BESTful

강경민 고유민 김경륜 김윤경 박경진 정원우

신한투자증권 프로디지털 아카데미 4기 미니미플젝

목차

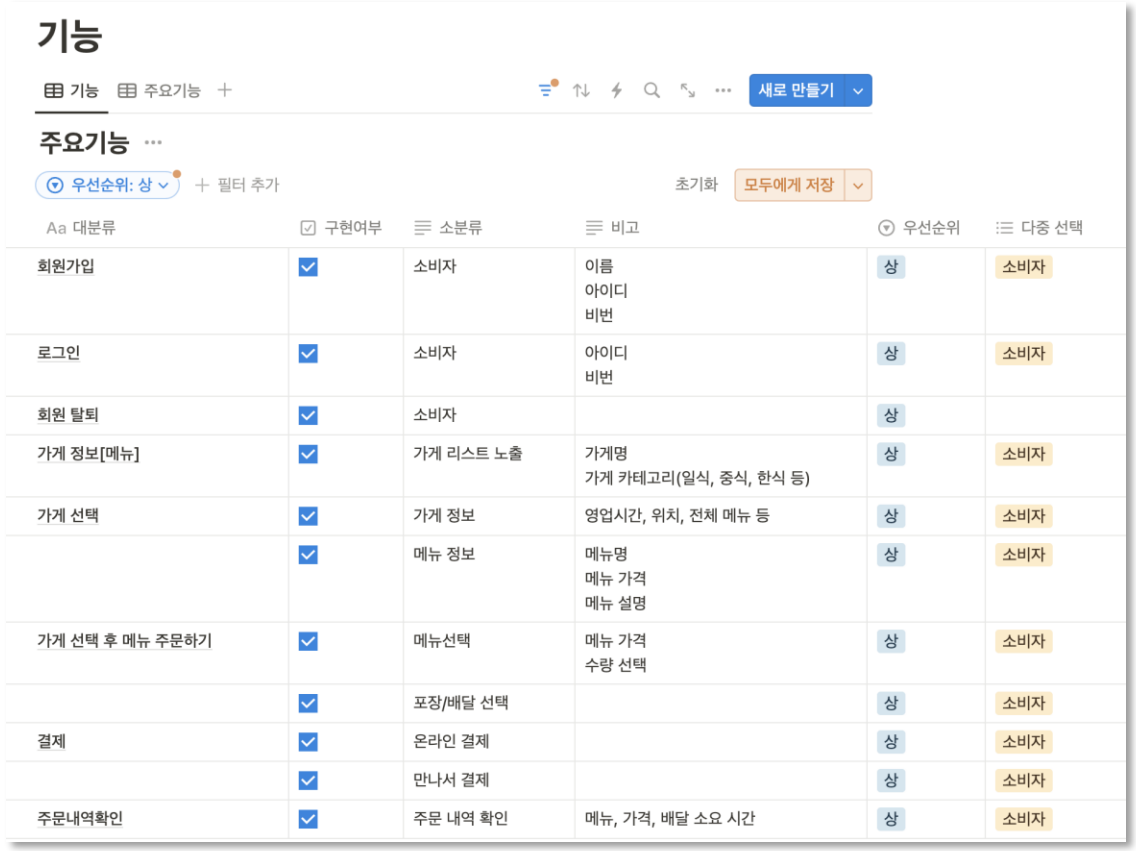
1. 주요 기능
2. 패키지 구조와 이유
3. 클래스 설계와 역할
4. 우리 팀이 객체 지향을 구현한 방법
   - 중요한 요소가 무엇이었는지
   - 어떻게 지향했는지
5. 프로젝트 중 이슈/고민
   - 선택/이유/해결 방안
6. 특히 신경 쓴 부분 또는 자랑거리

01. 주요기능

주요 기능

CLI를 활용한 배달의 민족 어플리케이션
(소비자)

- ✅ 로그인/ 로그아웃/ 회원가입/ 회원탈퇴
- ✅ 주문/ 주문 수정/ 장바구니/ 결제
- ✅ 주문내역/ 주문내역 상세
- ✅ 메뉴 네비게이트



기능

기능 | 주요기능 | + | 새로 만들기

주요기능 ···

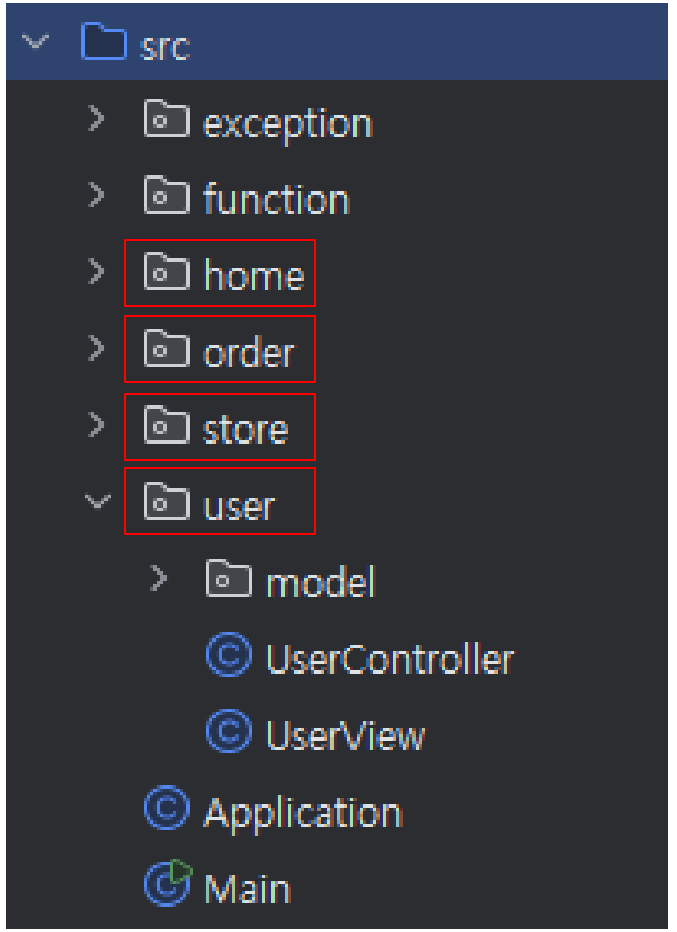
우선순위: 상 | + 필터 추가 | 초기화 | 모두에게 저장

Aa 대분류	구현여부	소분류	비고	우선순위	다중 선택
회원가입	☑	소비자	이름 아이디 비번	상	소비자
로그인	☑	소비자	아이디 비번	상	소비자
회원 탈퇴	☑	소비자		상	
가게 정보[메뉴]	☑	가게 리스트 노출	가게명 가게 카테고리(일식, 중식, 한식 등)	상	소비자
가게 선택	☑	가게 정보	영업시간, 위치, 전체 메뉴 등	상	소비자
	☑	메뉴 정보	메뉴명 메뉴 가격 메뉴 설명	상	소비자
가게 선택 후 메뉴 주문하기	☑	메뉴선택	메뉴 가격 수량 선택	상	소비자
	☑	포장/배달 선택		상	소비자
결제	☑	온라인 결제		상	소비자
	☑	만나서 결제		상	소비자
주문내역확인	☑	주문 내역 확인	메뉴, 가격, 배달 소요 시간	상	소비자

02. 패키지 구조와 이유

객체 지향 🌱

- ✅ 도메인 바탕 패키지 분리:
  home, user, store, order

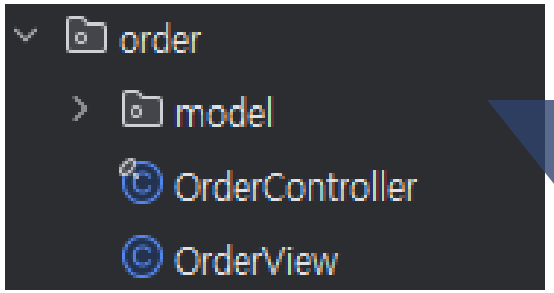


03. 클래스 설계와 역할

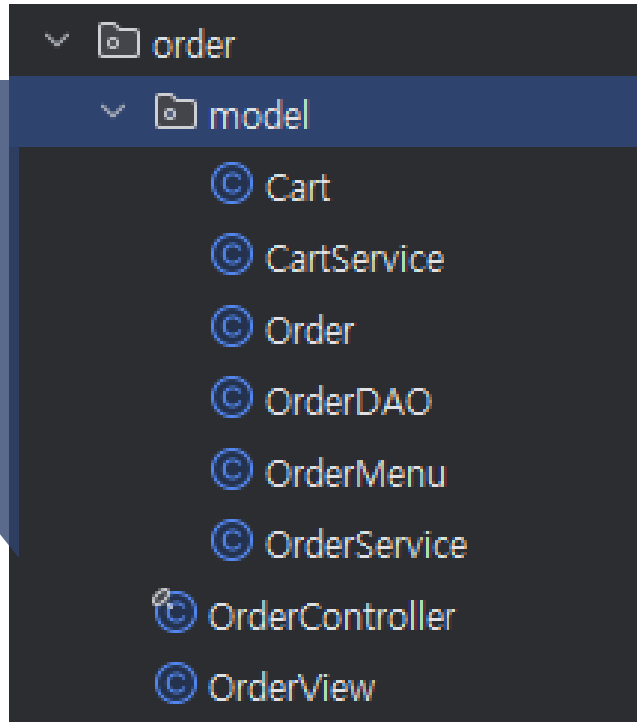
객체 지향🌱

- ✅ 클래스 역할 분리/구현

MVC



Model



Model

- Service
- DAO
- Entity

04. 객체 지향을 구현한 방법

객체 지향🌱

✅ **중요하게 생각하는 객체 지향 요소 정의**

➡️ 단일 책임

- **MVC 패턴**
  데이터 관리, 사용자 인터페이스, 비즈니스 로직에 대한 **책임을 독립적**으로 담당

- **캡슐화**
  데이터와 메소드를 하나의 단위로 묶어서 데이터 **무결성**을 보호하고,
  코드의 변경이 **외부**에 미치는 영향을 **최소화**

객체 지향 🌱

✅ 중요하게 생각하는 객체 지향 요소 구현 **MVC 패턴**

```
public class UserService {  3 usages   do-yoongyo2 +1 *
    private static final UserDAO userDAO = new UserDA

    public User login(String loginId, String wp) {  1
        User user = userDAO.select(loginId, wp);
        if (user == null) {
            throw new UserNotFoundException();
        }
        return user;
    }

    public void signup(String userName, String loginI
        if (userName.isEmpty() || loginId.isEmpty() |
            throw new SignupFailedException("Invalid
        }

        User newUser = new User( id: 0, loginId, userNa
        userDAO.insert(newUser);
    }
```

```
public class UserView {  2 usages   do-yoongyo2 +1
    private static final Scanner scanner = new Scanner(
    private static final StringBuilder view = new Strin

    public String[] loginView() {  1 usage   KimRiun +1
        view.setLength(0);
        view.append("************************** 로그인
            .append("            아이디와 비밀번호를 스페이
            .append("**********************************
        System.out.print(view);
        view.setLength(0);

        return getInput();
    }

    public void loginSuccessView(User user) { System.ou

    public void loginFailView() {  1 usage   KimRiun
        System.out.println("로그인에 실패하였습니다.");
        System.out.println("아이디와 비밀번호를 정확히 입력하
    }

    public void loginInputFailView() {  1 usage   KimRiun
        System.out.println("로그인에 실패하였습니다.");
        System.out.println("아이디와 비밀번호를 스페이스 간2
    }
```

```
public class UserController {  3 usages   KimRiun +3
    private static final UserService userService = new UserServic
    private static final UserView view = new UserView();  11 usages
    private final int FAIL = -1;  6 usages

    public String[] loginPage() {  1 usage   KimRiun
        return view.loginView();
    }

    public int login(String[] inputs) {  1 usage   KimRiun
        if (!validateInputCount(inputs, requiredInputCount: 2)) {
            loginInputFail();
            return FAIL;
        }

        try {
            String loginId = inputs[0];
            String wp = inputs[1];
            User user = userService.login(loginId, wp);
            loginSuccess(user);
            return user.getId();
        } catch (UserNotFoundException e) {
            loginFail();
            return FAIL;
```

주문 내역 어디에?

```java
public class User {  16 usages  KimRiun +1
    private int id;  3 usages
    private String loginId;  2 usages
    private String userName;  2 usages
    private String wp;  2 usages

    public User(int id, String loginId, String userName, String wp) {  1 usage
        this.id = id;
        this.loginId = loginId;
        this.userName = userName;
        this.wp = wp;
    }

    public int getId() { return id; }

    public void setId(int id) { this.id = id; }

    public String getLoginId() { return loginId; }

    public String getUserName() { return userName; }

    public String getWp() { return wp; }
}
```

```java
public class OrderDAO {  2 usages  YaRkyungmin
    private HashMap<Integer, Order> orderDb = new HashMap<>();  4 usages
    private int autoIncrementIndex = 1;  2 usages

    public void insert(Order order) {  1 usage  YaRkyungmin
        order.setId(autoIncrementIndex);
        orderDb.put(autoIncrementIndex++, order);
    }

    public Order[] fetchAll() { return orderDb.values().toArray(new Order[0]);

    public Order findById(int id) { return orderDb.get(id); }

    public int[] fetchAllId() {  1 usage  YaRkyungmin
```

객체 지향🌱

✅ 중요하게 생각하는 객체 지향 요소 구현

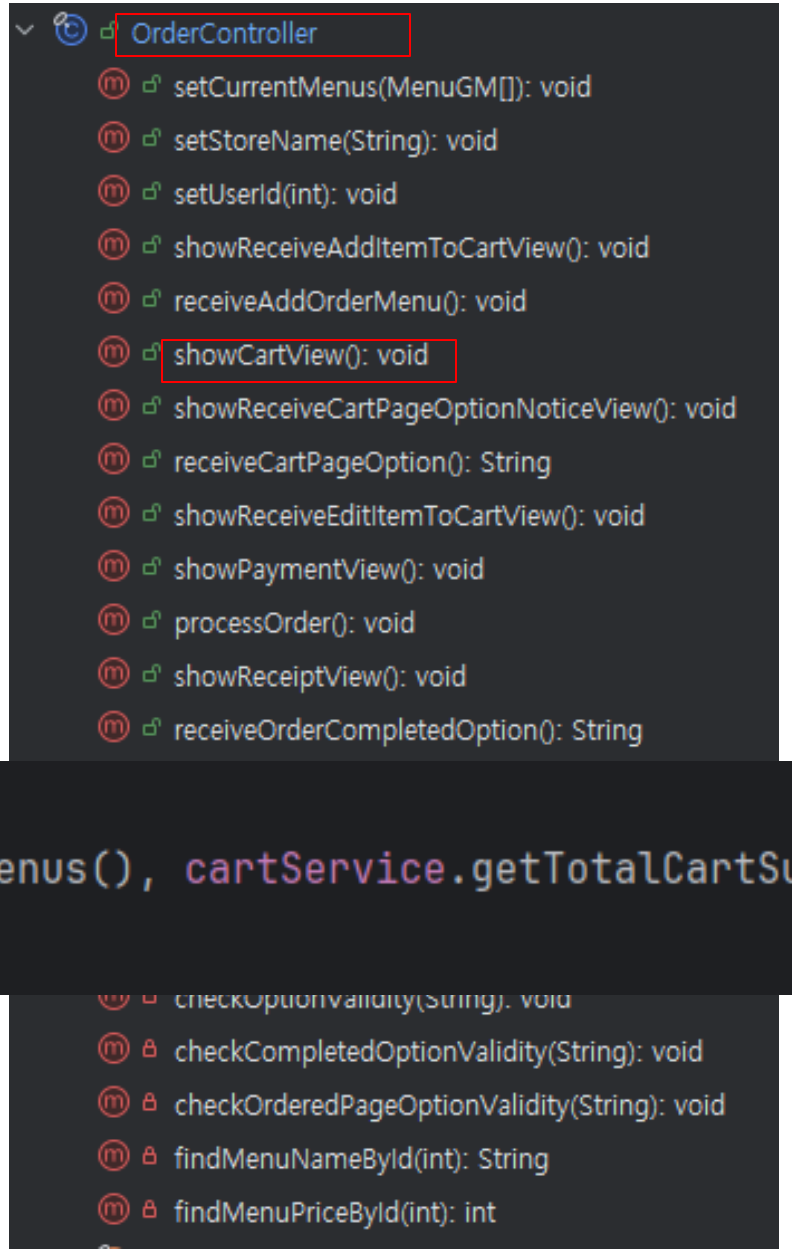
캡슐화

```
public class Store {  39 usages  잰진
    private String storeName;  2 usages
    private String storeCategory;  2 usages
    private String mainDish;  2 usages
    private String storeLocation;  2 usages
    private Menu menu;  3 usages
    private int averagePrice;  2 usages

    public Store(String storeName, String storeCategory,
        this.storeName = storeName;
        this.storeCategory = storeCategory;
        this.mainDish = mainDish;
        this.storeLocation = storeLocation;
        this.menu = new Menu(storeName);
    }

    public void setAveragePrice() { averagePrice =  menu.

    public String getStoreName() { return storeName; }

    public int getAveragePrice() { return averagePrice; }
```

클린 코드

✅ **하나의 메소드는 하나의 기능만 합니다.**

예) OrderController.java




```
public void showCartView() {  1 usage  YaRkyungmin
    orderView.showCartPage(cartService.getOrderMenus(), cartService.getTotalCartSum());
}
```

클린 코드 🫧

✅ **메소드 이름이 명확합니다.**

(메소드명을 읽었을 때, 코드를 이해하는 데 모호하지 않습니다)

예) UserDAO

```
public class UserDAO {  2 usages   do-yoongyo2 +1 *
    private HashMap<Integer, User> userDb = new HashMap<>();
    private int autoIncrementIndex = 1;  2 usages

    public void insert(User user) {...}

    public User select(String loginId, String wp) {...}

    public boolean update(User user) {...}

    public boolean delete(String loginId, String wp) {...}
}
```

클린 코드

✅ 불필요한 주석은 없습니다. (중복되는 주석 포함)

➡️ 주석 제거/ 유효 검사 메소드 분리/ 매직넘버 상수화/ null 반환 지양

```java
public class UserController {  3 usages  do-yoongyo2 +3 *
    private static final UserService userService = new UserService();
    private static final UserView view = new UserView();  10 usages

    public int login(String[] inputs) {  1 usage  new *
        // 입력 개수 예외처리
        if (inputs.length != 2) {
            return -1;
        }

        String loginId = inputs[0];
        String wp = inputs[1];
        User user = userService.login(loginId, wp);

        if (user == null) {
            loginFail();
            return -1;
        }

        loginSuccess(user);

        return user.getId();
    }
```

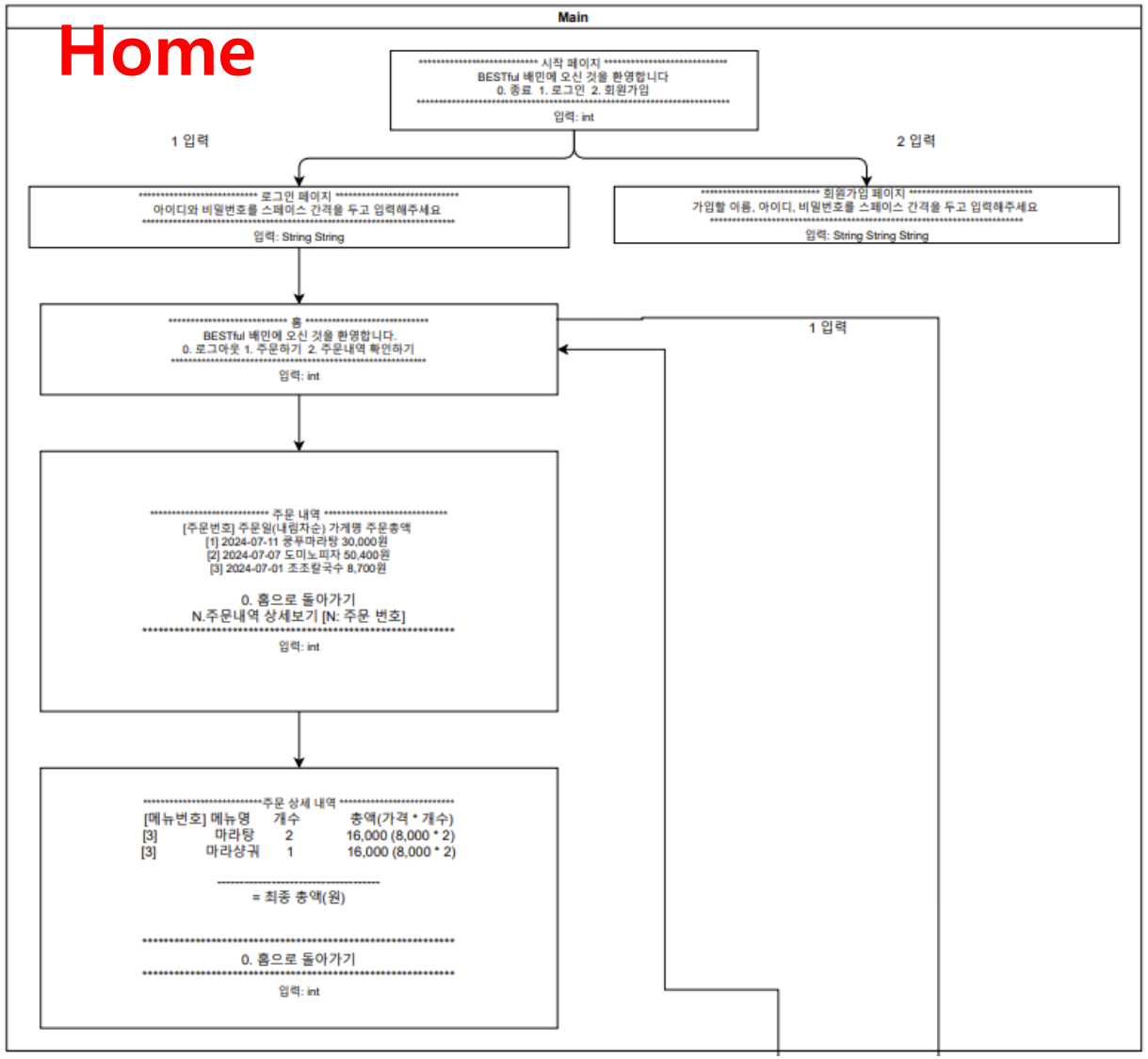
```java
public class UserController {  3 usages  do-yoongyo2 +3 *
    private static final UserService userService = new UserService();
    private static final UserView view = new UserView();  10 usages
    private final int FAIL = -1;  2 usages

    public int login(String[] inputs) {  1 usage  KimRiun
        if(!validateInputCount(inputs,  requiredInputCount: 2)) {
            loginInputFail();
            return FAIL;
        }

        try {
            String loginId = inputs[0];
            String wp = inputs[1];
            User user = userService.login(loginId, wp);
            loginSuccess(user);

            return user.getId();
        } catch (UserNotFoundException e) {
            loginFail();
            return FAIL;
        }
    }
```

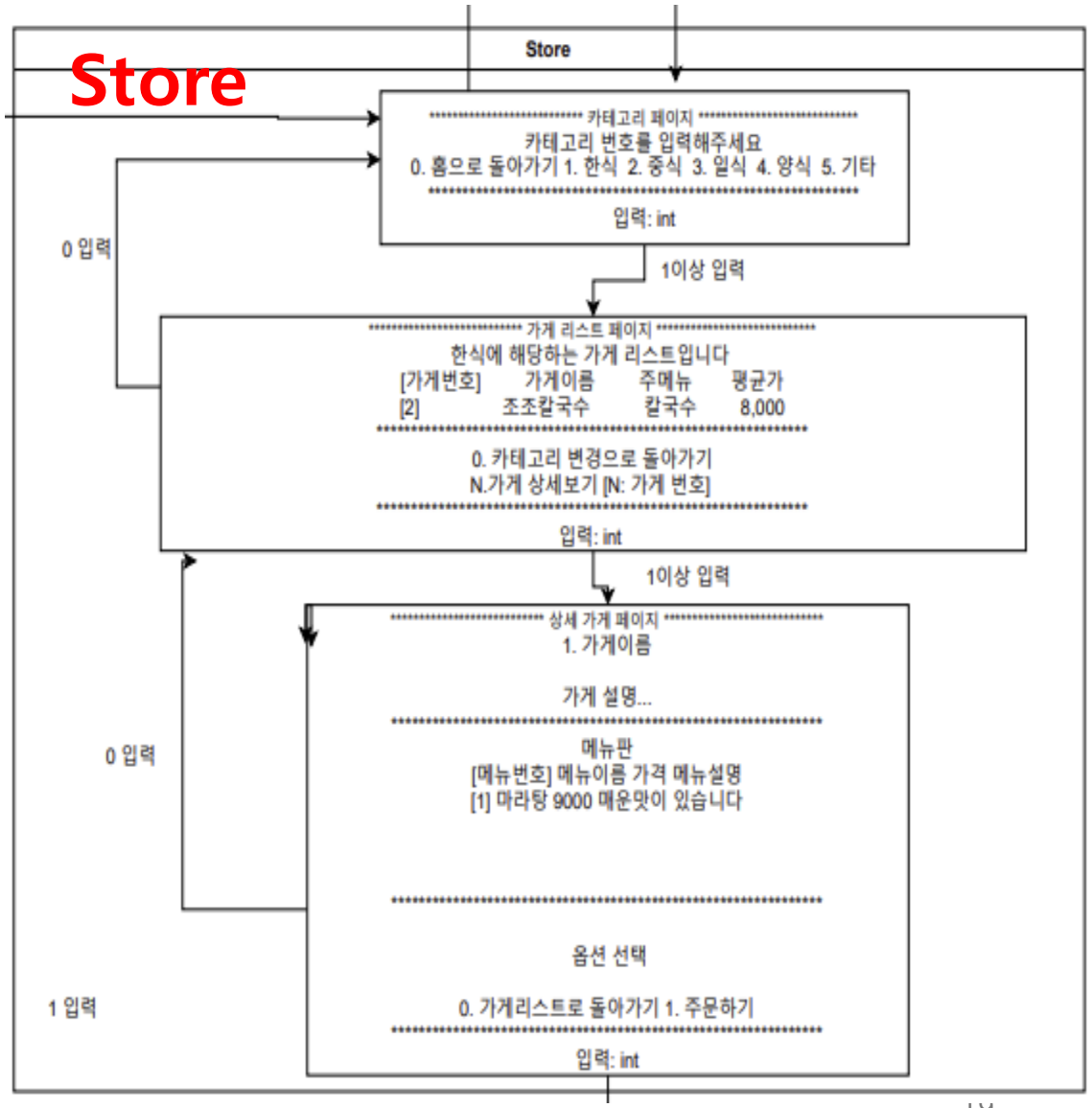
05. 이슈 및 해결방안

이슈/고민 📌

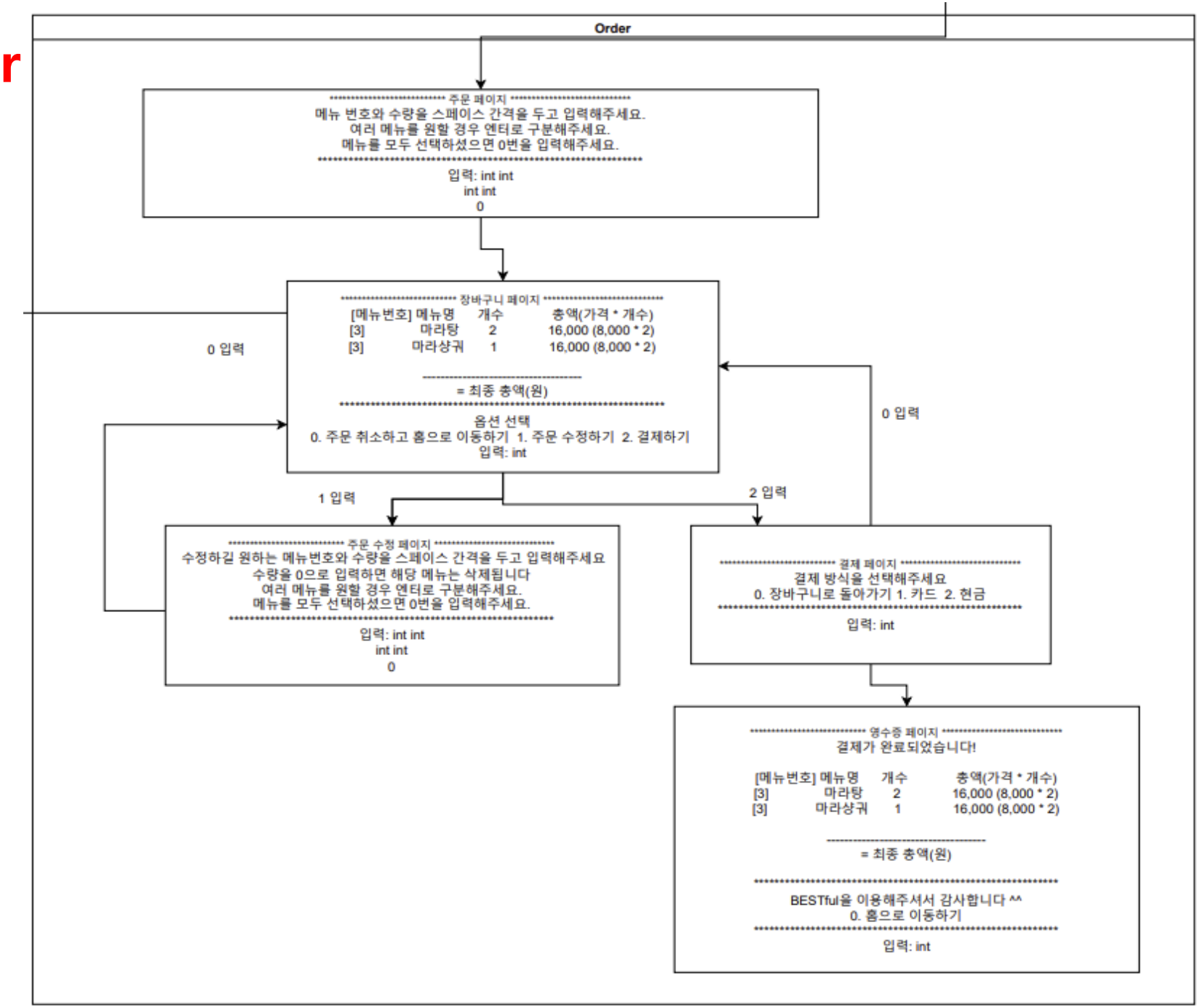
- 입출력 흐름과 패키지 구조
- 설계 vs 구현: 설계의 중요성
  - 설계의 범위와 투자 시간
- 기능 구현 범위 설정

입출력 흐름


Home
Main
시작 페이지
BESTful 배민에 오신 것을 환영합니다
0. 종료 1. 로그인 2. 회원가입
입력: int
1 입력
2 입력
로그인 페이지
아이디와 비밀번호를 스페이스 간격을 두고 입력해주세요
입력: String String
회원가입 페이지
가입할 이름, 아이디, 비밀번호를 스페이스 간격을 두고 입력해주세요
입력: String String String
홈
BESTful 배민에 오신 것을 환영합니다.
0. 로그아웃 1. 주문하기 2. 주문내역 확인하기
입력: int
1 입력
주문 내역
[주문번호] 주문일(내림차순) 가게명 주문총액
[1] 2024-07-11 궁루마라탕 30,000원
[2] 2024-07-07 도미노피자 50,400원
[3] 2024-07-01 조조칼국수 8,700원
0. 홈으로 돌아가기
N.주문내역 상세보기 [N: 주문 번호]
입력: int
주문 상세 내역
[메뉴번호] 메뉴명 개수 총액(가격 * 개수)
[3] 마라탕 2 16,000 (8,000 * 2)
[3] 마라샹궈 1 16,000 (8,000 * 2)
= 최종 총액(원)
0. 홈으로 돌아가기
입력: int


Store
Store
카테고리 페이지
카테고리 번호를 입력해주세요
0. 홈으로 돌아가기 1. 한식 2. 중식 3. 일식 4. 양식 5. 기타
입력: int
0 입력
1이상 입력
가게 리스트 페이지
한식에 해당하는 가게 리스트입니다
[가게번호] 가게이름 주메뉴 평균가
[2] 조조칼국수 칼국수 8,000
0. 카테고리 변경으로 돌아가기
N.가게 상세보기 [N: 가게 번호]
입력: int
1이상 입력
상세 가게 페이지
1. 가게이름
가게 설명...
메뉴판
[메뉴번호] 메뉴이름 가격 메뉴설명
[1] 마라탕 9000 매운맛이 있습니다
옵션 선택
0. 가게리스트로 돌아가기 1. 주문하기
입력: int
0 입력
1 입력

Order


Order
주문 페이지
메뉴 번호와 수량을 스페이스 간격을 두고 입력해주세요.
여러 메뉴를 원할 경우 엔터로 구분해주세요.
메뉴를 모두 선택하셨으면 0번을 입력해주세요.
입력: int int
int int
0
장바구니 페이지
[메뉴번호] 메뉴명 개수 총액(가격 * 개수)
[3] 마라탕 2 16,000 (8,000 * 2)
[3] 마라샹궈 1 16,000 (8,000 * 2)
= 최종 총액(원)
옵션 선택
0. 주문 취소하고 홈으로 이동하기 1. 주문 수정하기 2. 결제하기
입력: int
0 입력
1 입력
2 입력
0 입력
주문 수정 페이지
수정하길 원하는 메뉴번호와 수량을 스페이스 간격을 두고 입력해주세요
수량을 0으로 입력하면 해당 메뉴는 삭제됩니다
여러 메뉴를 원할 경우 엔터로 구분해주세요.
메뉴를 모두 선택하셨으면 0번을 입력해주세요.
입력: int int
int int
0
결제 페이지
결제 방식을 선택해주세요
0. 장바구니로 돌아가기 1. 카드 2. 현금
입력: int
영수증 페이지
결제가 완료되었습니다!
[메뉴번호] 메뉴명 개수 총액(가격 * 개수)
[3] 마라탕 2 16,000 (8,000 * 2)
[3] 마라샹궈 1 16,000 (8,000 * 2)
= 최종 총액(원)
BESTful을 이용해주셔서 감사합니다 ^^
0. 홈으로 이동하기
입력: int

패키지 구조 : 도메인별 vs MVC

도메인별

- 모듈별로 분업하여 통합하기 위해 도메인별 패키지 구조를 선택
- 도메인별 패키지 구조는 비즈니스 로직 중심의 개발
- 기능 추가 및 확장을 고려하여 채택

설계 vs 구현

설계

시간 고려/ 설계 범위

- 입출력 흐름
- 패키지 구조
- 인터페이스
  - 유효성 체크 개인 구현

- 원우 : 구현
  - 완벽한 설계는 못해도 명확한 설계는 필요하다 : 인터페이스(DTO 포함)
- 경민 : 설계
  - 분업에서 설계가 더 중요함
  - 구현단계에서 서로 생각하는 설계가 겹칠 수 있으니까
- 유민 : 설계
  - 동의!
- 경진 : 구현
  - 어떤 파라미터를 정해줄지나 데이터 어떻게 저장할지만 정해지면 그 다음부터는 구현이 더 중요하다
- 경륜 : 설계
  - 네이밍, 파라메터, 리턴타입 같은 인터페이스 설계 잘 해두고, 구현은 각자 하자
- 윤경 : 설계
  - 서로의 생각을 맞춰나가는게 설계라 생각
  - 개인의 뇌피셜로 구현하면 안된다

기능 구현 범위

가게주인, 배달원...

-> 우선순위

Aa 대분류	구현여부	소분류	비고	우선순위	다중 선택
회원가입	☐	가게주인		하	소비자
회원가입	☑	소비자	이름 아이디 비번	상	소비자
로그인	☐	가게주인		하	가게주인
로그인	☑	소비자	아이디 비번	상	소비자
	☐	배달기사		하	배달기사
	☐	관리		하	관리자
회원 탈퇴	☑	소비자		상	
가게 정보[메뉴]	☑	가게 리스트 노출	가게명 가게 카테고리(일식, 중식, 한식 등)	상	소비자
가게 선택	☑	가게 정보	영업시간, 위치, 전체 메뉴 등	상	소비자
	☑	메뉴 정보	메뉴명 메뉴 가격 메뉴 설명	상	소비자
가게 선택 후 메뉴 주문하기	☑	메뉴선택	메뉴 가격 수량 선택	상	소비자
	☑	포장/배달 선택		상	소비자

컨벤션

브랜치

`main`

`feat/user-login`

✅ main으로 바로 PR날리기

커밋 컨벤션

💡 커밋 메세지: **명사형** (이모지 제외)
ex) 로그인 구현(O)
로그인 구현함, 로그인 구현했음(X)

✨ feat: 새로운 기능 추가

🐛 fix: 버그 수정

⚡ update: Fix와 달리 원래 정상적으로 동작했지만 보완의 개념

♻️ refactor: 코드 리팩터링(기능 안바꿈)

💄 design: css 등 사용자 UI 디자인 변경

⚒️ chore: 빌드 업무 수정, 패키지 매니저 수정

📝 docs: .md, text 수정

💡 comment: 주석 추가 및 변경

🎨 style: 코드스타일에대한 변경

🔥 remove: 파일을 삭제하는 작업만 수행

🚚 rename: 파일 폴더명 수정 및 옮김

06. BESTfull 자랑거리

자랑거리 🔥

- 메뉴 카테고리
- 메뉴 담기 수정 가능
- 뒤로 돌아가기 기능
- 주문 내역 기록

Q&A

감사합니다 :)