**이룸오더**

**high-level Architecture**

**UML Diagrams**

<팀명: 이룸핑>

<팀원: 2022920028 박수빈

2022920050 이희진

2022920062 최진영

2022920040 양나슬

2022920016 김은지

2022920058 주영은>

변 경 이 력

|  |  |  |  |
| --- | --- | --- | --- |
| 버전 | 일자 | 변경 내역 | 작 성 자 |
| V1 | 2024-11-11 | 프론트엔드 엔티티 단위 클래스 다이어그램 추가 | 박수빈, 이희진, 최진영 |
| V1.1 | 2024-11-14 | 프론트엔드 패키지 단위 클래스 다이어그램 추가 | 박수빈, 이희진 |
| V1.2 | 2024-11-18 | 백엔드 클래스, 시퀀스 다이어그램 추가 및  설명 작성 | 주영은, 양나슬, 김은지 |
| V1.3 | 2024-11-18 | 프론트엔드 클린 아키텍처와 클래스 다이어그램 설명 작성 | 박수빈, 이희진 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1. High-level Architecture 4](#_Toc182840105)

1.1 Frontend

1.2 Backend

[2. Class Diagram 4](#_Toc182840106)

2.1 Frontend

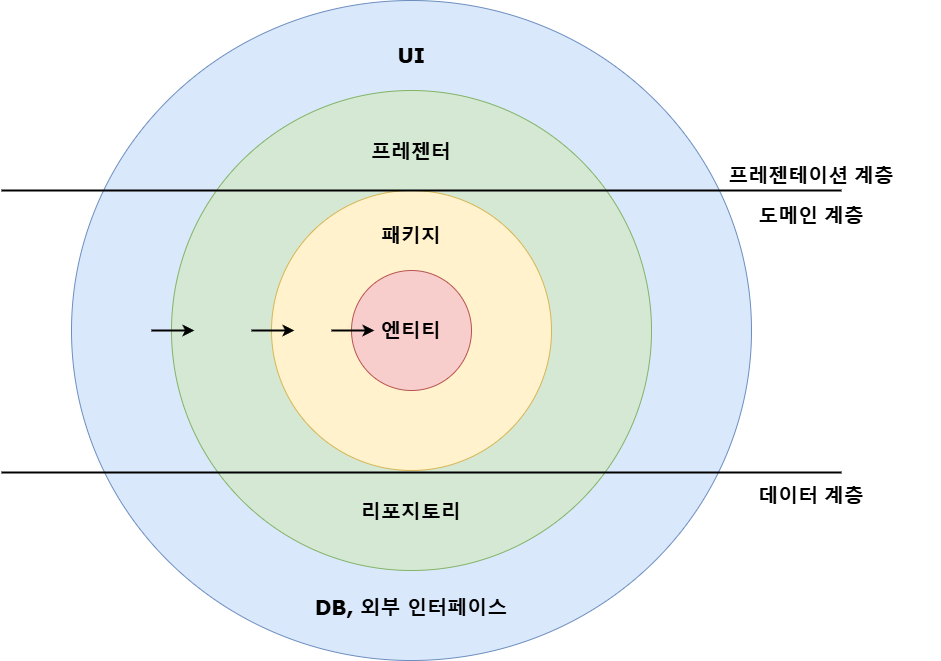
2.2 Backend

[3. Sequence Diagram 26](#_Toc182840107)

[4. style guide 29](#_Toc182840108)

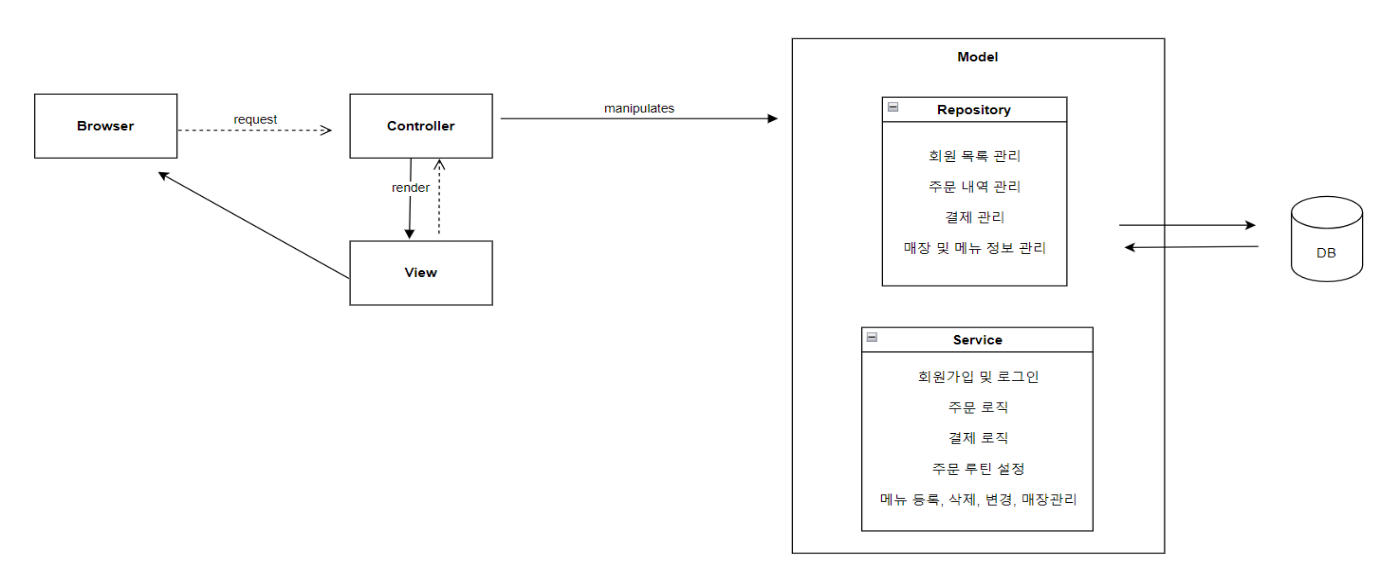
# High-level Architecture

1.1 Frontend - Clean architecture



프론트엔드는 각 계층 간의 독립성을 유지하기 위해서 clean 구조를 사용하였다. 각 계층은 엔티티, 유즈케이스, 인터페이스 어뎁터, 프레임워크&외부도구로 총 4계층으로 이루어져있다. 유즈케이스 계층은 패키지 단위로 구성되어 엔티티와 상호작용하며 요구 사항을 충족한다. 의존성 규칙에 의해 의존성은 항상 외부에서 내부로 흐르기 때문에 안쪽 계층은 바깥쪽 계층에 의존하지 않는다. 따라서 각 UI, DB를 독립적으로 구현 가능하다.

1.2 Backend - MVC architecture

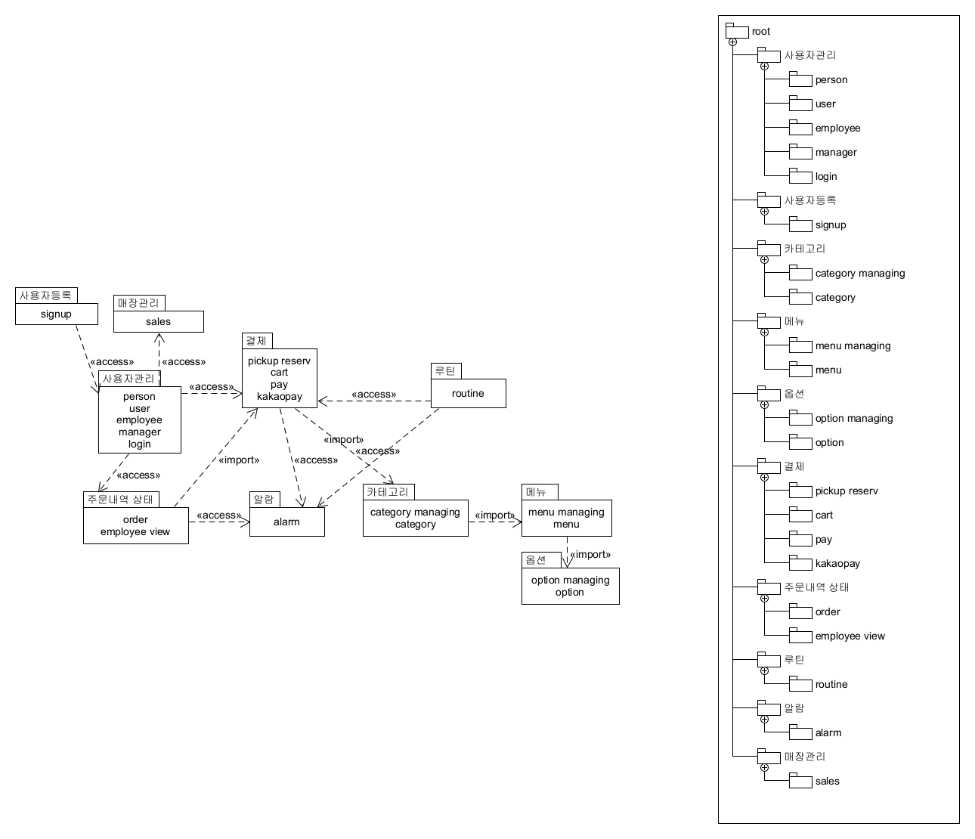


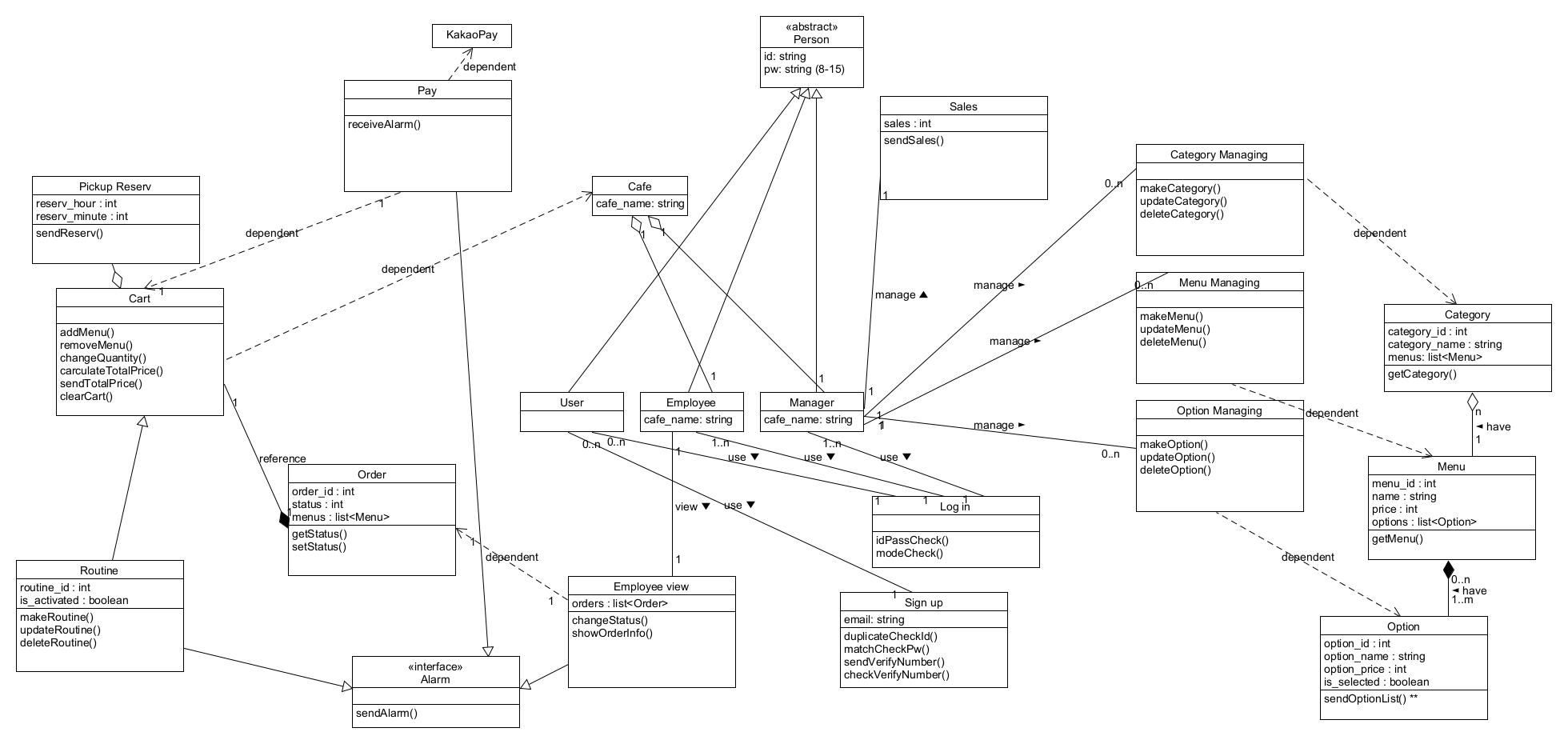
백엔드는 MVC 구조를 사용하여 구현하였다. Browser에서 Controller로 요청을 보내고, Controller는 browser, Model과 상호작용하여 데이터를 조작한다. Repository에서 DB의 데이터 조회, 저장, 수정 작업을 관리한다. Service에서 비즈니스 로직을 담당하여 처리한다.

MVC 구조를 사용하여 각 구성 요소 간의 책임을 명확히 나누어 코드를 모듈화하고, 애플리케이션의 유지보수성과 확장성을 향상시킨다.

# Class Diagram

* 1. Frontend



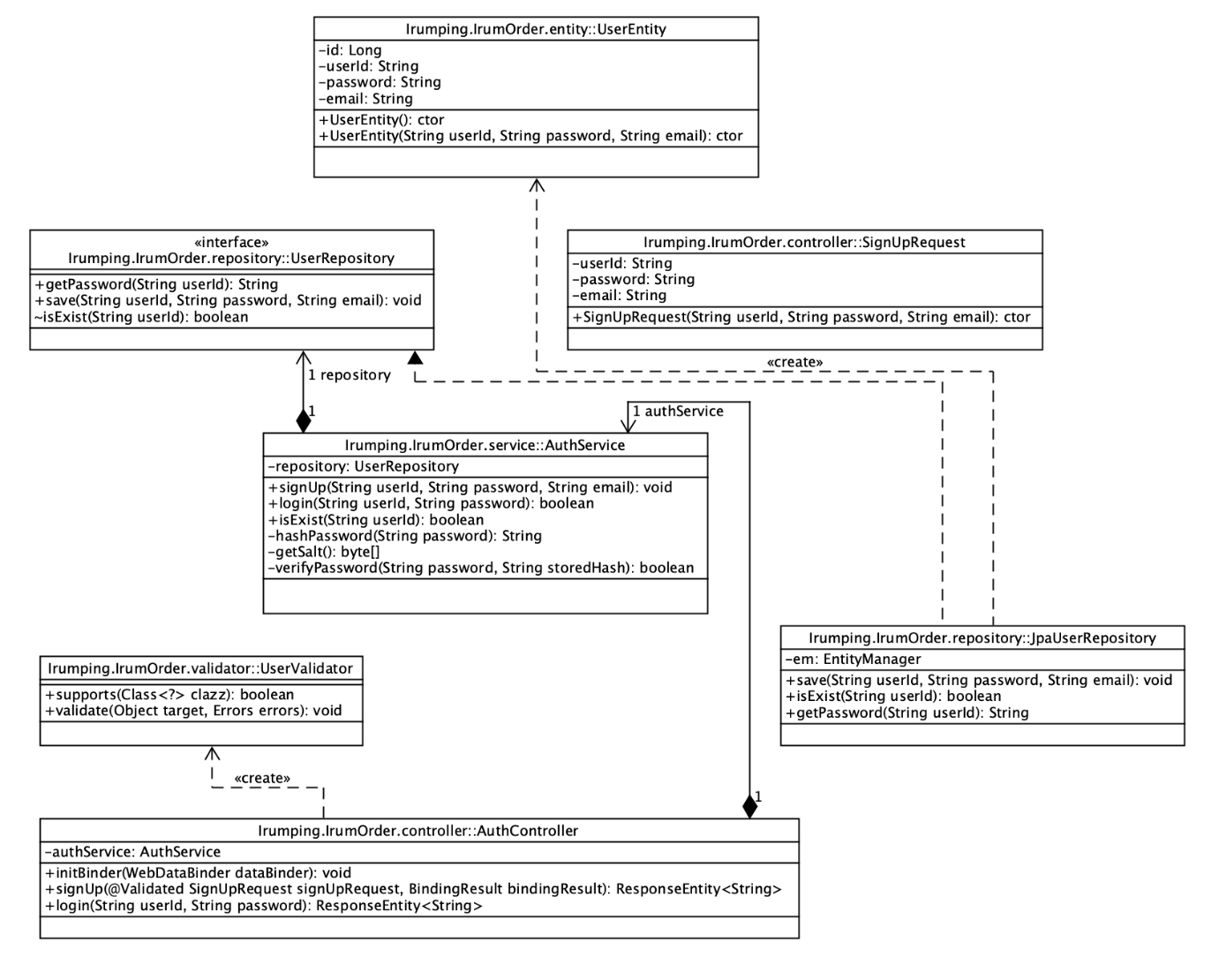


모듈 단위로 엔티티를 패키지로 묶어 관계를 표현했다.

사용자 관리, 사용자 등록, 카테고리, 메뉴, 옵션, 결제, 주문내역 상태, 루틴, 알람, 매장관리 기능별로 패키지를 만들었다.

* 1. Backend

2.2.1 class\_auth



SignUp

아이디, 비밀번호, 이메일을 입력하고 validator를 통해 검증을 거친 후 회원 정보를 DB에 저장하면 회원등록이 완료된다.

Login

회원, 직원, 매장관리자가 아이디와 비밀번호 입력을 통해 로그인한다.

2.2.2 class\_orderPay

텍스트, 도표, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명

OrderPay : 주문과 결제기능을 수행한다.

Order

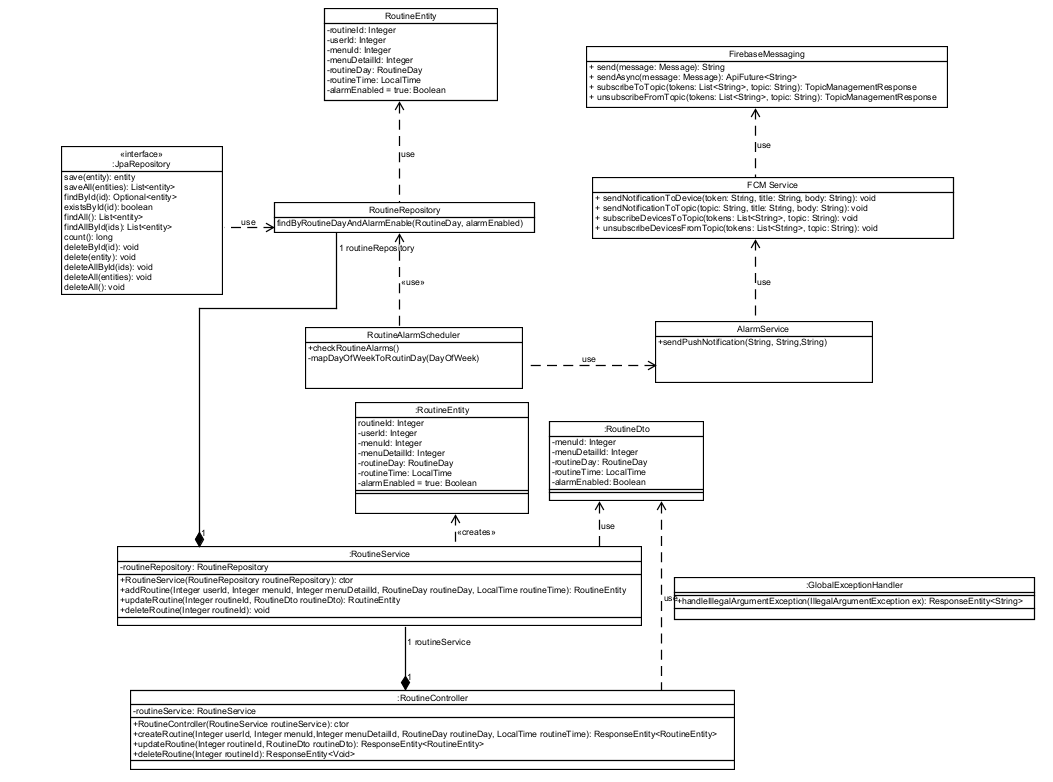
OrderRequestDto에서 받아온 정보들을 OrderEntity, OrderMenuEntity, MenuDetailEntity에 각각 저장해 db에 저장하고 OrderResponseDto를 반환함

Pay

PayController의 payReady 메서드에서 결제를 진행하기 위한 정보를 받고, 이를 PayService의 payReady 메서드로 전송한다. PayService에서 카카오페이 결제 url을 받아 클라이언트에 리턴한다.

클라이언트가 리턴받은 링크에서 결제를 완료한 후 얻은 pg\_token과 tid를 이용하여 결제확인을 수행한다(PayController의 payApprove 메서드, PayService의 payApprove 메서드드). 결제확인이 완료되면 DB에 결제승인 시각을 작성하고(payRepository 사용) 클라이언트에 결제승인 정보를 전송한다.

2.2.3 class\_routineAndAlam



Routine설정 및 Routine 알람기능

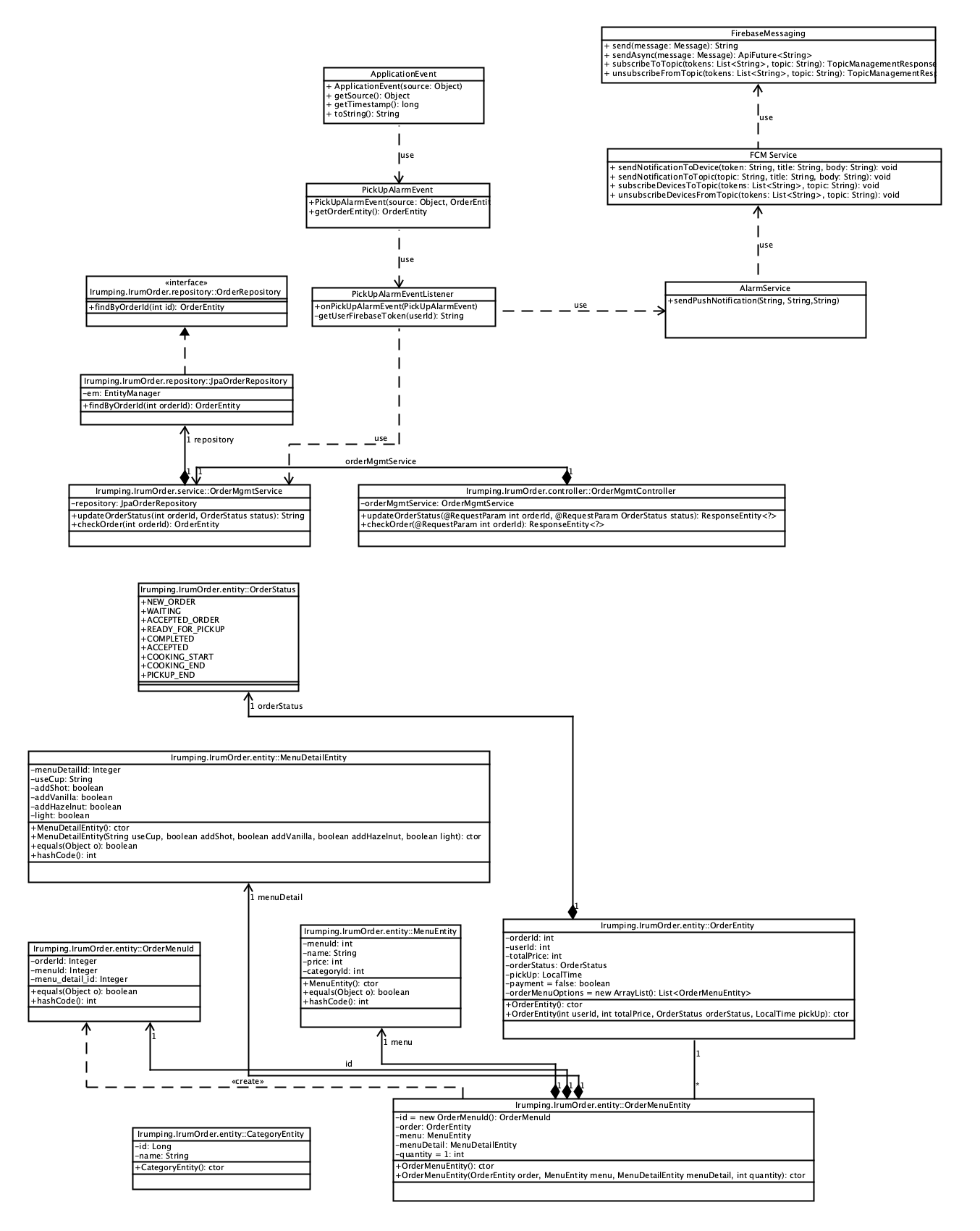
RoutineController로 루틴 추가, 수정, 삭제 요청이 들어온다.

이후 RoutineService 클래스에서 routineRepository를 통해 루틴을 DB에 저장/수정/삭제한다.

Routine Alarm

Scheduler를 이용해 10분마다 디비를 조회하여 픽업 예약시간 2시간전에 user들에게 루틴에 따라 Firebase를 이용하여 주문예약 팝업 알람을 보낸다.

2.2.4 class\_pickUpAlarm+orderManagement



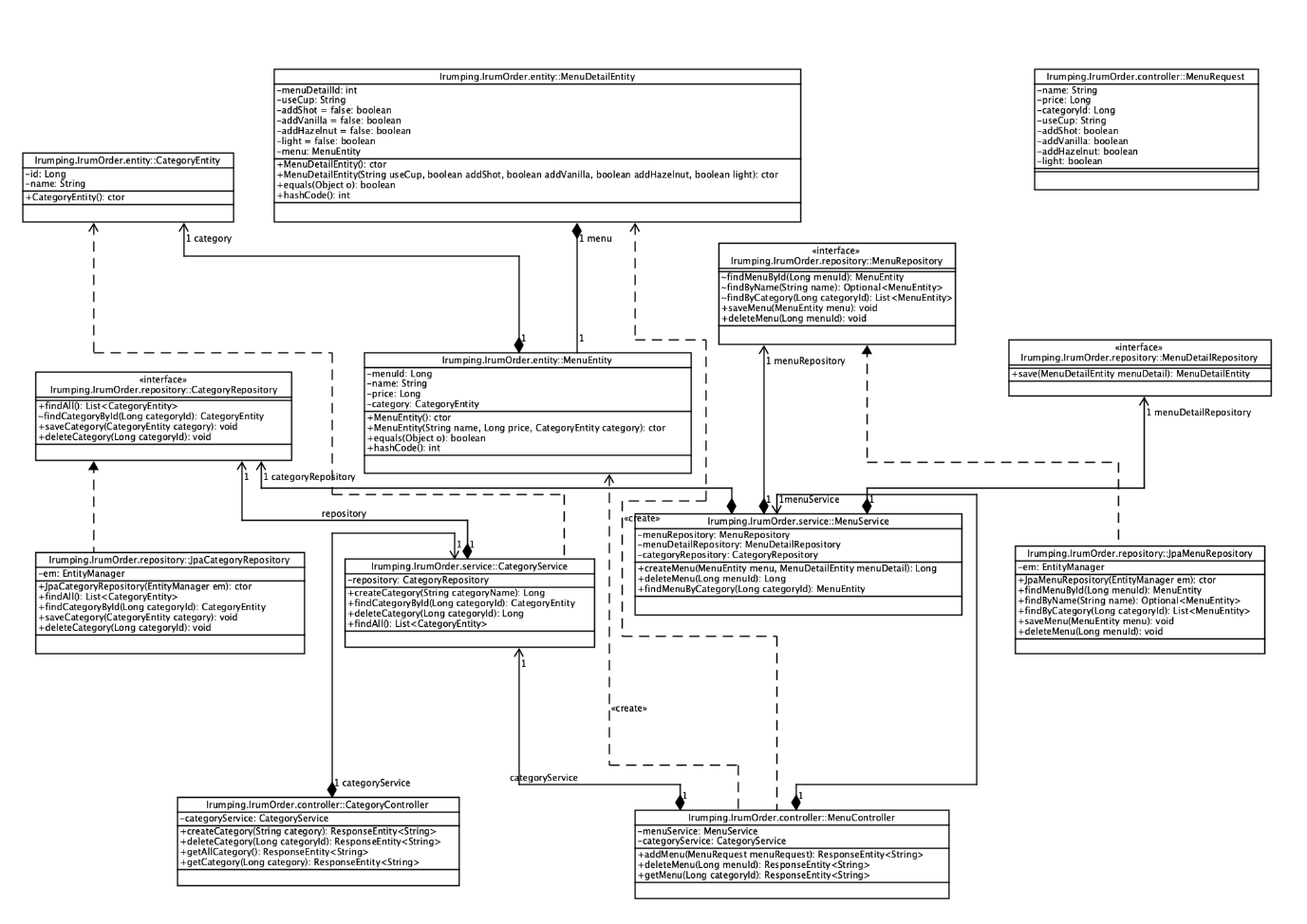
Order 상태 관리

직원이 상태 변경을 요청하면 DB에 변경된 상태를 저장한다.

Pick Up Alarm

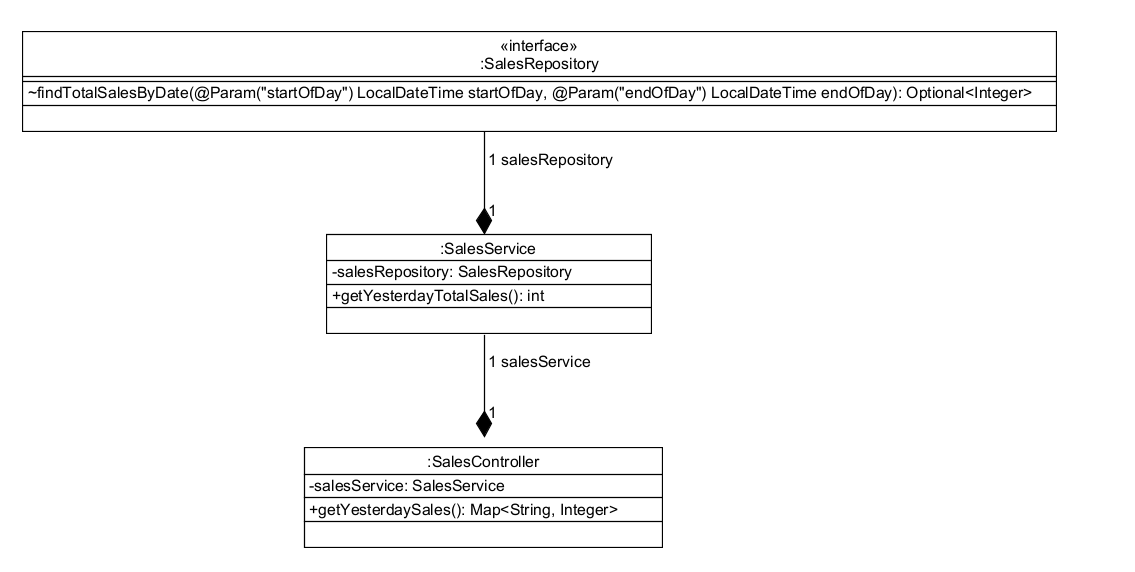
order가 PICKUP\_END 상태가 되면 회원에게 알림을 보낸다.

2.2.5 class\_menu



매장 관리자는 카테고리를 등록, 삭제할 수 있고, 각 카테고리에 대해서 메뉴를 등록, 삭제할 수 있다.

2.2.6 class\_salesCheck



SalesController로 작일 매출정보 열람 요청이 들어오면 SalesService의 getYesterdayTotalSales를 호출하고, salesRepository의 findTotalSalesByDate

메서드를 통해 작일의 총매출 정보를 DB에서 가져온다. 이후 정보를 클라이언트에 전송한다.

# Sequence Diagram

3.1 sequence\_auth

텍스트, 도표, 평행, 번호이(가) 표시된 사진

자동 생성된 설명

3.2 sequence\_orderPay

텍스트, 도표, 평행, 평면도이(가) 표시된 사진

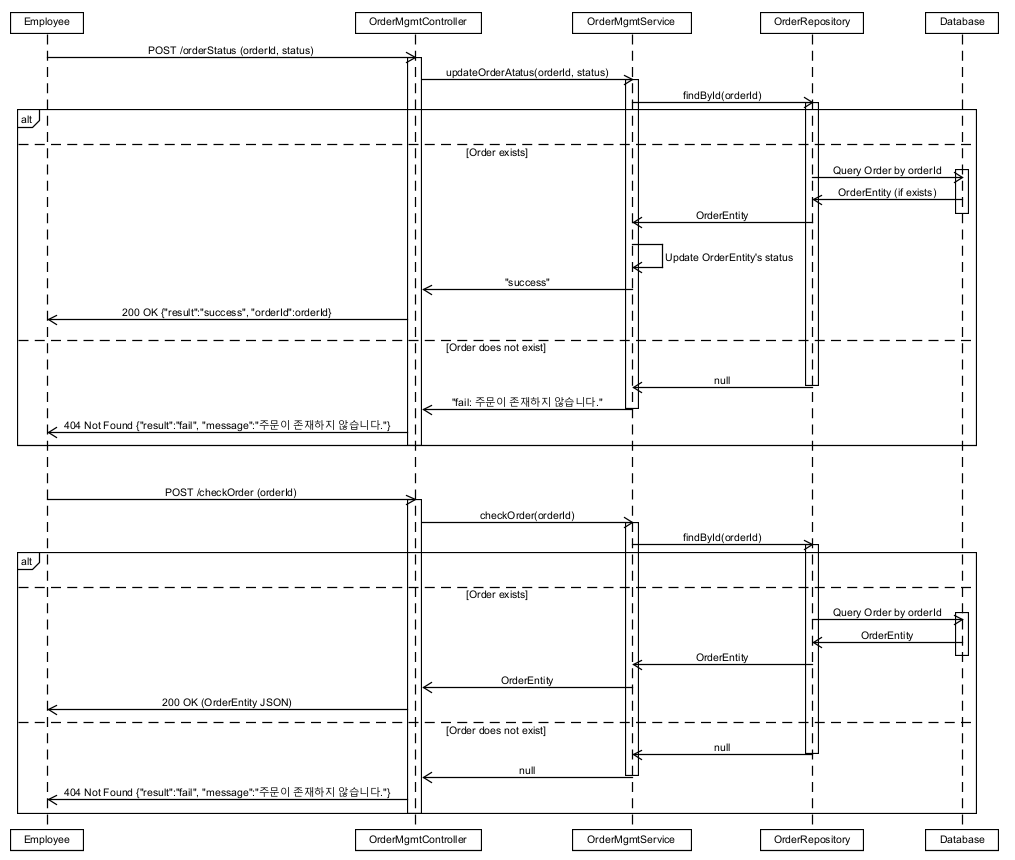
자동 생성된 설명

3.3 sequence\_routineAndAlarm

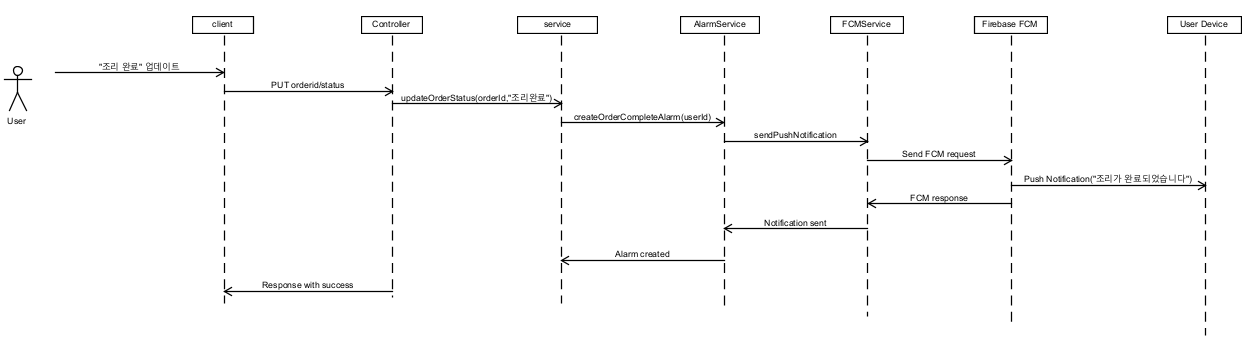
도표, 텍스트, 기술 도면, 평면도이(가) 표시된 사진

자동 생성된 설명

3.4 sequence\_orderManagement



3.5 sequence\_pickUpAlarm



3.6 sequence\_menu

텍스트, 도표, 평행, 라인이(가) 표시된 사진

자동 생성된 설명

3.7 sequence\_salesCheck

