
3차 발표 보고서

과 목 명	소프트웨어공학개론_02	
담 당 교 수	이 호 정	
학 과	컴퓨터공학과	
팀 명	트래블	
팀 원	조광호	2018112072
	이희준	2018112070
	이대엽	2019113577
	조영승	2019112074

목 차

0. 서론	3
- 팀원	3
1. 프로젝트 목표	3
2. 설계 및 구현 비교	4
3. 작동 사례	11
3. 테스트	20
- 예약	20
- 조회	21
- 환불	22
4. Github 주소	22

0. 서론

- 팀원

본 팀(트래블)의 인원 구성은 다음과 같다.

이름	학번	역할
조광호	2018112072	팀장, 백 엔드
이희준	2018112070	백 엔드, 프론트 엔드
이대엽	2019113577	백 엔드
조영승	2019112074	프론트 엔드

1. 프로젝트 목표

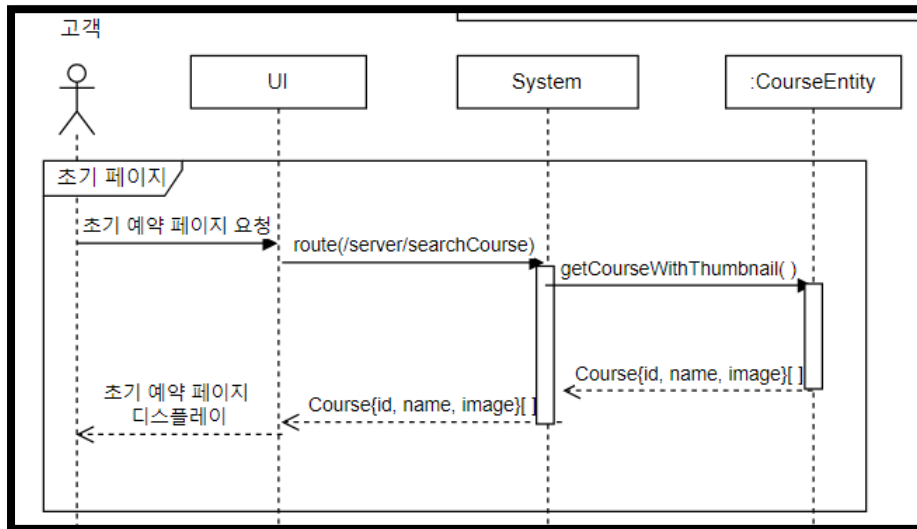
본 팀은 버스 관광 예약을 주제로 프로젝트를 진행했다. 버스를 통한 도시 관광은 현재 대표적인 관광 방식 중 하나로, 다른 관광 방식들에 비해 가격이 저렴하고 비교적 짧은 시간 내에 유명 관광지를 구경할 수 있다는 점에 있어서 다양한 고객들에게 인기가 많다.

이때 탑승 수속, 표 실물 발급 과정 등 오프라인 기반으로 진행되는 과정을 온라인 기반으로 이동시킴으로써 기존에 진행되던 절차를 생략하거나 줄이고, 표 분실 등 위험성을 줄일 수 있다고 파악하게 되었다. 따라서 현재 팀은 오프라인 기반의 표 발급을 온라인 기반으로 수행하고, QR 코드나 바코드를 제공하여 탑승 수속을 간편하게 하여 고객들의 편의성을 높이는 것을 목적으로 프로젝트를 진행했다.

구현 범위의 경우 팀의 규모가 다른 팀에 비해 작은 편이고, 외국인 학생도 포함되어 있는 점을 감안하여 사용자와 시스템의 상호작용 수준으로 제한했다.

2. 설계 및 구현 비교

01. 상품 조회



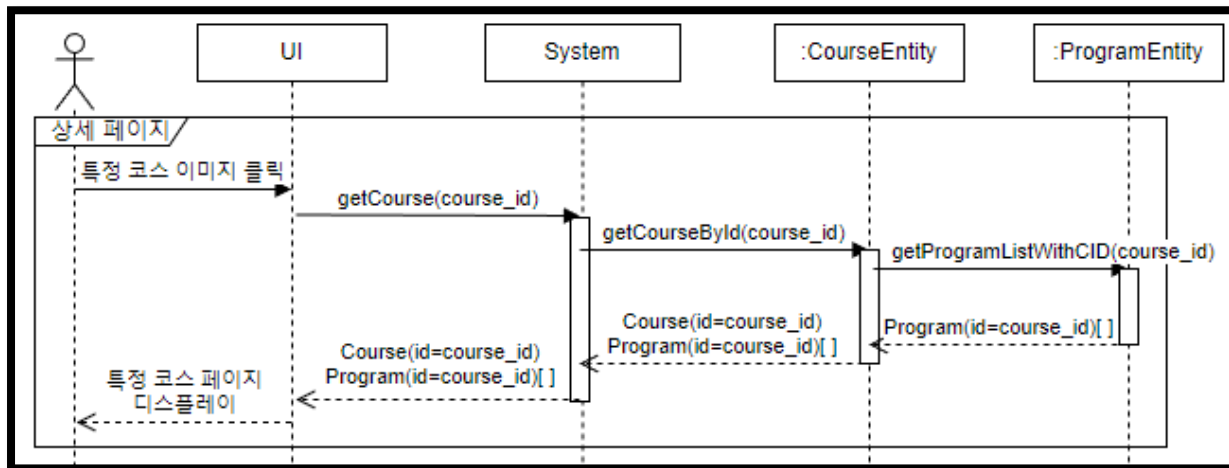
```
route("server/searchCourse") {
  get {
    // course 정보 반환
    val allCourse = courseController.getAllWithThumbnail()

    call.respond(allCourse)
  }
}
```

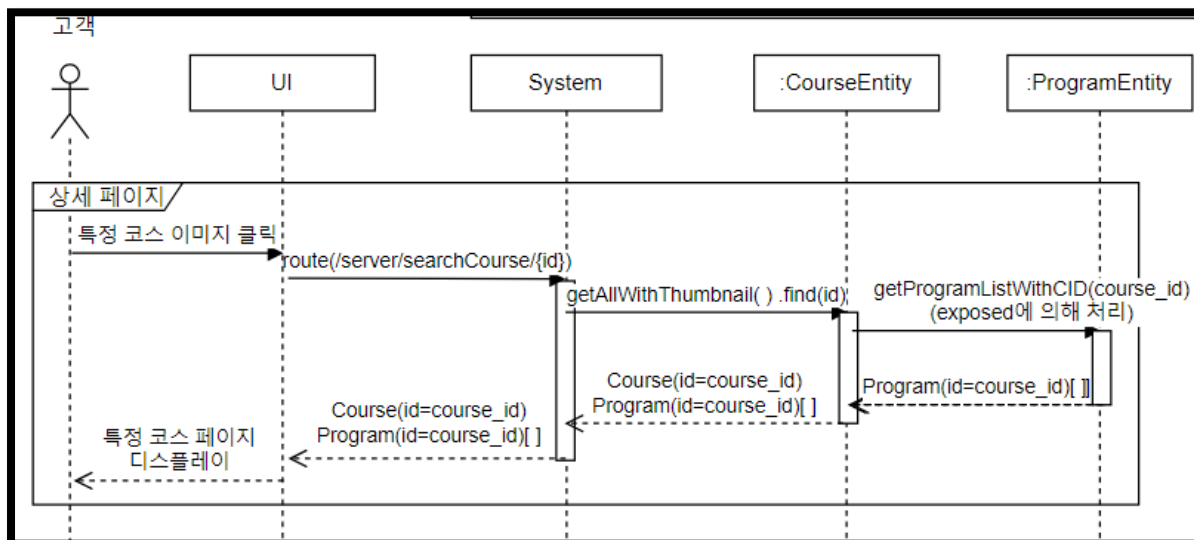
사용자가 /server/searchCourse 경로에 대해 GET 요청을 보내면 courseController의 getAllWithThumbnail 메서드를 호출하여 대응되는 정보를 전달한다.

02. 상세 조회

과거 설계 버전



현재 구현된 버전



```

route("server/searchCourse/{id}") {
  get {
    // order의 id로 찾은 course 정보 반환
    val id = call.parameters["id"]!!.toInt()
    val course = Json.encodeToString(
      courseController.getAllWithThumbnail().find { it.id == id }
    )

    call.respond(course)
  }
}

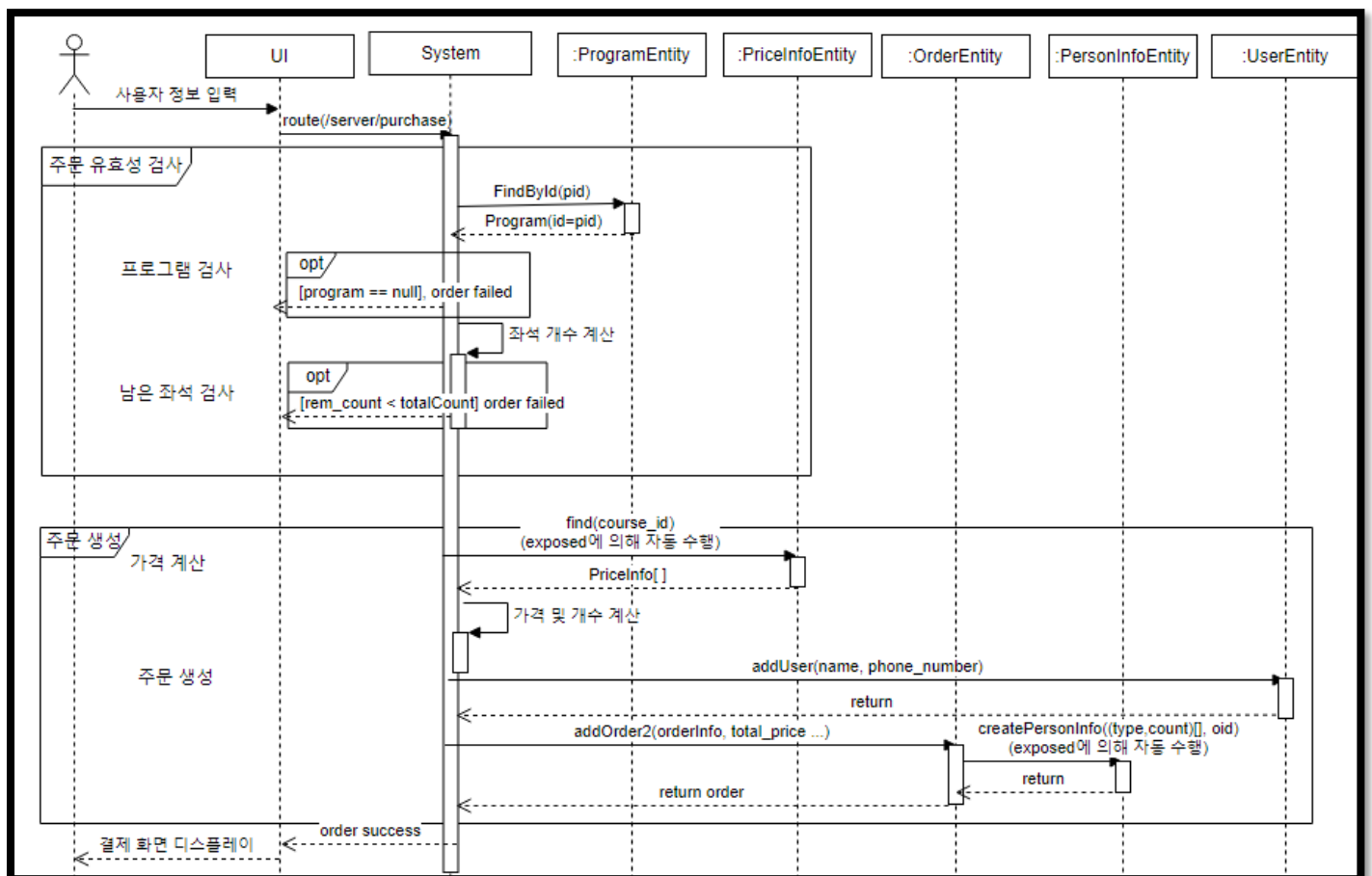
```

사용자가 `/server/searchCourse/{id}` 경로에 GET 메서드로 http 메서드를 보내면 서버에서는 경로의 id값을 파싱하여 대응되는 정보를 `courseController`의 `getAllWithThumbnail().find`을 통해 전달한다.

기존 시퀀스 다이어그램에서는 `getCourseById`을 통해 특정 id 값을 가진 데이터만 가져오도록 설계했으나, 실제 구현 단계에서는 `getAllWithThumbnail()` 함수를 통해 모든 데이터를 가져온 후 배열에 대해 `find` 메서드를 통해 특정 id 값을 가져오도록 구현했다.

현재 프로젝트에서는 `exposed`라는 ORM 라이브러리가 사용되었다. 기존 `getProgramListWithCID`는 해당 라이브러리가 지원하는 기능에 의해 `course` 정보를 가져오는 순간 자동으로 수행된다.

03. 예약



예약 단계는 주문 유효성 검사와 주문 생성 단계로 나뉜다. 예약은 사용자가 UI를 통해 `/server/purchase` 경로로 POST 요청을 보내는 경우 성립한다.

```

route("server/purchase") {
    post p@{
        val purchase = call.receive<PurchaseInfo>();
        var program: Program? = null;
    }
}
  
```

서버에서는 `/server/purchase` 경로를 통해 요청을 받을 수 있다. POST의 body 부분은 `purchaseInfo`라는 형태로 저장된다. 해당 데이터 클래스는 유저 정보 등의 결제 정보(인원 수 등 포함)를 가진다.

```

transaction {
    val prog = ProgramEntity.findById(purchase.pid);
    program = prog?.getProgram() ?: null;
}
if (program == null) {
    call.respond(status = HttpStatusCode.BadRequest, message = "대응되는 프로그램이 존재하지 않음")
}

```

```

var totalCount = 0; // 전체 개수
purchase.priceinfos.forEach { cur ->
    run {
        totalCount += cur.count
    }
}; // 개수 구하기

// 남은 개수가 더 많으면 오류 상황
if (totalCount > program!!.rem_count) {
    call.respond(status = HttpStatusCode.BadRequest, message = "주문 가능한 개수 초과")
}

```

위 두 코드는 주문의 유효성 검사 과정을 보여주고 있다. 조건을 만족하지 못하는 경우 에러 메시지를 반환한다.

```

transaction {
    //개수 업데이트.
    val prog = ProgramEntity.findById(purchase.pid)!!;

    prog.rem_count -= totalCount;

    val course = prog.course;
    val priceinfos = course.priceinfos;

    var total_price = 0;

    val list = mutableListOf<Personinfo>()

    purchase.priceinfos.forEach { it ->
        run {
            val priceinfo = priceinfos.find { v -> v.type == it.type };
            if (priceinfo != null) {
                total_price += it.count * priceinfo.price;
                val value = Personinfo(it.type, it.count, priceinfo.price)
                list.add(value)
            }
        }
    }
}

```

가격 계산 단계에서는 코스 id를 기반으로 priceinfo를 가져와야 하나, exposed에서 1:N 등 관계로 연결된 데이터에 대해 자동으로 가져와주는 기능을 제공하여 따로 구현되지 않는다. 가격 계산 시점에는 가격 및 주문 생성을 위한 데이터를 생성하고 있다.

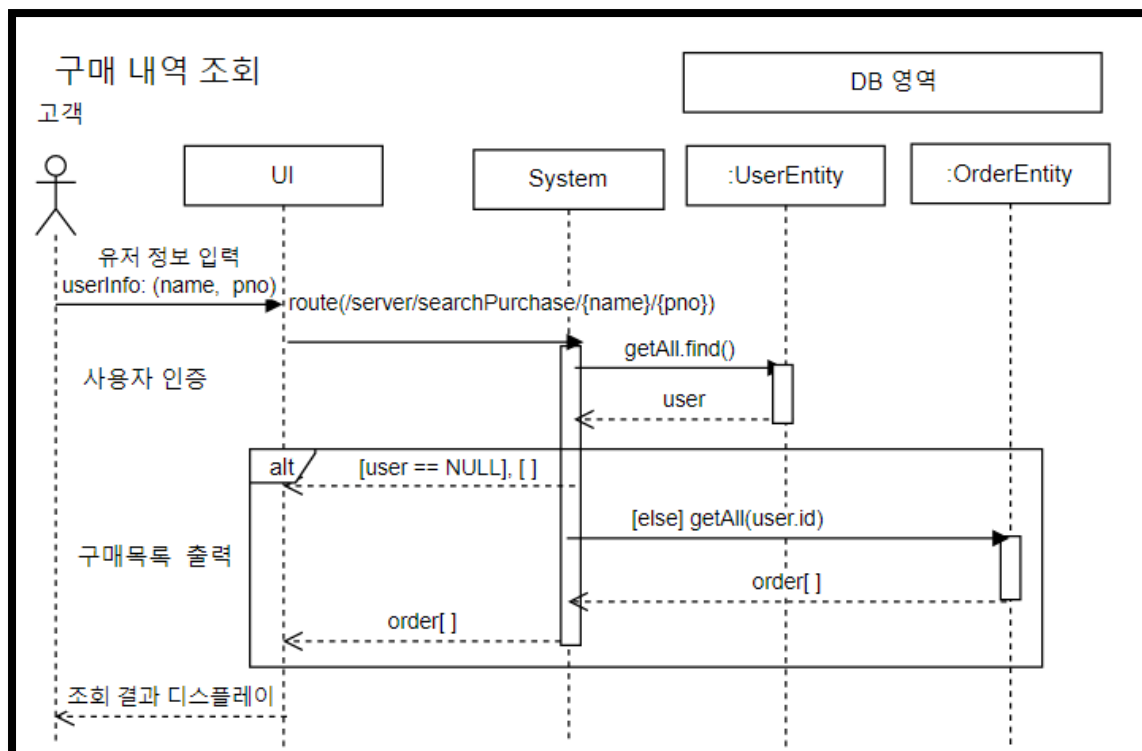
```
val user = userController.addUser(purchase.name, purchase.phone_number);

val order = orderController.addOrder2(
    total_price,
    purchase.card_number,
    prog,
    user,
    list
);
oid = order.id.value;
}

call.respond(status = HttpStatusCode.OK, oid);
```

이후 시퀀스 다이어그램에서 묘사된 것처럼 유저 및 주문 정보를 생성한다. addUser의 경우 이름 및 전화번호 조합에 대해 데이터베이스 상 중복이 존재하는 경우 해당 유저를 가져오고, 그렇지 않은 경우 새로운 유저를 생성하도록 구성되어 있다.

04. 구매내역 조회




```

route("server/searchPurchase/{userName}/{phoneNumber}") {
  get {
    val userName = call.parameters["userName"]!!.toString()
    val phoneNumber = call.parameters["phoneNumber"]!!.toString()
    val user = userController.getAll().find { (it.name == userName) && (it.phone_number == phoneNumber) }

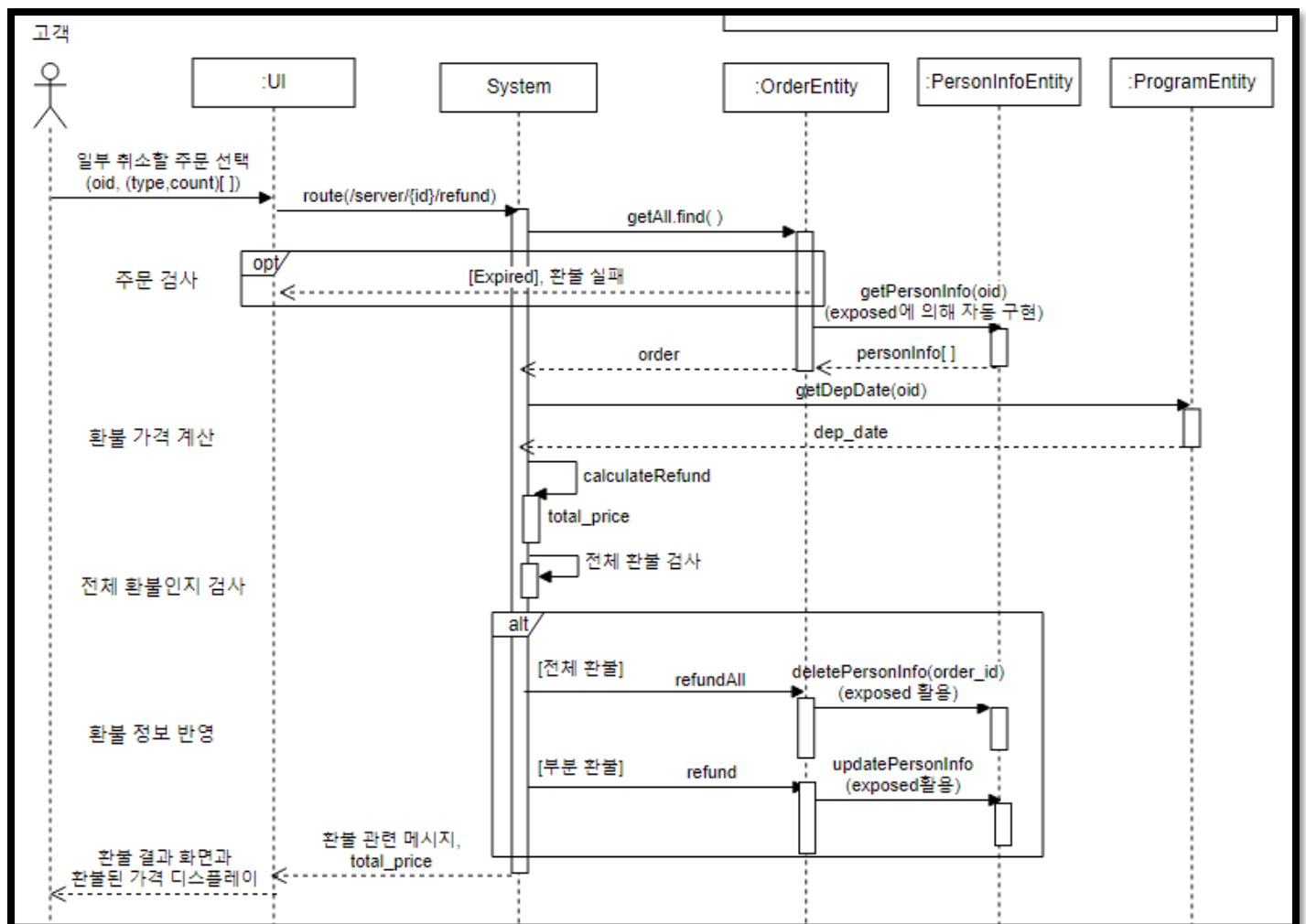
    var orderArr: Array<Order>? = null
    if (user != null) {
      orderArr = orderController.getAll(user.id)
    }
    val result = Json.encodeToString(orderArr)

    call.respond(result)
  }
}

```

사용자는 /server/searchPurchase/{name}/{pno} 경로에 접근하여 구매내역을 조회할 수 있다. 입력된 파라미터는 각각 userName, phoneNumber으로 파싱되어 내부적으로 유저를 얻는데 사용된다. 이후 시퀀스 다이어그램처럼 주문 정보를 가져와 사용자에게 반환한다.

05. 환불



```

route("/server/{id}/refund") {
    post {
        @Serializable
        data class pData(val p1: Int, val p2: Int, val p3: Int)

        val data = call.receive<pData>()
        val id = call.parameters["id"]!!.toInt()
        val sum = data.p1 + data.p2 + data.p3
        val order = orderController.getAll().find{ it.id.toInt() == id }!!

        if(order.state == "expired") {
            call.respond("expired")
        }

        val size: Int? = order.personinfos.size
        var personCount: Int = 0

        for (i: Int in 0 until size!!){
            personCount += order.personinfos[i].count
        }

        // 전체 환불
        if(sum == personCount) {
            orderController.refundAll(order)
        }
        // 부분 환불
        else {
            val difference = personCount - sum
            orderController.refund(order, data.p1, data.p2, data.p3, difference)
        }

        call.respond("ok")
    }
}

```

사용자는 /server/{id}/refund 경로로 POST 요청을 보내 환불을 진행할 수 있다. 데이터는 pData 형식으로 전달되며, int 형으로 형 변환된 id 값을 이용하여 order 정보를 가져온다. 만약 가져온 주문이 만료되었다면 환불 실패로 간주하고 반환한다.

주문 실패가 아닌 경우 환불 타입을 계산한다. 환불 타입이 전체 환불인 경우 refundAll을, 부분 환불인 경우 refund를 진행한다.

현재 프로젝트 코드에서는 findById 형태로 구현되어야 하는 코드들이 상당 부분 getAll().find 형식으로 구현되어 있으며, 일부 http 메서드 사용 방식이 잘못된 부분이 존재한다. 이러한 부분은 ktor 기반 서버 환경에 익숙하지 않은 점 및 구현 기간에 팀원 2명이 코로나에 감염되는 바람에 개발을 수행할 수 있는 기간이 상당히 제한되어 코드를 리팩토링 하기에 시간이 촉박해 구현 자체에 초점을 맞추고 개발을 진행했기 때문이다. 만약 장기적으로 현재 프로젝트를 진행하게 된다면 데이터베이스 부하를 고려한 컨트롤러 구현, 추가적인 에러 처리 코드 작성 및 http 메서드 사용 개선 등이 수행되어야 할 것으로 판단된다.

3. 작동 사례

구현한 시스템에 대하여 작동 사례를 제시하며 기존에 작성된 UI 설계와 비교한다.

1. 여행지 찾기에서 상품 리스트 출력(상품 조회)

여행은 트래블

여행지 찾기구매내역 조회

여행지 찾기

이미지 1

속초
한줄 설명

이미지 2

강릉
한줄 설명

이미지 3

서울
한줄 설명

이미지 4

부산
한줄 설명

여행지 찾기

여행지 검색 ex) 강원도 강릉



여행지 목록



서울
서울 투어



부산
부산 투어



강원도
강원도 투어

사용자가 /search 경로에 접근하거나 상단 바의 '여행지 찾기' 버튼을 클릭하는 경우 현재 페이지로 이동한다. 현재 페이지에서는 서버 내에 존재하는 여행지 목록을 출력한다. 각 코스를 보여주는 부분이 UI 설계와 배치 상의 차이가 존재하지만, 큰 차이 없이 설계 사항을 구현하고 있다.

2. 상품 클릭 시 해당 상품의 상세 정보 출력(상세 조회)

여행은 트래블

여행지 찾기

구매내역 조회

속초 여행

대표 이미지

가격

성인	~~~~
청소년	~~~~
아동	~~~~

기간


기간 1

기간 2

...

상세 설명 ...

서울



가격	
아동	5000
청소년	7000
성인	10000

기간	
출발	도착
22년 11월 30일 오전 12:00	22년 12월 01일 오후 12:33
22년 12월 10일 오전 12:00	22년 12월 11일 오후 12:33

간략한 설명

서울 투어

상세 설명

서울 ~~ 투어

예약하기

사용자가 '여행지 찾기' 페이지에서 특정 코스를 선택하는 경우 도착하는 페이지이다. '상세 조회'에 해당하며 전반적으로 UI 설계 사항을 잘 구현하고 있다. 아래 '예약하기' 버튼을 클릭하는 경우 예약 절차를 진행할 수 있다.

3. 예약하기 버튼을 누르면 나오는 구매 화면(예약)

여행은 트래블

여행지 찾기

구매내역 조회

예약

상품 정보

상품 이름

숙초~

기간

세부 기간 선택

(남은 좌석 N개)

인원

성인

명

청소년

명

아동

명

구매 가능 메시지

가격

N원

구매자 정보

이름

휴대폰 번호

결제 수단 정보

예약하기

예약

상품 정보

상품 이름

서울

기간

22년 12월 10일 오전 12:00 ~ 22년 12월 11일 오후 12:33

전체 좌석 45개

남은 좌석 37개

인원

아동

1

청소년

4

성인

37

- 구매할 수 없습니다

가격

0원

고객 정보

구매자 이름

성함

*

휴대폰 번호

-을 빼고 입력하세요 (5~16자)

*

결제수단 정보

-을 빼고 입력 (14~16자)

*

예약하기

투어 기간과 인원, 사용자 정보의 정보를 입력하여 예약할 수 있다. 역시 UI 설계와 크게 다르지 않다. 한 번에 예약할 수 인원 수가 일반적인 관광 버스 한 대에 탑승할 수 있는 최대 인원인 45명 수준이라는 점과 사용자의 입력 실수를 방지하기 위한 목적으로 마우스 클릭을 통해서만 인원 수를 늘릴 수 있도록 설정되어 있다. 만약 더 규모가 큰 시스템으로 확장되는 경우 변경될 수 있는 사항이다.

구매자 정보의 경우 사용자의 입력 실수를 방지하기 위하여 사용자의 입력을 정규표현식을 통해 검사한 후 입력을 수행하도록 구현되어 있다. 만약 사용자가 휴대폰 번호에 숫자가 아닌 다른 문자를 입력하려고 시도하는 경우 입력이 제한된다. 사용자가 모든 입력을 유효하게 설정하면 예약하기 버튼이 활성화되며, 예약을 진행할 수 있다.

4. 구매 상세 조회 페이지

여행은 트래블

여행지 찾기
구매내역 조회

예약 완료

상품 정보

상품 이름

속초~

기간

선택한 기간

인원

성인

N명

청소년

N명

아동

N명

가격

N원

구매자 정보

이름

~~~~~

휴대폰 번호


~~~~~

결제 수단 정보

~~~~~

~~~세부 메시지들

구매 확인



서울

서울 투어

상품 정보

상품 이름

서울

출발일

2022-11-30T00:00

도착일

2022-12-01T12:33:32

인원

아동

1 (5000)

청소년

2 (14000)

성인

3 (30000)

총 가격

49000

구매자 정보

구매자 이름

구매테스트

휴대폰 번호

01000000000

주문 정보

주문 ID

29

주문 시각

2022-12-09T18:16:16.949460

주문 갱신 시각

2022-12-09T18:16:16.949460

결제수단 정보

0000000000000000

주문 상태

ok

조회 페이지로 이동

예약이 성공적으로 수행되는 경우 '구매 확인' 페이지로 이동한다. 기존 설계에서는 예약 완료 페이지, 환불 완료 페이지 등을 개별적으로 구현할 예정이었으나, 해당 페이지들이 제공하는 정보가 근본적으로 상품에 대한 상세 정보를 제공한다는 공통점을 발견하여 하나의 페이지를 통해 언급한 페이지의 정보를 제공하기로 하였다.

여행은 트래블

여행지 찾기

구매내역 조회

구매자 정보

이름

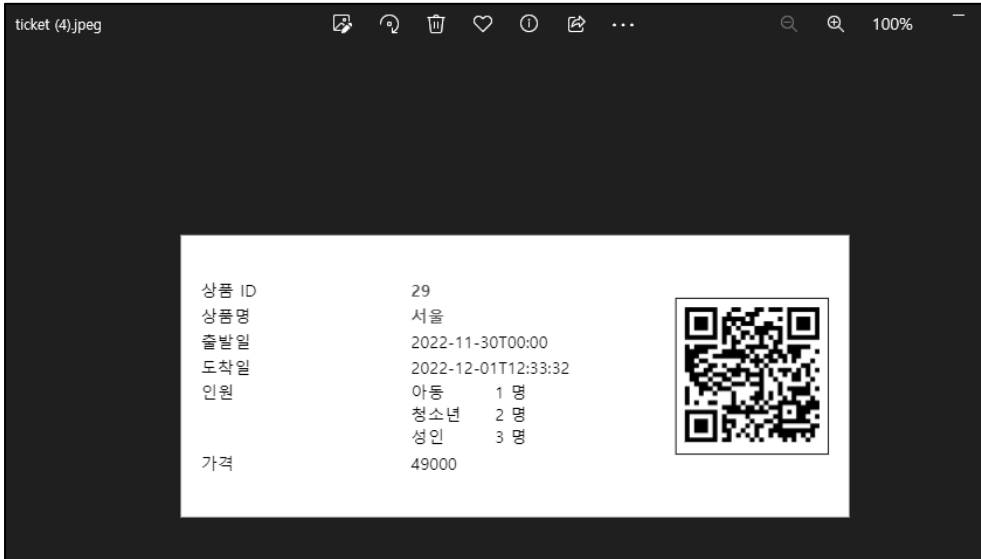
휴대폰 번호

조회

조회 내역

| | |
|--|--|
| <div>속초~
구매 정보를...
상태 결제</div> <div>리셋하기</div> <div>환불하기</div> | <div>대구~
구매 정보를...</div> <div>환불하기</div> |
|--|--|

구매내역 조회 페이지에서는 사용자의 이름 및 전화번호를 이용하여 구매 내역을 조회할 수 있도록 구현하였다. UI 설계 상에서는 각 구매 목록에서 해당 구매 내역에 대한 전체 구매 정보를 제공할 예정이었으나, 구매 확인 페이지에서 확인할 수 있듯이 구매 관련 정보가 상당히 많은 편이라는 점을 고려하여 상세 정보 확인의 경우 상품 ID를 클릭하여 이동할 수 있는 '구매 확인' 페이지를 통해 제공하고, 구매내역 조회 페이지에서는 구매 내역을 표로 사용할 수 있도록 일부 정보 및 QR 코드를 제공하도록 변경했다.



'표 출력'을 누르면 로컬 영역에 표가 다운로드 되는 기능이 존재한다. 사용자는 해당 기능을 이용하여 차후 페이지에 방문하지 않고 탑승 수속을 진행할 수 있다.

6. '환불'을 누르면 나오는 화면(환불)

환불

상품 정보

| | |
|-------|--|
| 상품 이름 | 서울 |
| 출발일 | 2022-11-30T00:00 |
| 도착일 | 2022-12-01T12:33:32 |
| 인원 | 아동 1 (5000)
청소년 2 (14000)
성인 3 (30000) |
| 총 가격 | 49000 |

주문 정보

| | |
|----------|----------------------------|
| 주문 ID | 29 |
| 주문 시각 | 2022-12-09T18:16:16.949460 |
| 주문 갱신 시각 | 2022-12-09T18:16:16.949460 |
| 결제수단 정보 | 0000000000000000 |
| 주문 상태 | ok |

환불

| | |
|-------|--------------------------------------|
| 환불 타입 | <input type="button" value="전체 환불"/> |
| 아동 | <input type="text" value="1"/> |
| 청소년 | <input type="text" value="2"/> |
| 성인 | <input type="text" value="3"/> |

'구매내역 조회' 페이지에서 구매 상품의 '환불' 버튼을 클릭하는 경우 환불 페이지로 이동한다. 환불 페이지에서는 전체 환불 및 부분 환불을 선택하여 환불 절차를 진행할 수 있다.

환불

환불 타입

부분 환불

아동

1

청소년

0

▼

성인

3

환불

환불 타입

전체 환불

아동

1

청소년

0

성인

3

전체 환불 버튼을 한번 눌러 부분 환불로 전환 후, 환불 인원을 선택할 수 있다. 환불 버튼은 토글 방식으로 동작한다.

환불

상품 정보

상품 이름

출발일

도착일

인원

총 가격

주문 정보

주문 ID

주문 시각

주문 경신 시각

결제수단 정보

주문 상태

환불 확인

환불이 수행된 이후 사용자는 환불을 반복할 수 없습니다.

예를 들어 사용자의 실수로 예정보다 더 많은 인원을 환불하더라도 환불을 취소할 수 없으므로 사용자는 전체 표를 취소한 이후 다시 결제 과정을 진행해야 합니다.

따라서 착오로 인한 불편이 발생하지 않도록 점검 후 환불을 진행해주시길 바랍니다.

환불이 완료되면 구매 확인 페이지로 이동합니다.

☒ 위 제시된 설명을 확인하였습니다.

확인

취소

환불 전 실제로 환불을 수행할 것인지 모달 창을 띄워 확인한다.

구매 확인



서울

서울 투어

상품 정보

| | |
|-------|--|
| 상품 이름 | 서울 |
| 출발일 | 2022-11-30T00:00 |
| 도착일 | 2022-12-01T12:33:32 |
| 인원 | 아동 1 (5000)
청소년 0 (0)
성인 3 (30000) |
| 총 가격 | 35000 |

구매자 정보

| | |
|--------|-------------|
| 구매자 이름 | 구매테스트 |
| 휴대폰 번호 | 01000000000 |

주문 정보

| | |
|----------|----------------------------|
| 주문 ID | 29 |
| 주문 시각 | 2022-12-09T18:16:16.949460 |
| 주문 갱신 시각 | 2022-12-09T18:16:16.949460 |
| 결제수단 정보 | 0000000000000000 |
| 주문 상태 | ok |

조회 페이지로 이동

- 구매 내역 확인에서 확인한 모습

구매 목록

| | |
|-------|-----------------------------|
| 상품 ID | 28 |
| 상품명 | 서울 |
| 출발일 | 2022-11-30T00:00 |
| 도착일 | 2022-12-01T12:33:32 |
| 인원 | 아동 1 명
청소년 0 명
성인 3 명 |
| 가격 | 35000 |



표 출력

환불하기

- 전체환불 확인(주문 상태가 expired로 되어있음)

주문 정보

| | |
|----------|----------------------------|
| 주문 ID | 28 |
| 주문 시각 | 2022-12-09T17:17:32.951009 |
| 주문 갱신 시각 | 2022-12-09T17:17:32.951009 |
| 결제수단 정보 | 0000000000000000 |
| 주문 상태 | expired |

3. 테스트

올바른 형식으로 작동되는 것은 위의 시연 과정에서 확인했기 때문에, 올바른 형식이 아닌 입력 값이 들어왔을 때 작동을 테스트해보았다. (비정상적 동등 클래스 테스트)

| 입력 | 정상적인 동등 클래스 | 비정상적 동등 클래스 |
|------------------|---|--|
| 인원
(예약) | EQ1: 남은 좌석보다 작은 정수 | IEQ1.1: 남은 좌석보다 큰 정수
IEQ1.2: 음수나 정수가 아닌 입력 |
| 고객 정보
(예약) | EQ2: 정상적인 핸드폰 번호 길이의 정수
EQ3: 정상적인 결제수단 정보 길이의 정수 | IEQ2.1: 정수가 아닌 입력
IEQ2.2: 정상적인 핸드폰 번호의 길이보다 길거나 짧은 길이의 정수
IEQ3.1: 정수가 아닌 입력
IEQ3.2: 정상적인 핸드폰 번호의 길이보다 길거나 짧은 길이의 정수 |
| 고객 정보
(구매 확인) | EQ4: 존재하는 구매자의 정보 형식에 맞게 입력 | IEQ4.1: 존재하지 않는 사용자 정보 입력
IEQ4.2: 핸드폰 번호에 정수가 아닌 입력
IEQ4.3: 정상적인 핸드폰 번호의 길이보다 길거나 짧은 길이의 정수 |
| 환불 | EQ5: 사용가능한 표를 환불시도 | IEQ6: 이미 전체 환불된 표를 환불시도 |

- 예약

예약

상품 정보

상품 이름

서울

기간

22년 11월 30일 오전 12:00 ~ 22년 12월 01일 오후 12:33 ▼

인원

전체 좌석 45개

남은 좌석 32개

아동 3

청소년 8

성인 33

가격

- 구매할 수 없습니다

0원

고객 정보

구매자 이름

조광호

휴대폰 번호

0101122222222222

결제수단 정보

555 *

예약하기

고객 정보

구매자 이름

성함

*

휴대폰 번호

-을 빼고 입력하세요 (5~16)

*

결제수단 정보

-을 빼고 입력 (14~16자)

*

입력 형식을 제시한다. 인원은 사용자가 마우스 클릭으로 조절 가능하도록 되어있음. (사용자가 잘못된 형식의 정보를 입력하는 것을 막음) → IEQ1.1, IEQ1.2 테스트

남은 좌석보다 많이 선택하거나 고객 정보(이름, 번호, 결제수단)가 형식에 맞지 않으면 예약하기 버튼을 누를 수 없게 되어있다. → IEQ1, IEQ2.2, IEQ3.2 테스트

또한 정수가 아닌 문자를 휴대폰 번호와 결제수단 입력란에 입력하면 입력이 안된다. → IEQ2.1, IEQ3.1 테스트

- 조회

구매내역 조회

구매자 정보

이름

구매테스트

전화번호

999999

조회

구매 목록

조회된 구매 내역이 없습니다!

구매내역 조회

구매자 정보

이름

구매테스트

전화번호

01022227123

조회

구매 목록

조회된 구매 내역이 없습니다!

존재하지 않는 사용자 정보로 조회하면 아무것도 나오지 않는다. (전화번호가 숫자 형식이 아니거나 형

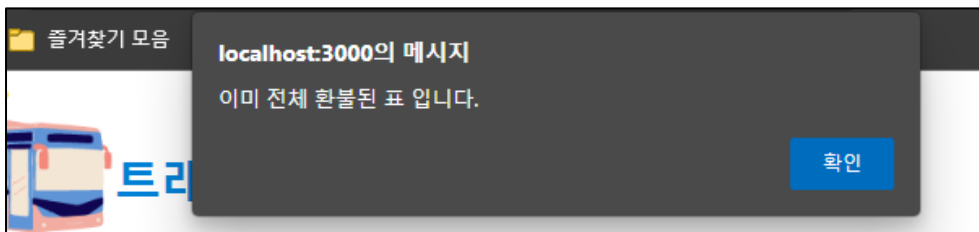
식에 맞지 않아도 조회 결과가 나오지 않음) → IEQ4.1, IEQ4.2, IEQ4.3 테스트

- 환불

환불

| | |
|-------|-------|
| 환불 타입 | 부분 환불 |
| 아동 | 1 |
| 청소년 | 0 |
| 성인 | 3 |

부분 환불 또한 사용자가 직접 입력하지 못하게 하였다. 마우스로 인원을 조절 가능하게 하여, 최대 인원을 넘거나 '-1'이 되지 않도록 제한한다. (사용자가 잘못된 형식의 정보를 입력하는 것을 막음)



이미 환불된 표(expired 상태)를 환불하려고 하면 환불하지 않고, 위와 같은 메시지 창을 띄운다.

→ IEQ6 테스트

4. Github 주소

현재 프로젝트에서 사용된 전체 코드는 아래 깃 허브 사이트 상에 공개되어 있다.

<https://github.com/fish9903/Bus-Tour>