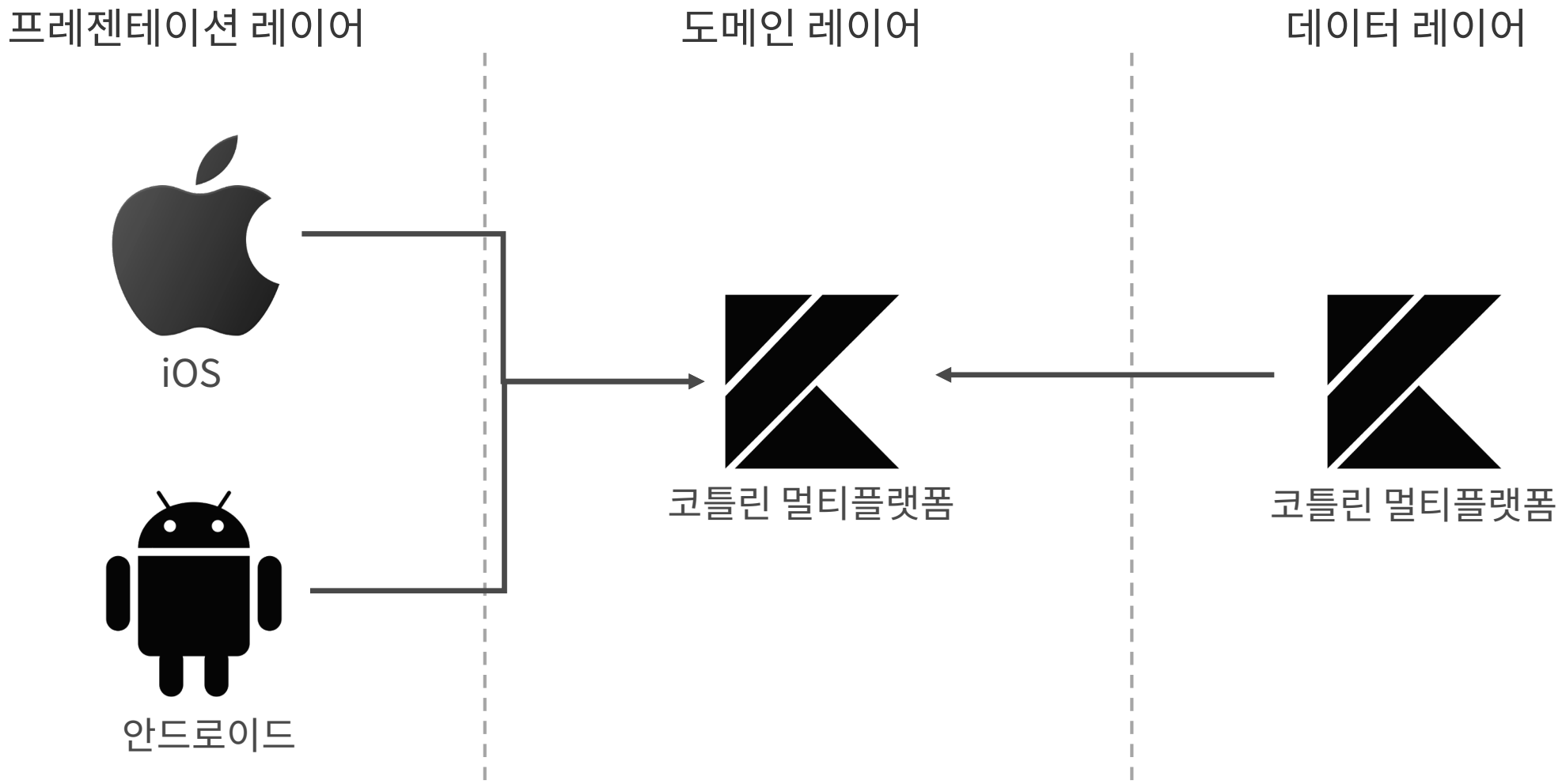
앞으로의 계획

클린 아키텍처 개발 히스토리

NHN FORWARD



코틀린 멀티플랫폼 도입



Q & A

NHN FORWARD

고맙습니다.