★ JUGAN COFFEE ★

# 주간 커-피

{ 리액트 & 스프링부트 기반 X 커피 구독 서비스 }

# 프로젝트 주제 및 동기

- **프로젝트 주제**

  리액트 & 스프링 부트 기반

  커피 원두 구독 서비스 & 클라이언트 서비스

- **프로젝트 동기**

  1. 코로나 이후 홈카페의 수요 증가
  2. 온라인 판매에 의한 소상공인 상생 가능 방안
  3. 온라인 구독 경제 활성화

# 사용기술

## 통신

RabbitMQ

## 프론트엔드 프레임워크 / 라이브러리

Bootstrap 5

NEXT.JS

Redux

Redux-Saga

## 백엔드 프레임워크 / 라이브러리

spring

spring boot

Lombok

Spring Data JPA

## 클라우드 서버

Amazon EC2

Amazon RDS

## 사용 언어

HTML

CSS

Sass

TS

Java

## 데이터베이스

PostgreSQL

프로젝트 개요

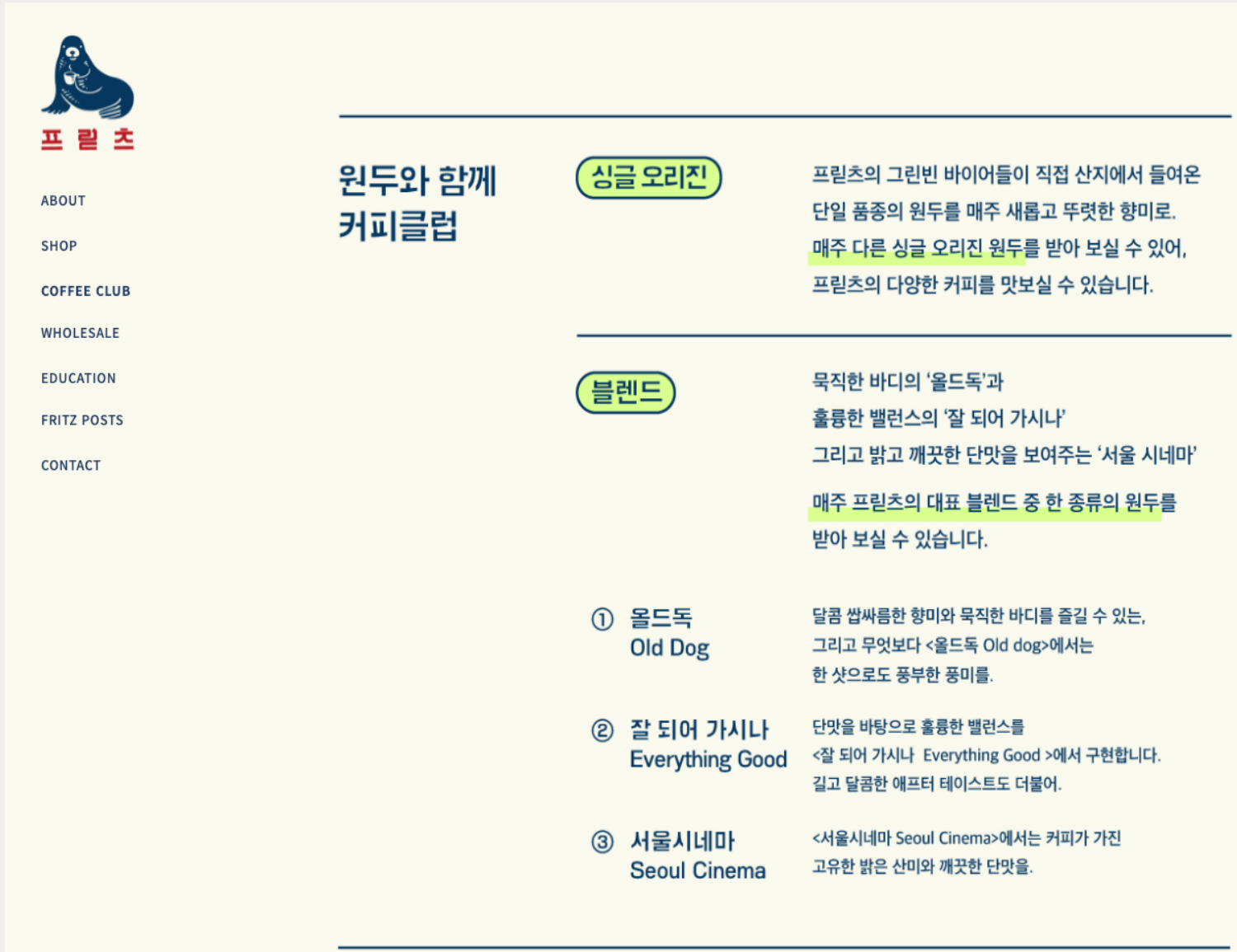
# 사례분석 - [코케]

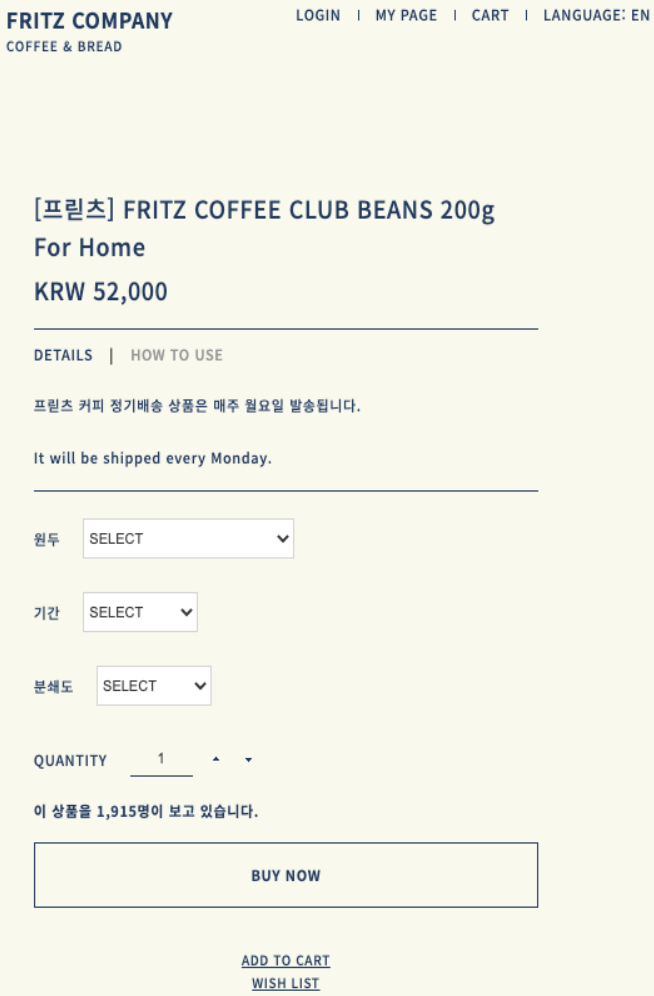
- 원두 구독 서비스를 운영하는 기업을 모델링 하여 프로젝트를 구상했습니다.
- 클라이언트 서비스가 구현해야 할 부분을 조사하여 반영하였습니다.

프로젝트 개요

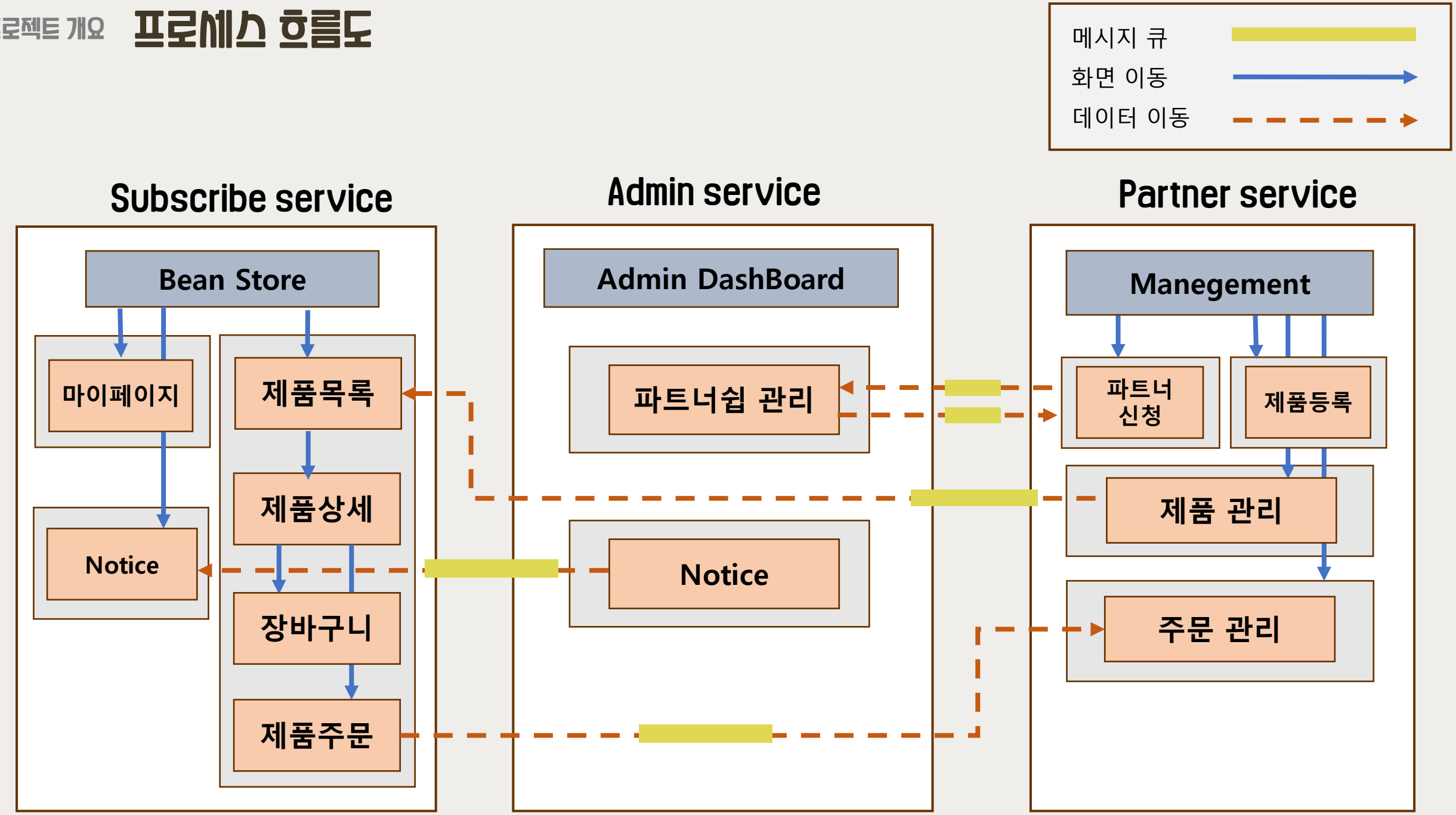
# 사례분석 - [프릳츠]

- 원두를 판매하는 로스터리의 홈페이지를 참고하여 UI를 설계했습니다.
- 원두 구독(판매) 시 필요한 데이터를 수집하였습니다.



# 프로세스 흐름도


메시지 큐
화면 이동
데이터 이동
Subscribe service
Admin service
Partner service
Bean Store
마이페이지
제품목록
제품상세
Notice
장바구니
제품주문
Admin DashBoard
파트너쉽 관리
Notice
Manegement
파트너 신청
제품등록
제품 관리
주문 관리

# 프로세스 일정

| 프로젝트 일정 | 1주차 | 2주차 | 3주차 | 4주차 | 5주차 | 6주차 | 7주차 | 8주차 |
|---|---|---|---|---|---|---|---|---|
| 기획 | | | | | | | | |
| 분석 및 설계 | | | | | | | | |
| 단위 서비스 구현 | | | | | | | | |
| 단위 테스트 | | | | | | | | |
| 통합 테스트 및 배포 | | | | | | | | |
| 산출물 정리 | | | | | | | | |

# 프로젝트 역할 분배

**Project Leader**

**이주은**

**구독자 서비스**

**협업 툴 관리 (Notion)**
**프론트엔드, 백엔드**
**UI 디자인**

**Developer**

**명재윤**

**파트너 서비스**

**Developer**

**김준호**

**어드민 서비스**

# 요구사항 분석

| BeanStore Service | | | |
|---|---|---|---|
| BeanStore Home | 홈페이지 | BeanStore 시작화면<br>- 상단 : 네비게이션바를 보여준다.<br>- 메인 : 간단한  브랜드 소개, 시작하기 [ 배경이미지 ]를 보여준다. | 이주은 |
| | 네비게이션 | 네비게이션바<br>- 네비게이션바 카테고리를 표시한다. [ 제품 목록 ], [ 마이페이지 ], [ contact us ] | |
| Notice | Notice | Notice 를 클릭하면 Notice 상세화면이 나온다<br>- 공지사항<br>  - 어드민에서 작성한 공지사항을 볼 수 있다, 테이블 형태로 보여진다.<br>  - 페이지를 넘길 수 있다. | |
| 제품목록 | 제품목록 | 제품 목록버튼 클릭시 목록 화면으로 이동한다.<br>- 제품 목록을 조회한다.<br>- 4x4의 [그리드 스타일]로 보여준다.<br>- 각 제품의 사진밑에 이름과 가격 등을 보여준다. | |
| 제품상세 | 제품 상세 | 제품을 클릭하면 해당 제품정보 및 구독 옵션 선택 창으로 이동한다.<br>- 제품정보 [ 제품사진, 원두정보, 판매자 정보, 좋아요버튼 ]을 보여준다.<br>- 구독옵션 [ 가격, 원두용량선택 , 구독기간 선택, 분쇄도 선택, 수량, 총 결제금액 ]을 보여준다. | |
| 구독 상세 | 구독 신청 | 구독 신청 버튼 클릭시 구독신청 상세 화면으로 이동한다.<br>-  [ 입력 폼 스타일 ]로 화면에 보여준다.<br>- 주문정보 확인, 이름, 카드정보, 배송지, 전화번호를 입력할수 있는 화면을 보여준다. | |
| 마이페이지 | 마이페이지 | 마이페이지 버튼 클릭시 해당 화면으로 이동한다.<br>- 현재 내 주문정보 확인 [ 테이블 형태 ]<br>- 테이블 각 행 마다 주문한 제품의 사진, 제품명, 결제금액, 주문요청결과를 확인할 수 있다.<br>– 해당 주문정보를 클릭하면 내가 주문한 제품의 주문옵션을 확인할 수 있다. | |

## 분석 및 설계

# WBS

| | 주간 커피구독 시스템 | | | | | 월 | 10 | | | | | | | | 11 | | | | | | | | | | | | | | | | | | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 총 프로젝트 기간 | | | | | 주 | 4주 | | 5주 | | | | 6주 | | 1주 | | | | | 2주 | | | | | 3주 | | | | | 4주 | | | | 5주 | | 1 |
| 번호 | 프로젝트 | 작업 | 예상시간 | 사용시간 | 차이 | 일 | 21 | 22 | 25 | 26 | 27 | 28 | 29 | 31 | 1 | 2 | 3 | 4 | 5 | 8 | 9 | 10 | 11 | 12 | 15 | 16 | 17 | 18 | 19 | 22 | 23 | 24 | 25 | 26 | 29 | 30 | 1 |
| | 주간 커피 | 유즈케이스 작성 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 분석 객체 모델 작성 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | UI Wireframe 작성 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 컴포넌트 다이어그램 작성 ( 인터페이스 ) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 클래스 다이어 그램작성 ( 클래스 설계 ) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 서비스 | Subscribe Service | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 컴포넌트(기능) | BeanStoreHome ( 홈페이지 ) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | Navigation.tsx를 생성하고 App.tsx 와 연동 | 0.5 | 0 | 0.5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | BeanStoreHome 화면 UI 프레임 구현 [ 네비게이션바, 홈화면 ] | 1 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 컴포넌트(기능) | Navigation ( 네비게이션 ) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 직업구현 | Navigation.tsx 하위 컴포넌트 UI 프레임 구현 | 1 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | BeanStoreHome, Products, Contact us, Mypage 컴포넌트를 Navigation.tsx 내부에 연동 | 1 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 컴포넌트(기능) | Products ( 제품목록 ) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 프론트엔드 UI 4X4 그리드 구현 [ BootStrap5 ] | 1 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 프론트엔드 상태 조회 구현 | 2 | 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 백엔드 API 구현 | 2 | 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 프론트엔드 - 백엔드 연동 구현 [ Redux-Saga 사이드이펙트 처리 구현 ] | 2 | 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 백엔드 - 데이터베이스 연동구현 [ Spring Entity 필드 ], [ JPA ] | 2 | 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 백엔드 메시지 큐 수신 연동 구현 [ amazonMQ [ rabbitMQ ]] | 2 | 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 컴포넌트(기능) | ProductsDetails ( 제품상세 ) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 직업구현 | 프론트엔드 UI 프레임 구현 ( 제품정보 / 구독옵션 ) [ BootStrap5 ] | 1 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | Products.tsx 에서 제품 사진 클릭하면 ProductsDetails.tsx로 넘어오게 페이지 이동 처리 [ useHistory , id값 ] | 1 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 프론트엔드 상태 조회 구현 | 2 | 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 백엔드 API 구현 | 2 | 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 프론트엔드 - 백엔드 연동 구현 [ Redux-Saga 사이드이펙트 처리 구현 ] | 2 | 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 구독옵션 선택 사항 넘겨주기 | 2 | 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 컴포넌트(기능) | Subscribe ( 구독신청 ) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 직업구현 | 데이터 설계 ( Partner, Admin 에게 보내질 주문 데이터 ) | 1 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 프론트엔드 입력 폼 UI 구현 [ BootStrap5 ] | 1 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | ProductsDetails.tsx 에서 [구독신청] 클릭하면 Subscribe.tsx로 넘어오게 페이지 이동 처리 [ useHistory ] | 1 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 프론트엔드 Reducer 추가 구현 | 2 | 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 프론트엔드 주소조회 API 심입 | 2 | 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | [ 주문완료 ] 클릭 시 주문 완료 모달창 구현 + Mypage 이동 | 2 | 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 백엔드 API 구현 | 2 | 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 프론트엔드 - 백엔드 연동 구현 [ Redux-Saga 사이드이펙트 처리 구현 ] | 2 | 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 백엔드 - 데이터베이스 연동구현 [ Spring Entity 필드 ], [ JPA ] | 2 | 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 백엔드 메시지 큐 연동 구현 [ amazonMQ [ rabbitMQ ]] ( Partner, Admin ) | 2 | 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 컴포넌트(기능) | Mypage ( 마이페이지 주문조회 ) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 직업구현 | 프론트엔드 주문 정보 조회 테이블 UI 구현 [ BootStrap5 ] | 1 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 프론트엔드 리덕스 처리 구현 | 2 | 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 프론트엔드 상태 조회 구현 | 2 | 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 백엔드 API 구현 | 2 | 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 프론트엔드 - 백엔드 연동 구현 [ Redux-Saga 사이드이펙트 처리 구현 ] | 2 | 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 백엔드 - 데이터베이스 연동구현 [ JPA ] | 2 | 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 컴포넌트(기능) | Mypage Detail ( 마이페이지 주문상세 조회 ) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 직업구현 | Mypage 내부에서 테이블 행 클릭 시 [주문상세] 로 이동하도록 리우팅 처리 | 1 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 주문 상세 정보 조회 테이블 UI 구현 [ BootStrap5 ] | 1 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 프론트엔드 상태 조회 구현 | 2 | 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 백엔드 API 구현 | 2 | 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 프론트엔드 - 백엔드 연동 구현 [ Redux-Saga 사이드이펙트 처리 구현 ] | 2 | 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 백엔드 - 데이터베이스 연동구현 [ JPA ] | 2 | 0 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 프론트엔드 [ 뒤로가기 ] 버튼 구현 | 1 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

분석 및 설계

# 테스트케이스 명세서

- 사용자의 관점에서 이벤트의 흐름, 선행조건, 예상결과를 시나리오로 정리했습니다.

### 1. 테스트 케이스(Test/Use Case)

**[] 대괄호**: Entity Object, 실제 데이터 속성을 가지고 있는 객체, 주로 테이블과 맵핑된 엔티티/리포지터리 객체

**{} 중괄호**: Control Object(시스템), 요청 및 응답에 대한 흐름을 처리하는 객체, 주로 API 컨트롤러/서비스 객체

**() 소괄호**: Boundary Object: 사용자와 시스템 간에 경계를 처리하는 객체, 주로 UI객체

| No | 케이스 명 | 액터 | 이벤트 흐름 | 선행조건 | 예상결과 |
|---|---|---|---|---|---|
| 1. | 홈페이지 접속 | subscriber | 1. [구독자]가 (Beanstore Home)에 접속<br>2. 상단에 (제품목록), (마이페이지), (Contact US) (장바구니) 항목이 있는 (네비게이션바) 출력<br>3. (네비게이션바) 하단에 (브랜드 소개), (배경이미지) 출력 | 홈화면을 (Beanstore Home)으로 지정해야함 | (네비게이션바)를 통해 [구독자]가 페이지 이동을 할 수 있음 |
| 2. | 제품목록 조회 | subscriber | 1. [구독자]가 네비게이션바의 (제품목록) 메뉴를 클릭<br>2. (제품목록) 화면으로 이동<br>3. {제품목록시스템}이 전체 [제품정보] 목록을 출력<br>4. 모든 [제품정보] 목록을 (4X4 그리드 형태)로 출력하고, 각 그리드에 (가격), (제품명), (제품사진), (업체명) 출력 | [제품정보] DB테이블이 있어야함<br><br>[Partner]와 메세지큐로 통신할 서버가 있어야함 | 모든 [제품정보] 목록을 (4X4 그리드 형태)로 출력<br><br>(가격), (원두이름), (제품사진), (업체명) 등 요약한 정보를 출력 |
| 3. | 제품상세 조회 | subscriber | 1. [구독자]가 (제품목록) 중 한 제품을 클릭<br>2. (제품상세) 화면으로 이동<br>3. 화면의 왼편에 (제품사진), (제품명), (원두정보), (파트너정보), (하트아이콘)을 출력<br>4. 화면 오른편에 (구독옵션) 선택 창을 출력함<br>4-1. [구독자]는 (가격), (원두용량), (구독기간), (분쇄도), (수량)을 선택할 수 있고, 선택 사항에 따라 (총 결제금액)을 보여줌<br><br>[구독자]는 (구독옵션) 항목을 모두 선택하고 (구독하기) (장바구니) 버튼 클릭 | [제품정보] DB테이블이 있어야함<br><br>{제품상세시스템}이 전체 [제품정보] 중 1개의 [제품정보]를 id로 조회해서 출력함<br><br>[구독자]는 (구독옵션) 항목을 모두 선택해야 다음 화면으로 넘어갈 수 있음 | 왼편에 한 건의 [제품정보]를 폼 형식으로 출력<br><br>오른편에 (구독옵션) 선택 폼 출력<br><br>(구독옵션)창은 오른편에 고정, (제품정보)는 왼편에서 스크롤 가능<br><br>(총 결제금액) 옆에 (구독하기) (장바구니) 버튼 출력<br><br>[구독자]가 옵션 선택 후 (구독하기) 클릭 시 (구독신청)으로 이동<br>(장바구니) 클릭 시 {장바구니시스템}이 해당 [주문정보]를 (장바구니)에 추가<br><br>(장바구니)를 클릭 시 (장바구니로 이동하시겠습니까 y/n) |
| 4. | 장바구니 | subscriber | 1. [구독자]가 (네비게이션바)의 (장바구니)클릭<br>1-1. 혹은 (제품상세)의 (장바구니) -> (장바구니 이동) 클릭 시 (장바구니) 화면 출력<br>2. {장바구니시스템}이 (제품상세)에서 (장바구니)에 담았던 모든 [주문목록]을 테이블 형식으로 조회<br>3. 테이블 한 행에 (제품명) (수량) (가격) (업체명) (배송비)와 (주문하기) 버튼 출력<br>4. (전체주문) (선택주문) 버튼이 있고, (체크박스)를 클릭해서 원하는 목록만 선택 주문 가능 | {장바구니시스템}에 들어갈 [주문정보] 임시데이터 테이블 있어야함<br><br>최소 한 개 이상의 [주문정보]를 {장바구니시스템}에 추가해야함 | (장바구니)에 제품이 없으면 (제품없음) 메시지 표시<br><br>테이블 형식으로 (장바구니)에 담은 [주문목록] 표시<br><br>(체크박스)를 클릭해서 주문 별 선택 가능<br>(전체주문) (선택주문) 버튼으로 {구독신청시스템}에 주문 건수 전송 |
| 5. | 구독신청 | subscriber | 1. (제품상세)에서 [구독자]가 (구독옵션) 항목을 모두 선택하고 (구독하기) 버튼 클릭<br>1-1. 혹은 (장바구니) 하단 (구독하기) 버튼 클릭<br>2. (구독옵션)에서 선택한 옵션을 테이블 한 행에 출력<br>3. (주문자정보입력폼)을 (구독신청)에 출력<br>3-1. [구독자]는 (주문자정보입력폼)에 (연락처) (이름) (카드정보) (배송지정보) (배송메모)를 입력함<br>3-2. [구독자]가 [주문자정보]를 (주문자정보입력폼)에 입력 완료<br>4. [구독자]가 (구독) 버튼 클릭<br>5. {구독신청시스템}에서 위 정보를 [주문정보] DB에 추가.<br>6. 페이지 이동하여 (주문완료) 메시지를 보여줌 | [구독자]는 (구독옵션)과 (주문자정보입력폼) 모두 기입 해야함<br><br>[주문정보] DB테이블이 있어야함<br><br>[Partner]/[Admin]와 메세지큐로 통신할 서버가 있어야함<br><br>{구독신청시스템}이 구독을 진행하는 [구독자]에게 임의의 [구독id] 발급 | {구독신청시스템}의 DB에 [구독자]가 등록한 [주문정보]의 데이터가 들어감<br><br>[Partner]/[Admin]에게 각각 [주문정보]를 보내줌<br><br>페이지 이동하여 (주문완료) 메시지를 보여줌 |
| 6. | 마이페이지 | subscriber | 1. [구독자]가 (마이페이지) 메뉴 클릭<br>2. {마이페이지시스템}이 모든 (주문정보) 항목을 테이블로 출력<br>3. (주문정보) 테이블 한 행에 (제품사진) (제품명) (결제가격) (구독기간)을 출력함 | {구독신청시스템}에서 저장한 [주문정보] DB테이블이 있어야함<br><br>(주문정보) 한 행을 클릭하면 (주문상세)로 넘어가도록 링크 연결 | (마이페이지)목록에 {마이페이지시스템}이 전체 [주문정보]를 테이블 형식으로 화면에 출력 |
| 7. | 주문상세 | subscriber | 1. [구독자]가 (마이페이지)에서 (주문정보)테이블 중 행 하나를 클릭<br>2. {주문상세시스템}이 1건의 [주문정보]를 [주문id]로 조회해서 (주문상세) 화면에 출력<br>3. (주문상세) 화면에 테이블 형식으로 (제품사진) (제품명) (수량) (결제가격) (구독기간) (배송지정보)를 출력<br>3-1. (배송지정보)는 (수령인) (연락처) (배송지) (배송메모) 포함 | {구독신청시스템}에서 저장한 [주문정보] DB테이블이 있어야함<br><br>[주문정보] DB테이블에 [주문id]가 포함 되어야함 | [구독자]가 입력한 [주문정보] 중 1건을 (주문상세)에서 보여줌 |

# 객체 분석 모델

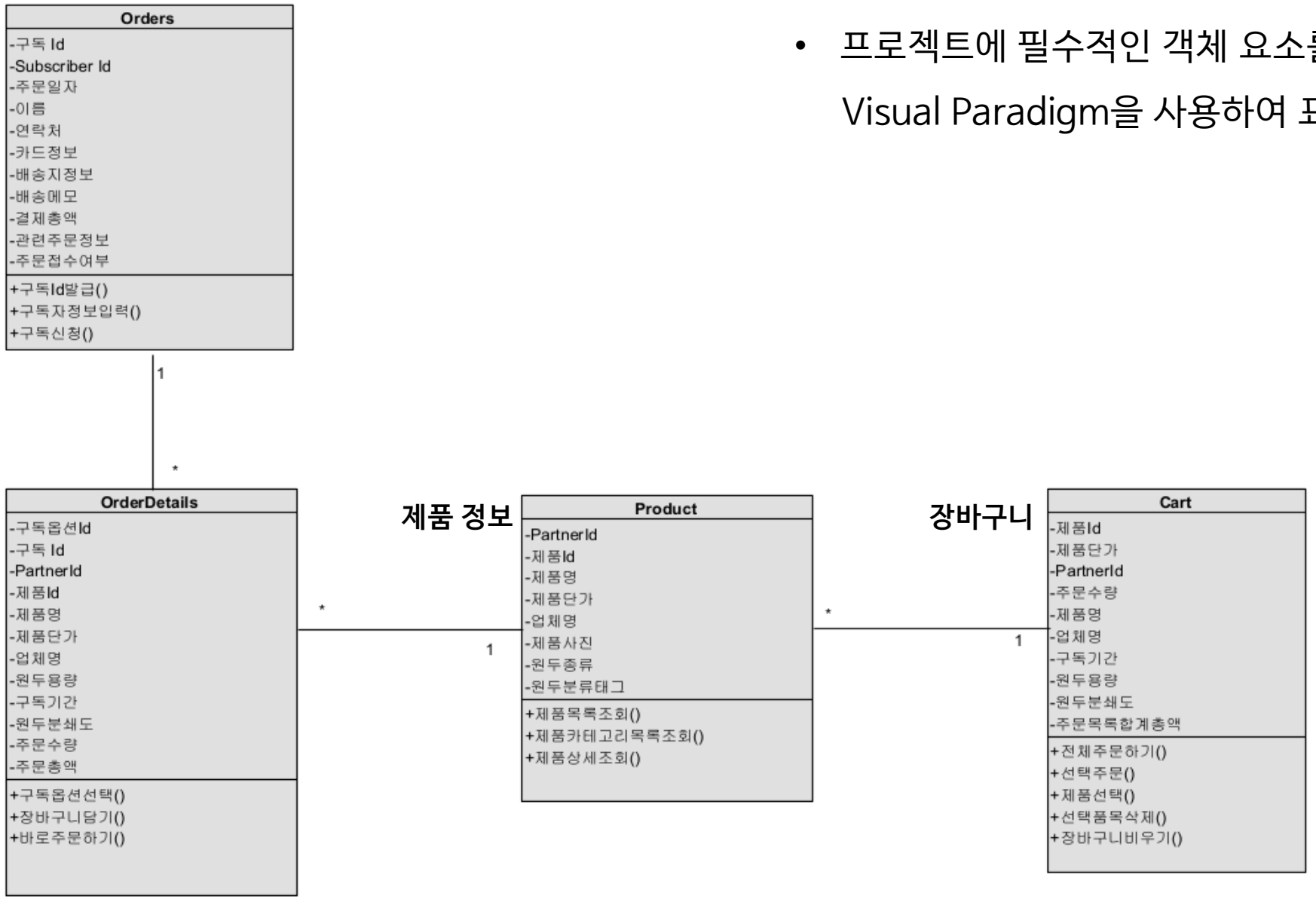


주문 정보

옵션 정보

제품 정보

장바구니

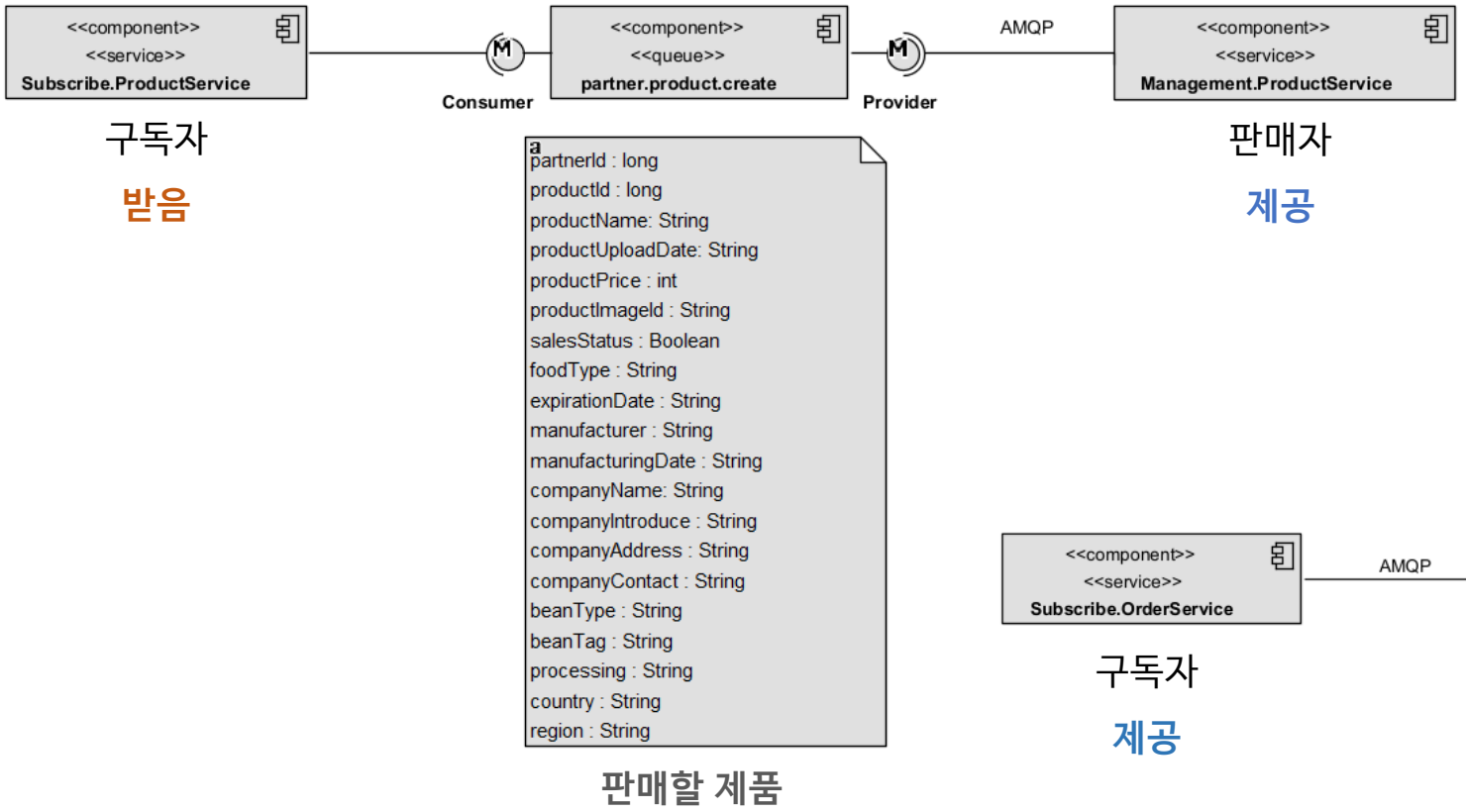



- 프로젝트에 필수적인 객체 요소를 Visual Paradigm을 사용하여 표현했습니다.

# Component Diagram - 인터페이스 설계



- 각 서비스 간에 AMQP를 통해 주고 받을 데이터를 설계하였습니다.



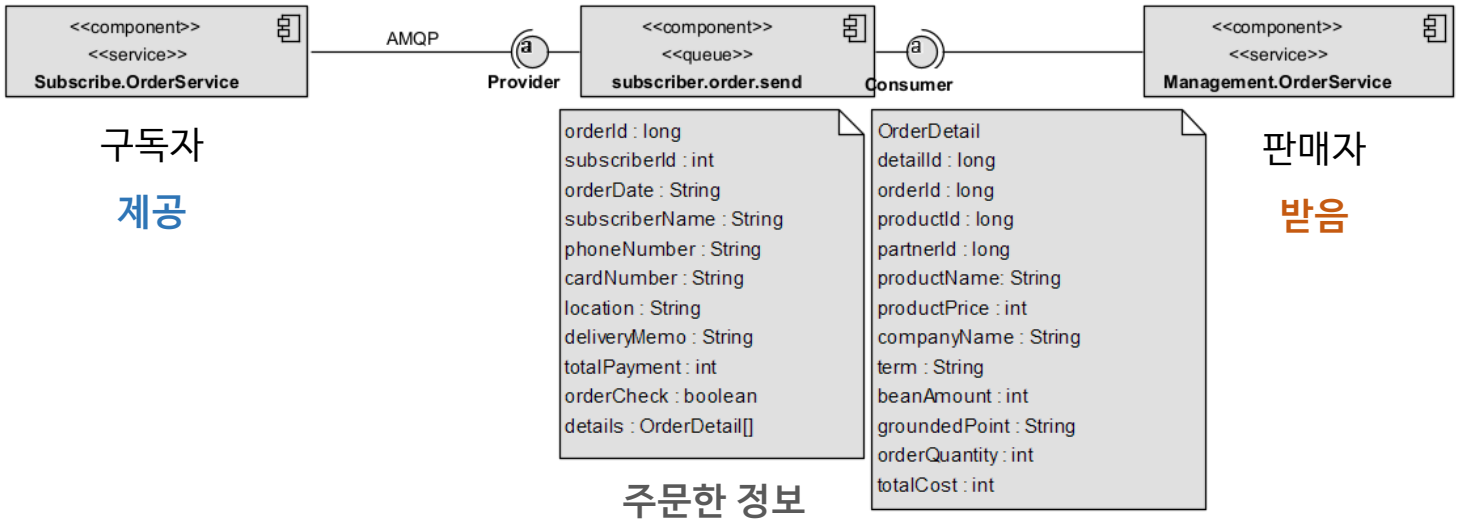

구독자

받음

판매할 제품

판매자

제공




구독자

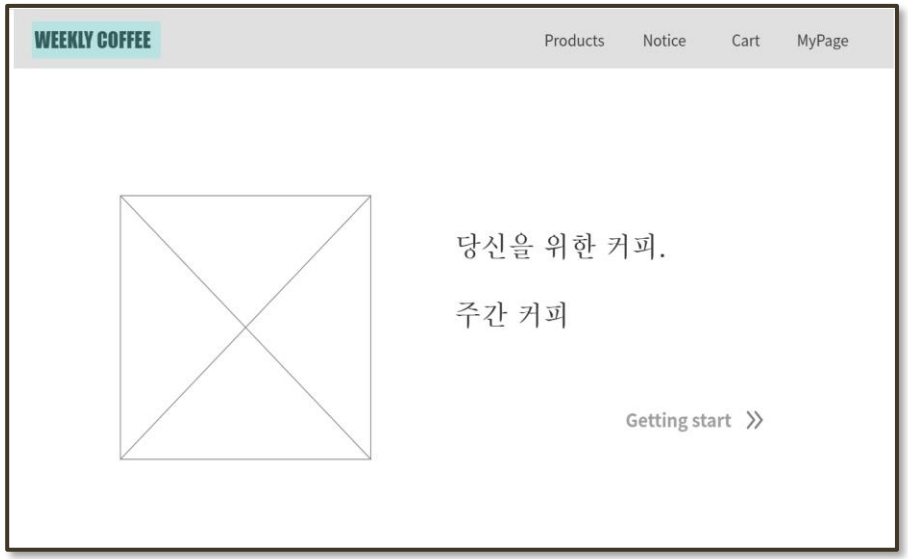
제공

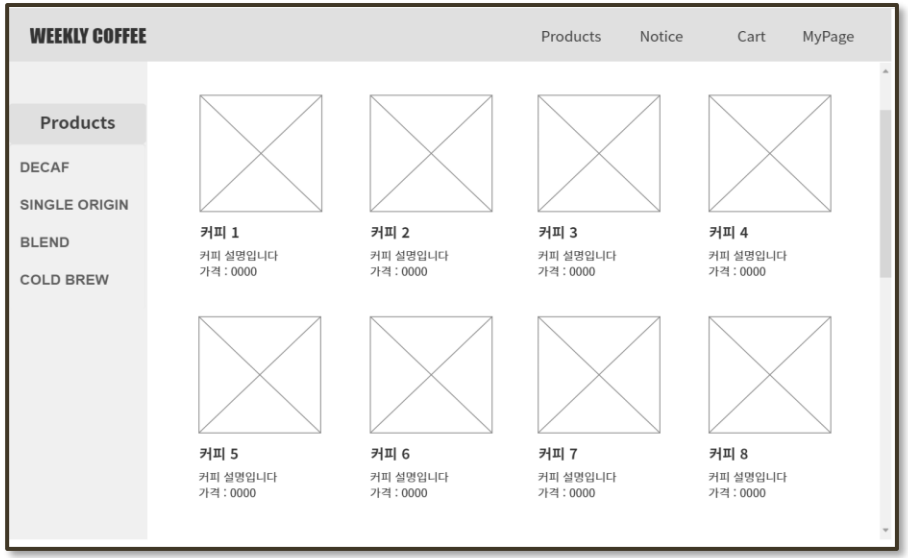
주문한 정보

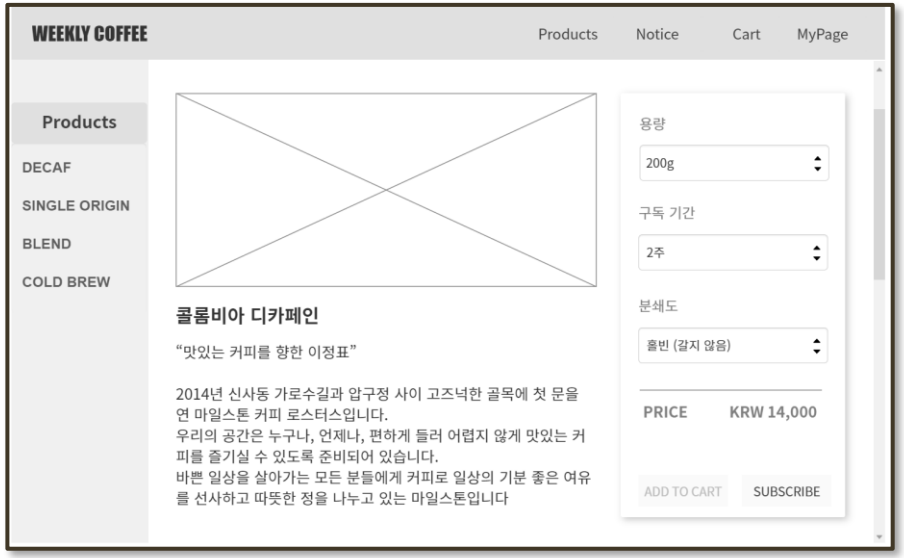
판매자

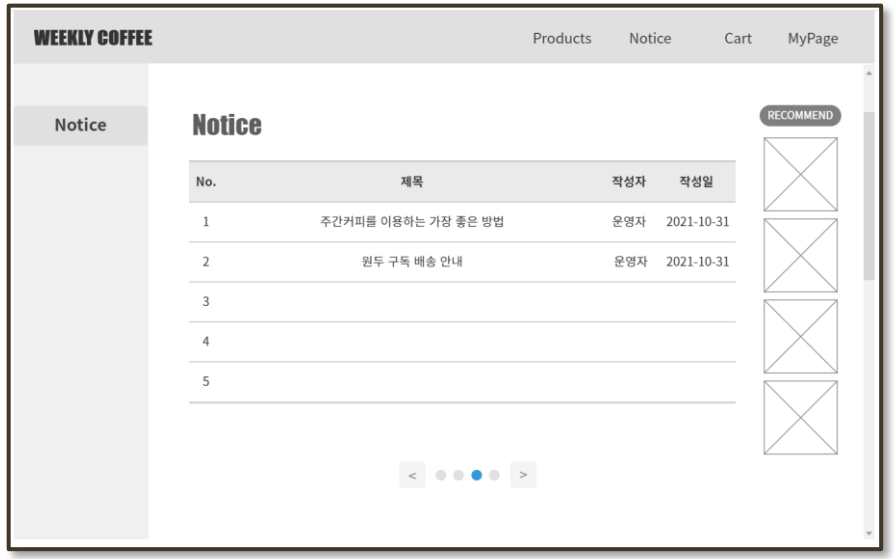
받음

- 카카오 오븐을 활용하여 주간커피 UI 프로토타입을 제작했습니다. (사진을 클릭하면 카카오 오븐 링크로 연결됩니다.)



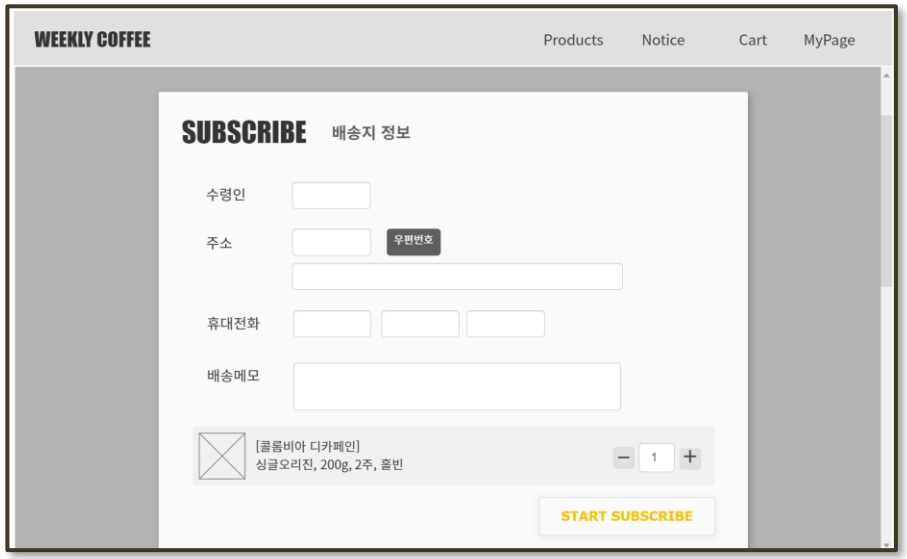



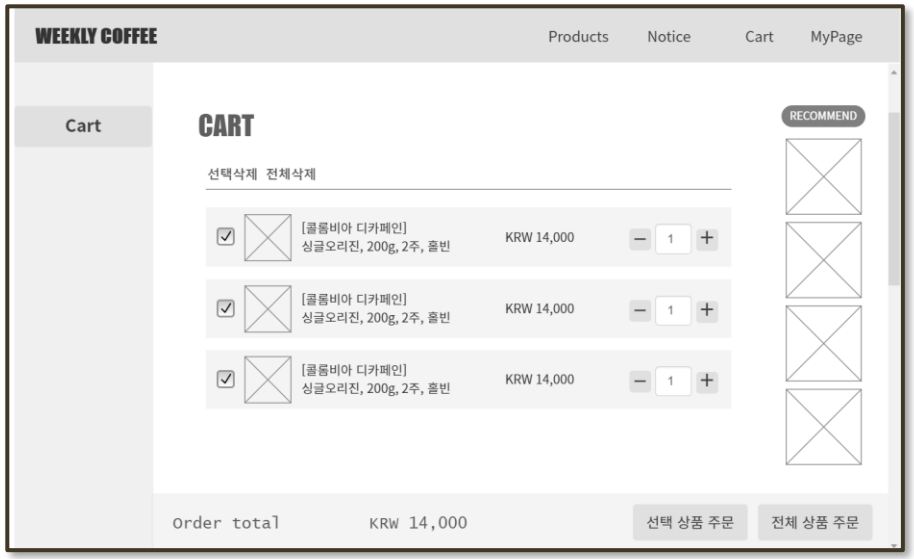



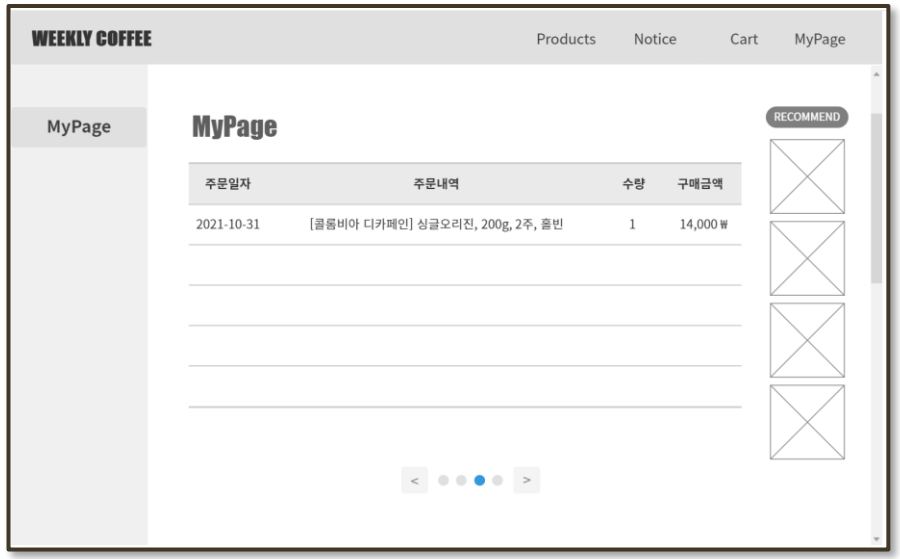



KAKAO OVEN

- 카카오 오븐을 활용하여 주간커피 UI 프로토타입을 제작했습니다. (사진을 클릭하면 카카오 오븐 링크로 연결됩니다.)

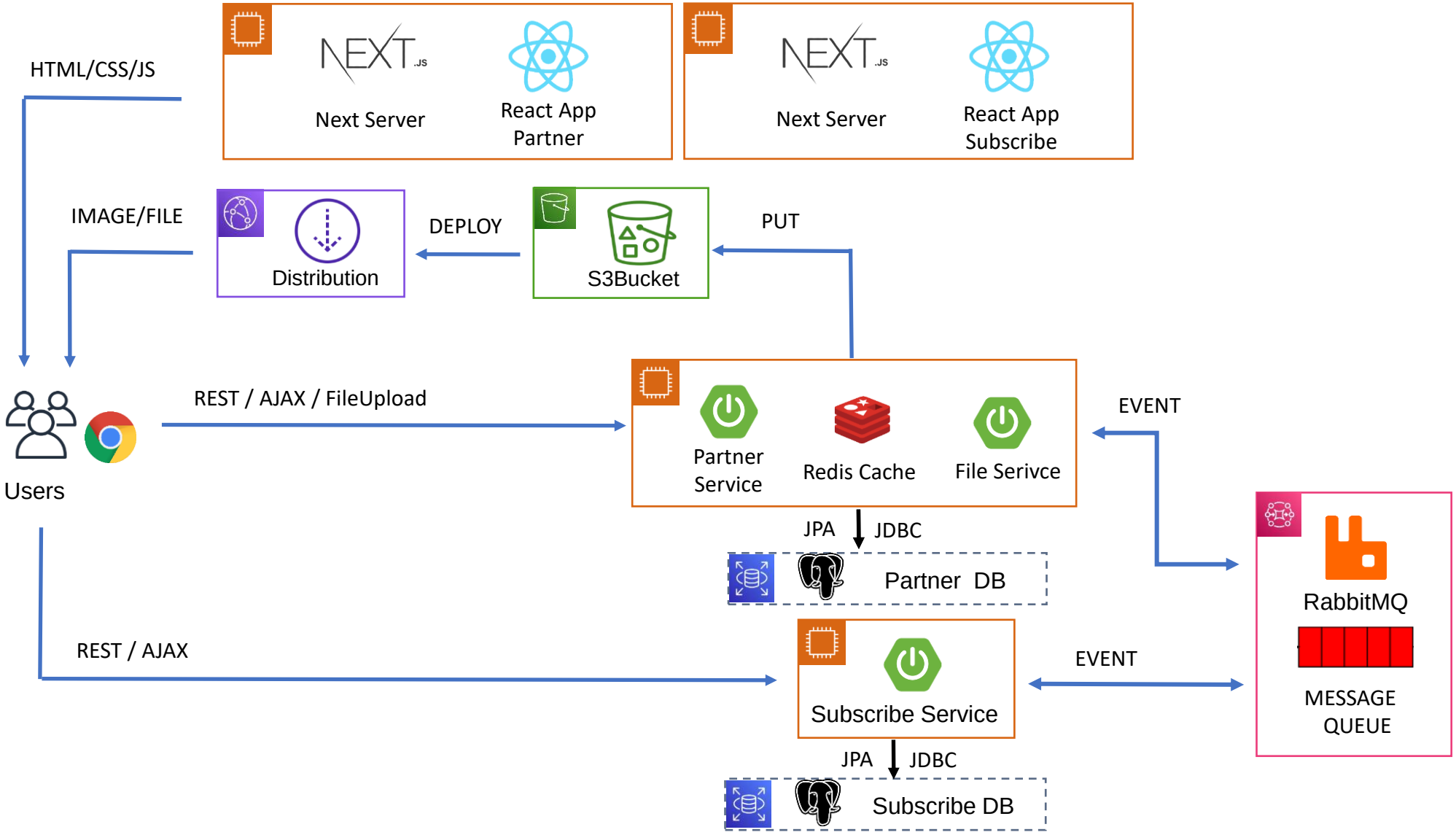


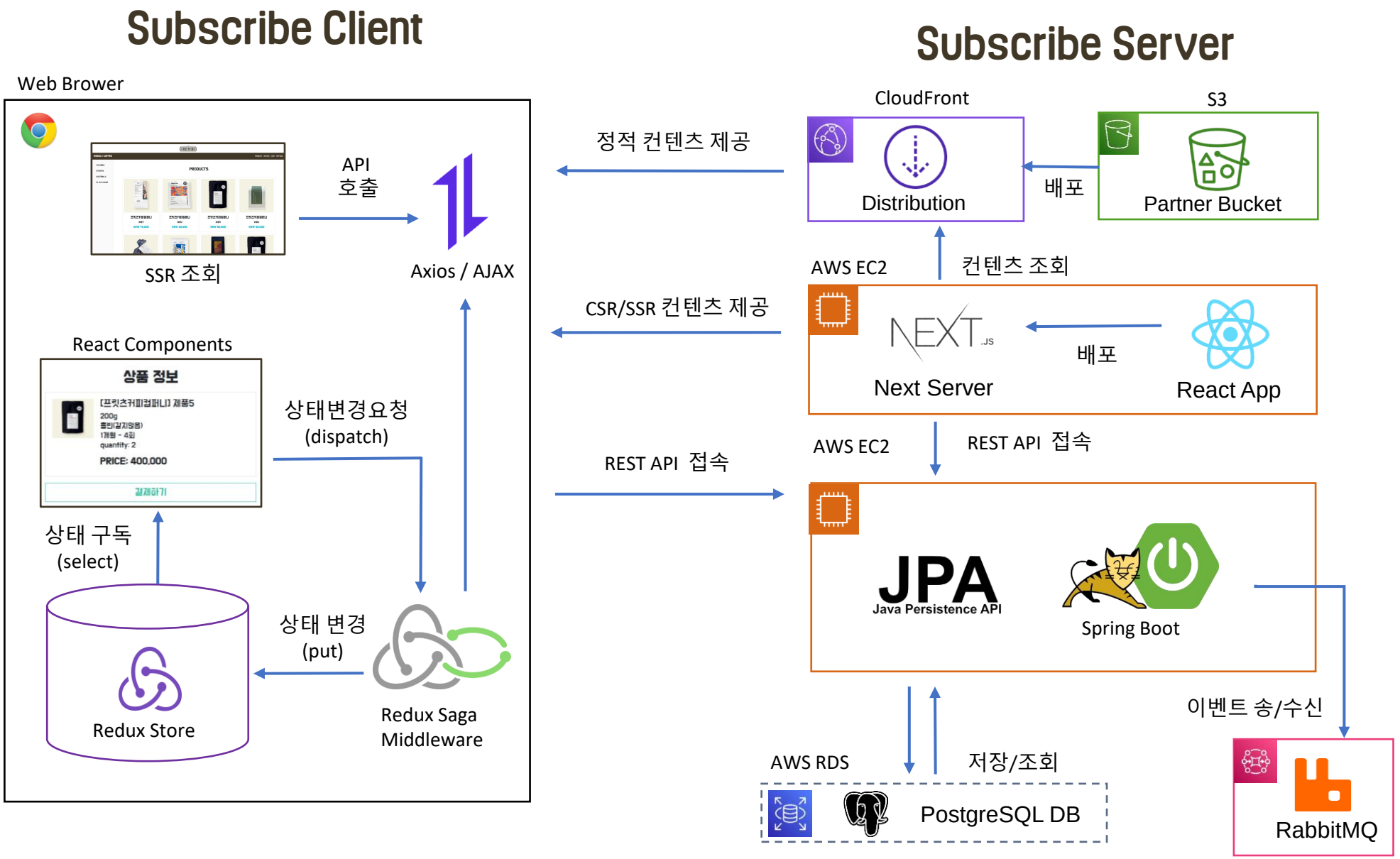




KAKAO OVEN

# 팀 아키텍쳐


HTML/CSS/JS
Next Server
React App
Partner
Next Server
React App
Subscribe
IMAGE/FILE
Distribution
DEPLOY
S3Bucket
PUT
Users
REST / AJAX / FileUpload
Partner
Service
Redis Cache
File Serivce
EVENT
JPA
JDBC
Partner DB
RabbitMQ
REST / AJAX
Subscribe Service
EVENT
MESSAGE
QUEUE
JPA
JDBC
Subscribe DB

# Subscribe 서비스 아키텍처



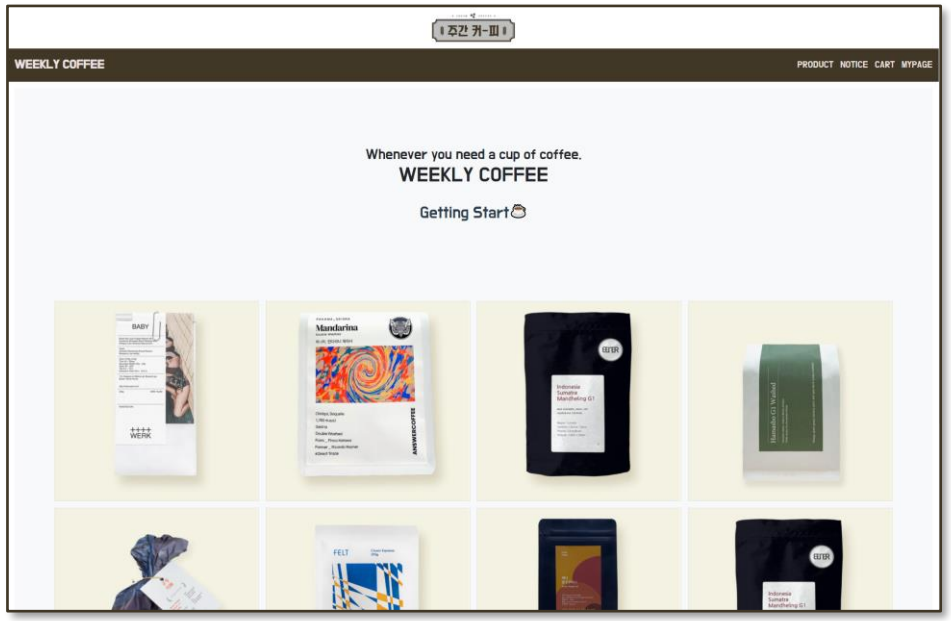
프로젝트 구현

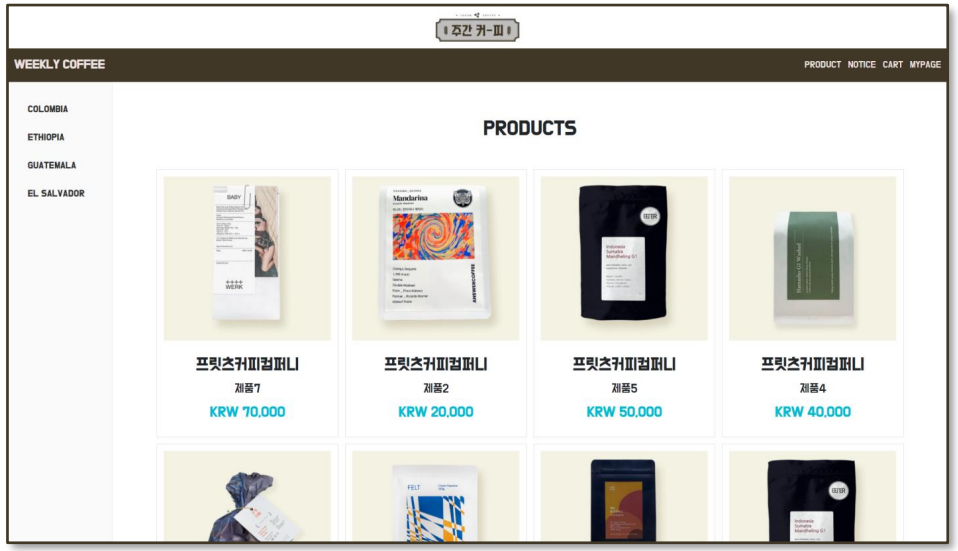
# Subscribe Client

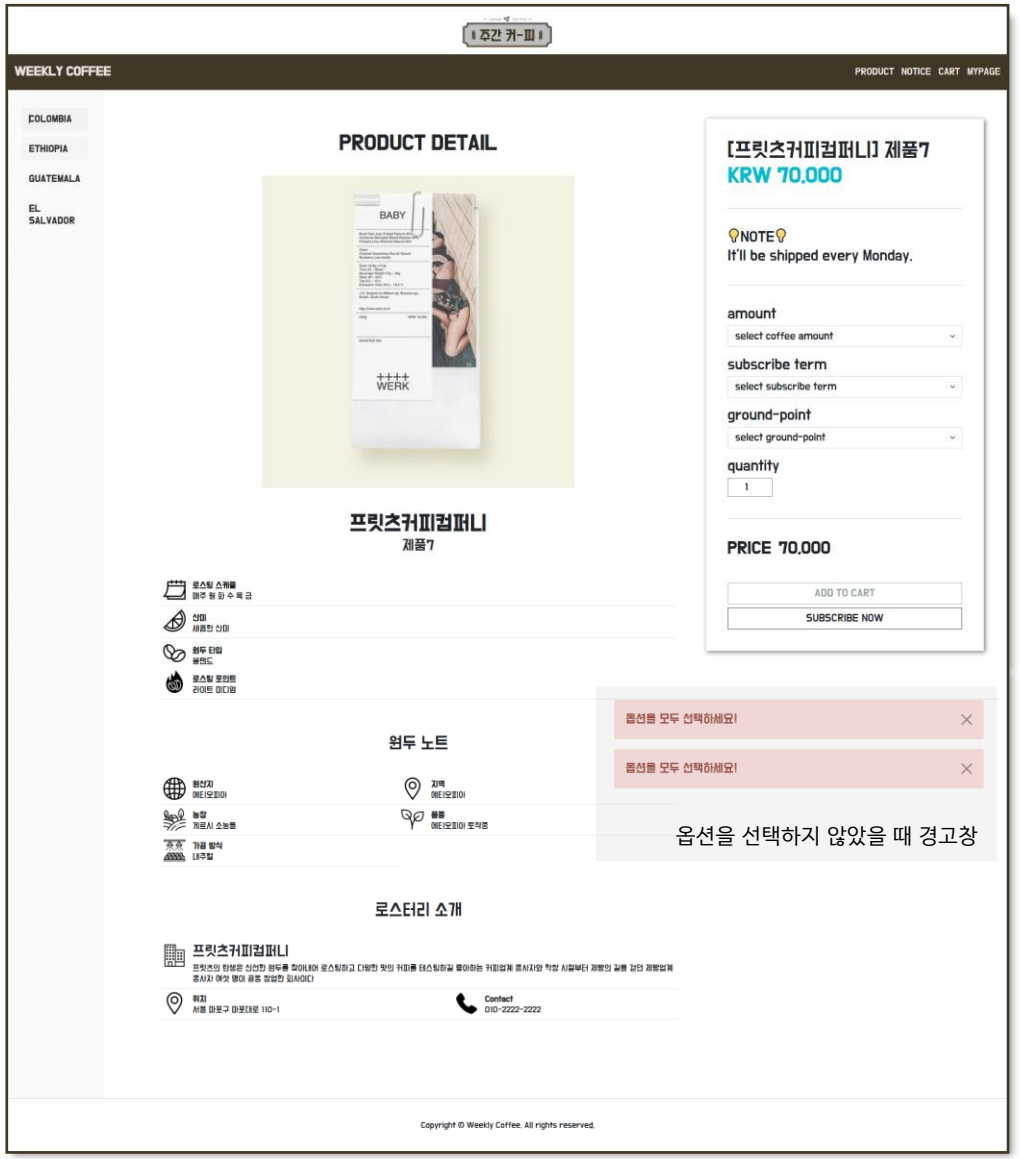
## UI style

## UI 구현

- UI 라이브러리를 사용하여 화면을 구현하였습니다.




옵션을 선택하지 않았을 때 경고창

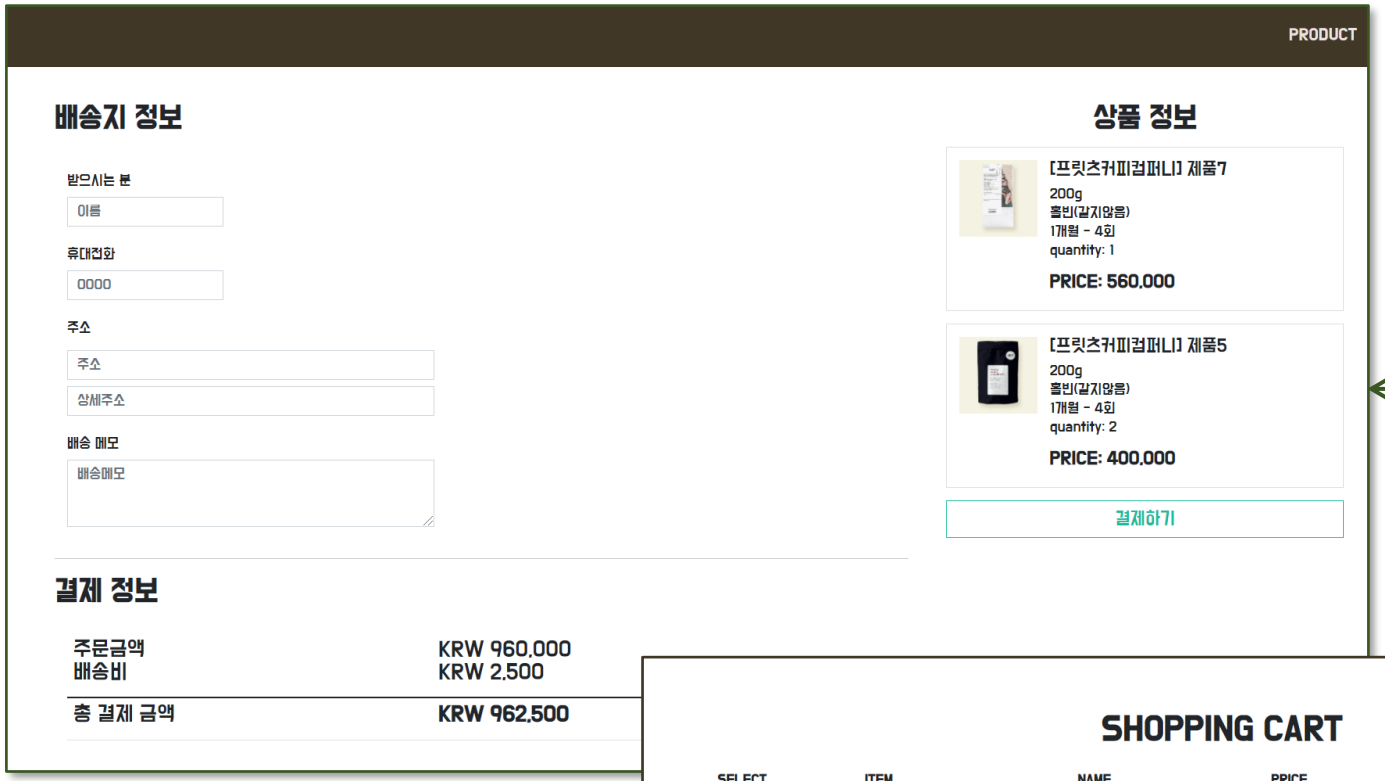
프로젝트 구현

Subscribe Client

UI style

# UI 구현

- UI 라이브러리를 사용하여 화면을 구현하였습니다.




클릭하면 주문 화면으로 이동하고

장바구니에 담긴 옵션을 우측에 보여줍니다.


*click!*

프로젝트 구현

Subscribe Client

UI style

# UI 구현

- UI 라이브러리를 사용하여 화면을 구현하였습니다.


*click!*

주문 한 건을 클릭하면

주문 상세 내역으로 이동합니다.


한 주문에 제품이 여럿이면 외 0건 으로 표시됩니다.

# Next.js 공통 컴포넌트 사용

Next.js
Layout


```tsx
function MyApp({ Component, pageProps }: AppProps) {
  return (
    <Layout>
      <Provider store={store}>
        <Component {...pageProps} />
      </Provider>
    </Layout>
  );
```

- 최상위 컴포넌트인 app.tsx에 Layout 컴포넌트를 import 해서 공통 컴포넌트로 사용하였습니다.
- Layout의 각 부분을 컴포넌트로 만들고 import 해서 레이아웃의 틀을 구현하였습니다.
- pageProps에 따라 application 페이지가 교체 됩니다. (Single-Page-Application)

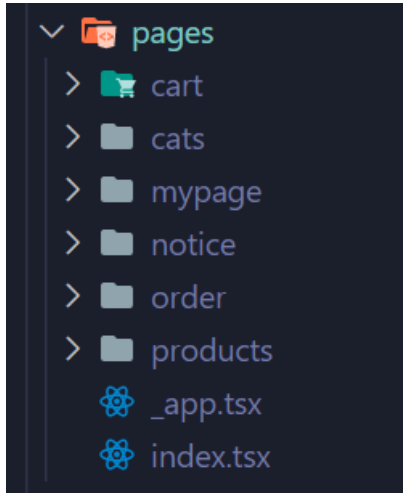
프로젝트 구현

Subscribe Client

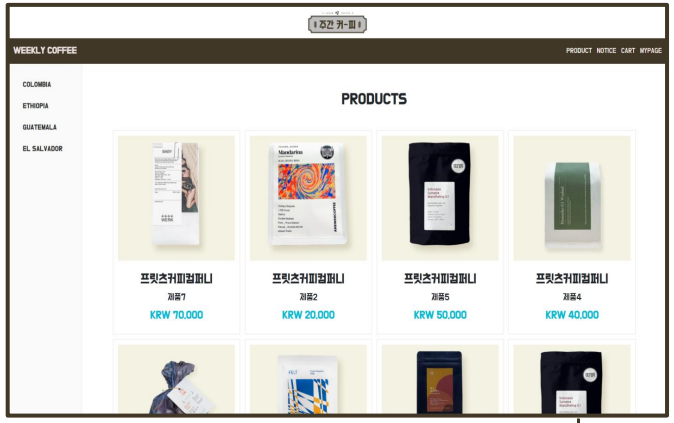
Next.js

Routing

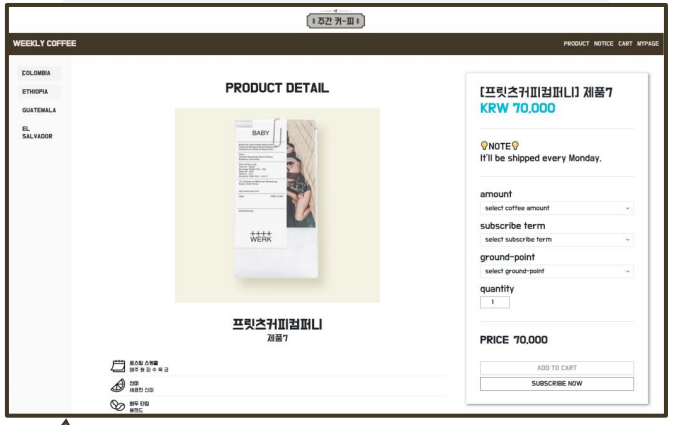
# Next.js 프레임워크의 파일 기반 Routing 사용 (컴포넌트 이동)




.com/products




.com/products/detail/2




Pages 폴더의 path 와 useRouter 로

컴포넌트 이동을 했습니다.

각 product의 id 값을 가지고

해당 product detail로 이동합니다

```
const router = useRouter();
```

```
onClick={() => {
  // id값을 물고 이동해야함
  router.push(`/products/detail/${item.productId}`);
}}
```

프로젝트 구현

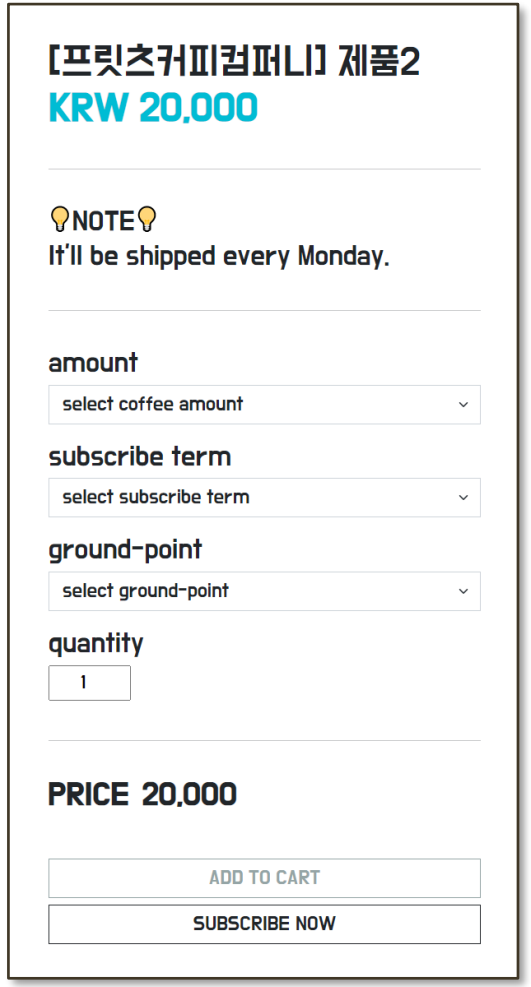
Subscribe Client

# React hooks 를 사용하여 함수형 컴포넌트를 구현했습니다.

## React Hooks

- useRef 코드와 화면에서 Ref input값을 받아오는 부분입니다.

```
const amountInput = useRef<HTMLSelectElement>(null);
const substermInput = useRef<HTMLSelectElement>(null);
const groundpointInput = useRef<HTMLSelectElement>(null);
const quantityInput = useRef<HTMLInputElement>(null);
```




- useEffect 에 dependency를 설정하여 데이터가 추가되어야 화면이 전환되도록 했습니다.

```
useEffect(() => {
  isAddcomplete && router.push("/mypage");
}, [isAddcomplete, router]);
```

- useState에 초기값을 설정해준 코드입니다.

```
const [addTotal, setAddTotal] = useState(0);
const [amount, setAmount] = useState("");
const [term, setTerm] = useState("");
```

프로젝트 구현

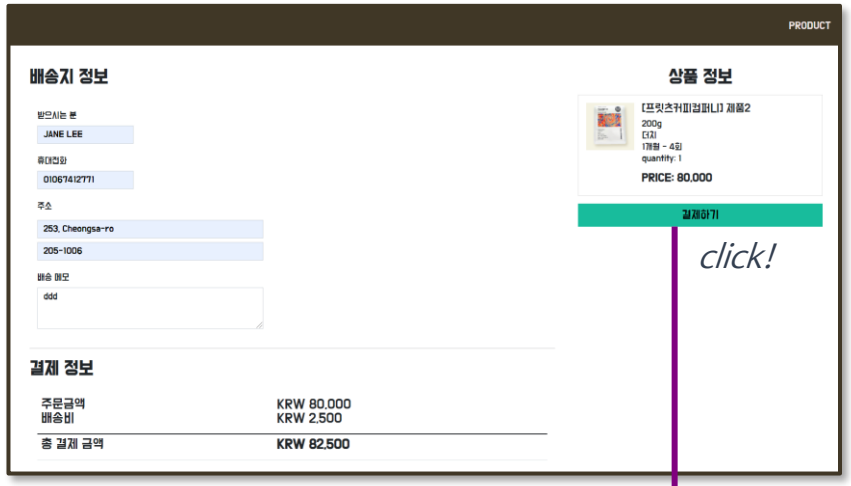
## Subscribe Client

Redux - toolkit

Redux - Saga

# Redux-toolkit을 사용해 글로벌 state를 관리 하였습니다.

Subscribe Component (주문 생성)




*click!*

Mypage Component (주문 조회)


Redux

useDispatch 사용 액션 생성
useSelector 사용 상태 조회

Redux-Saga

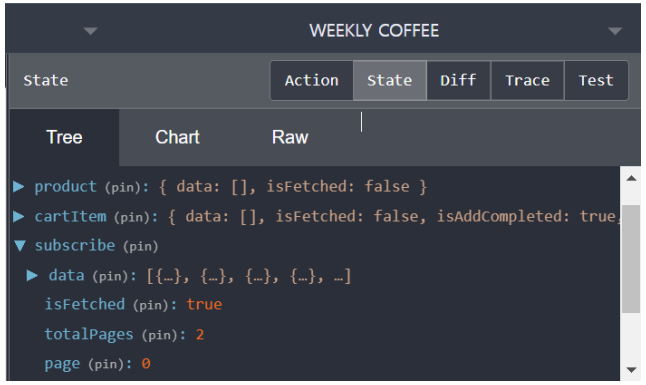
state change

API 호출

응답

Subscribe Server

- State에 주문이 추가됨

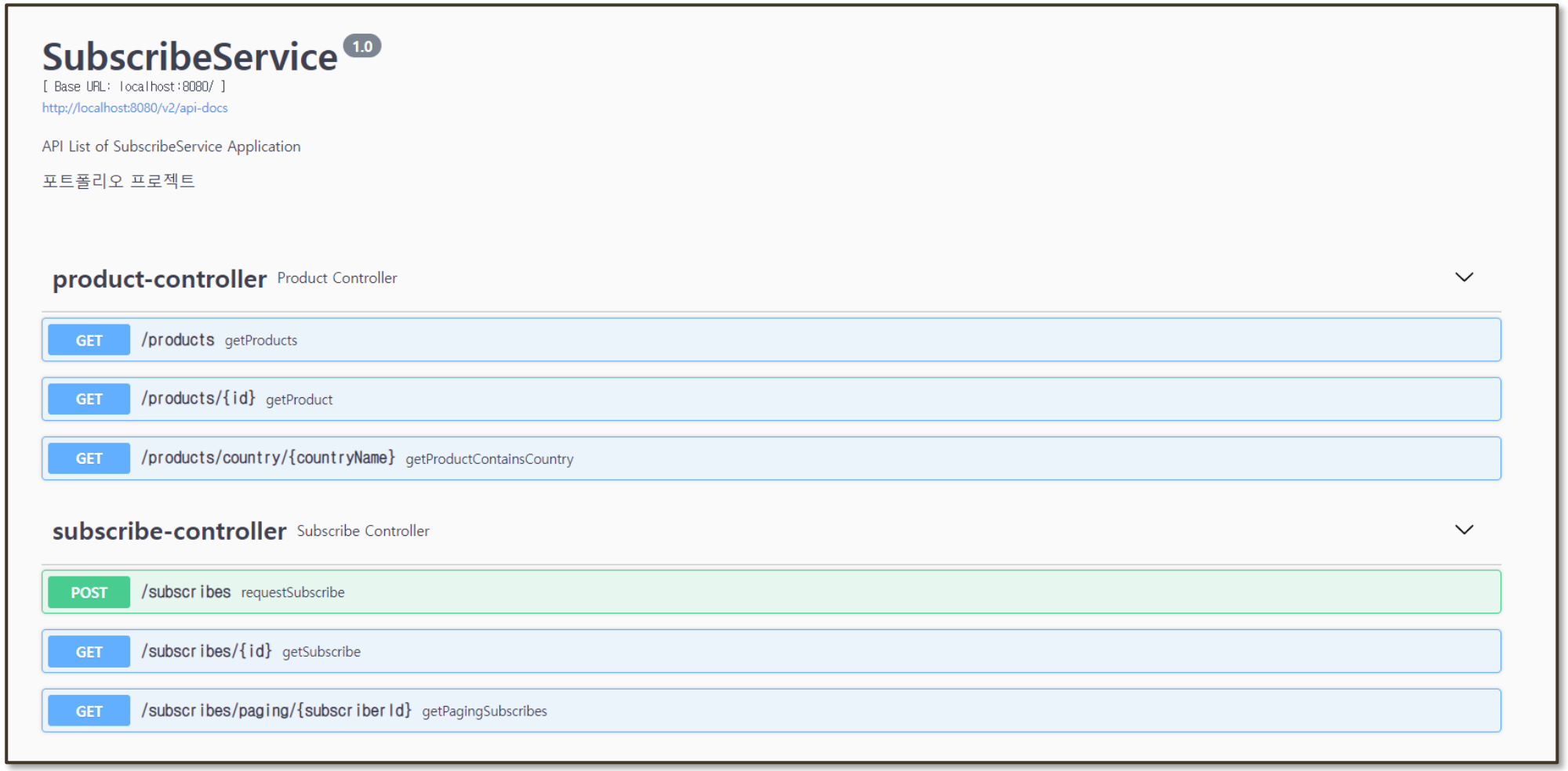


프로젝트 구현

Subscribe Server

# Swagger를 사용하여 서버의 API를 시각화 했습니다.

Swagger



# RabbitMQ를 활용해 서비스 간 데이터를 주고받도록 구현 했습니다.

판매 제품 메시지큐

Partner → Subscribe


동일한 제품 데이터를 다른 서버에서 출력합니다.

RabbitMQ

Partner Client

1. 제품 등록

Partner Data Table

2. 제품 데이터 저장

Subscribe Server

3.제품 데이터 송신

RabbitMQ

4. 제품 이벤트 수신

Subscribe Server

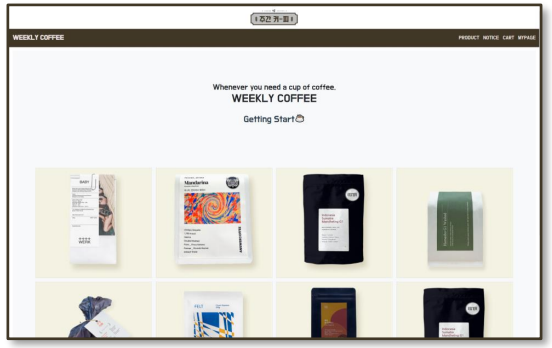
5. 제품 데이터 저장

2) 제품 데이터 조회

Subscribe Data Table

1) 데이터 요청

3) 데이터 응답




Subscribe Client

프로젝트 구현

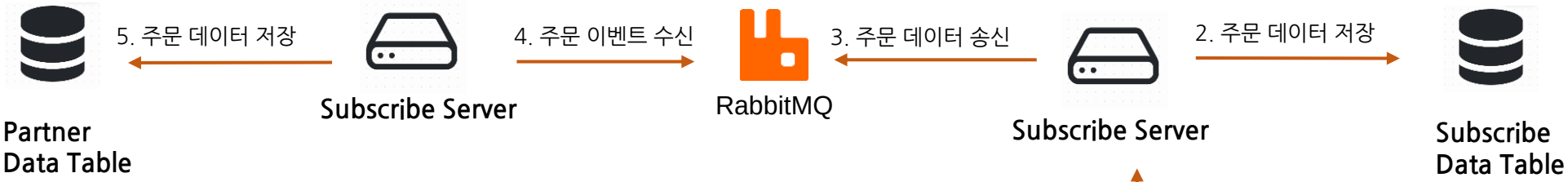
서비스 통합 구현

# RabbitMQ를 활용해 서비스 간 데이터를 주고받도록 구현 했습니다.

주문 메시지큐

Subscribe → Partner


동일한 주문 데이터를 다른 서버에서 출력합니다.





1. 주문 생성

Subscribe Client

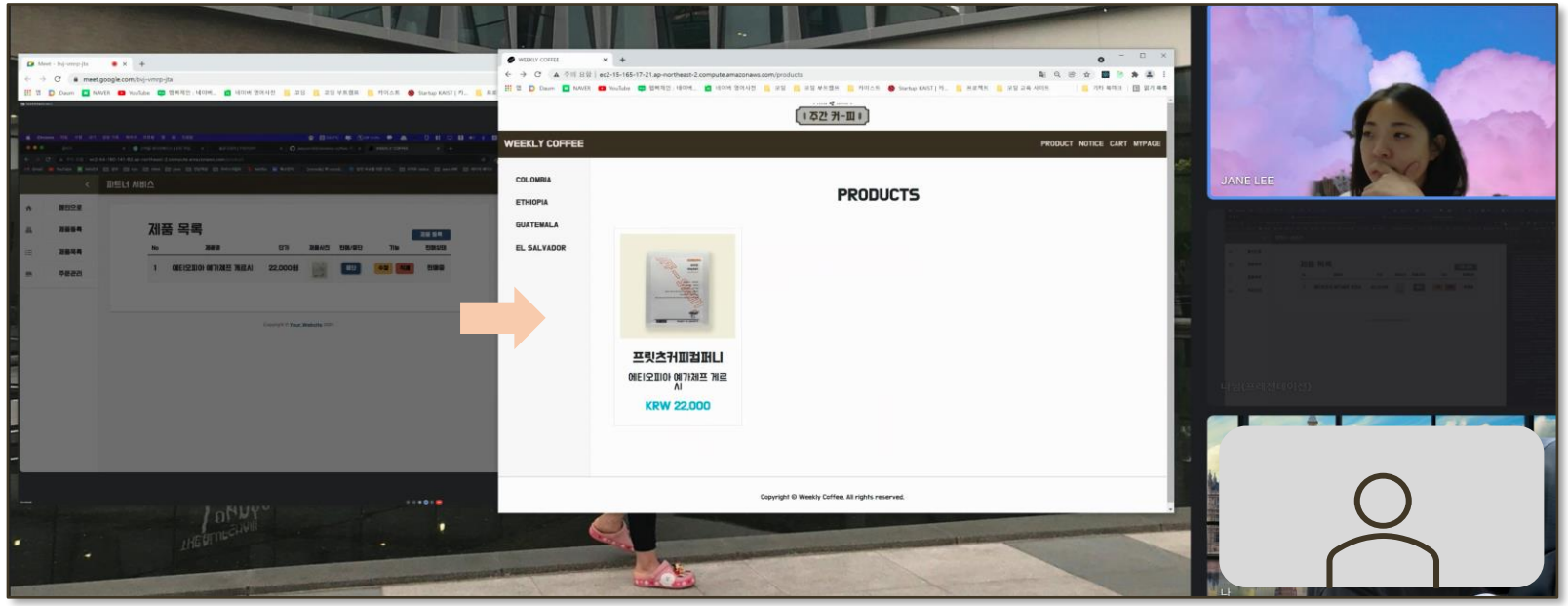
통합 운영 테스트

## About Us

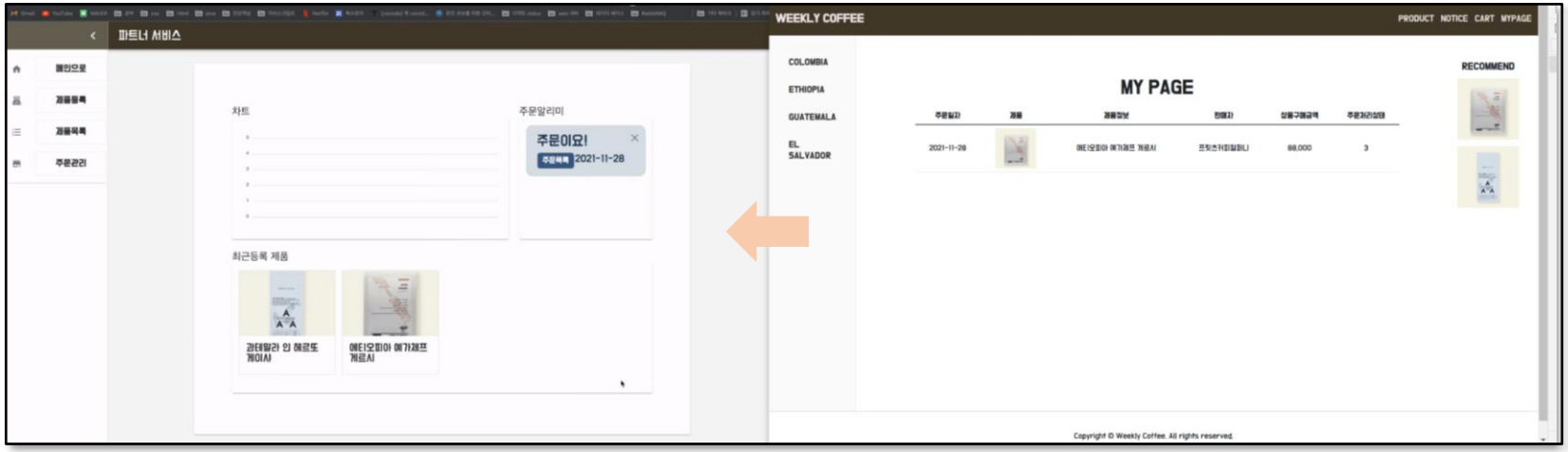
Google Meet

# 구글 밋업을 활용해 배포 서버에서 메시지 큐가 상대 서버에 잘 전달 되는지 테스트 하였습니다.

Partner → Subscribe



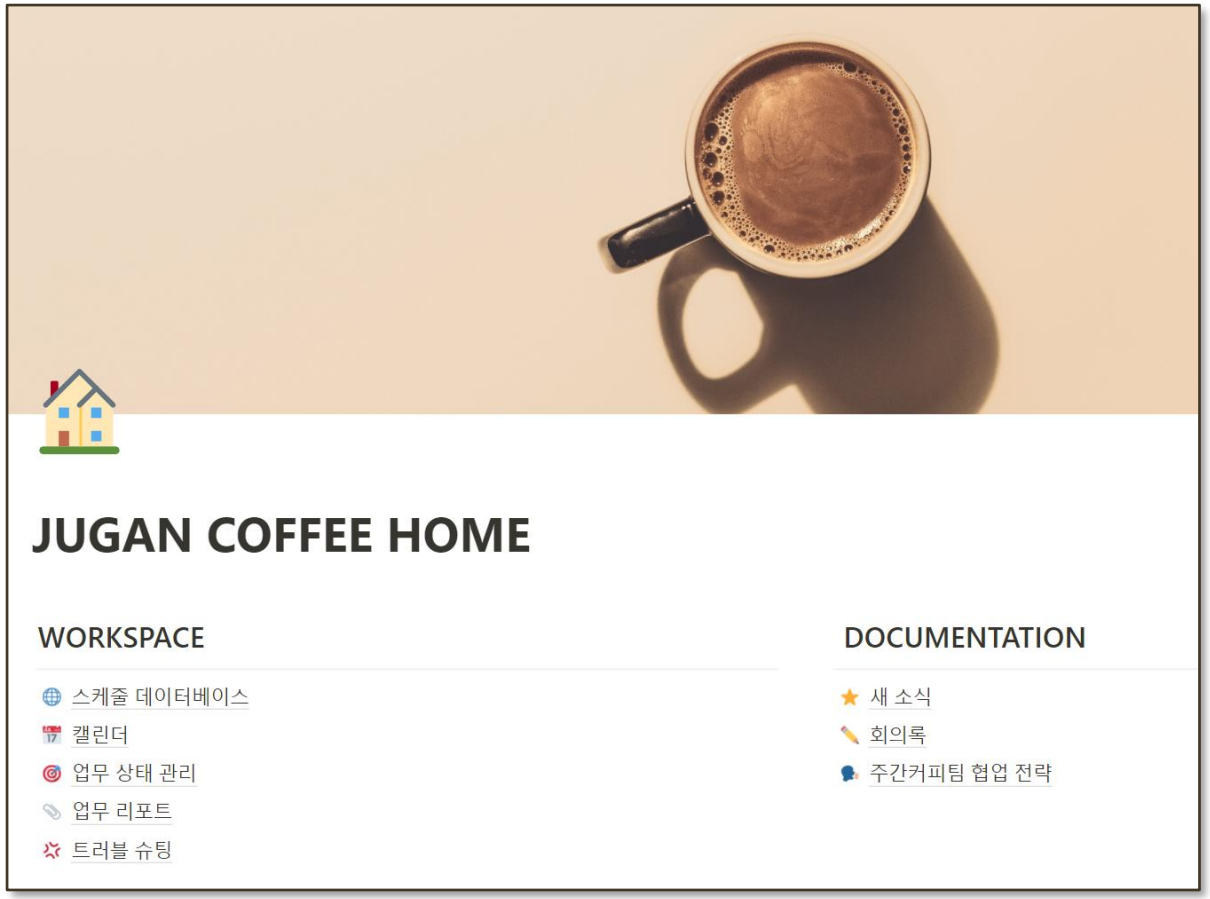

Subscribe → Partner

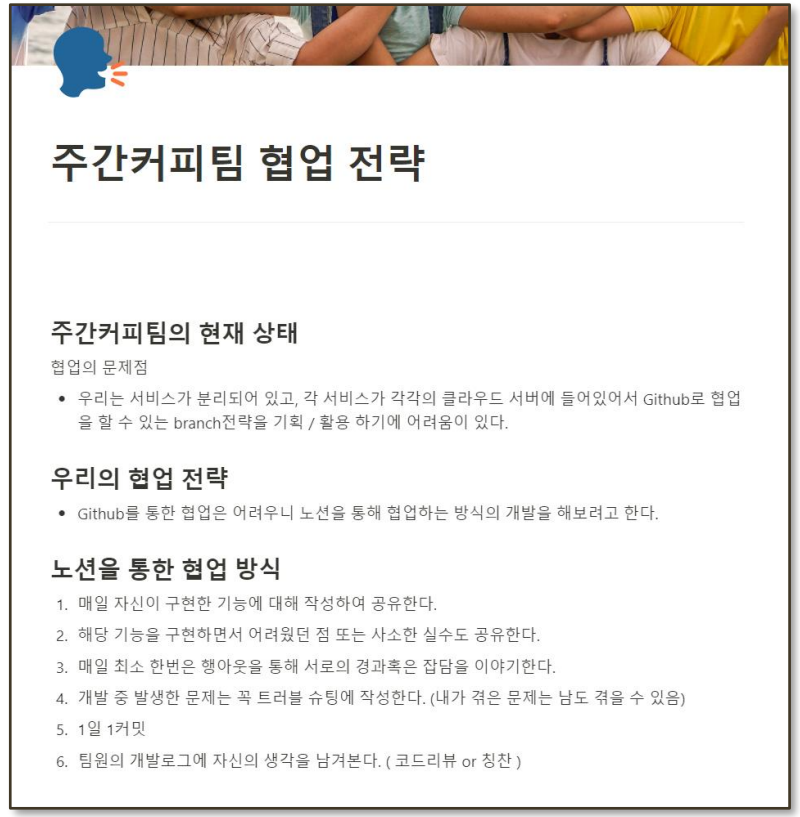


Notion

# Notion 협업 툴 활용, 프로젝트 진척도 관리

- 구성원이 적극적으로 Notion을 활용할 수 있도록 홈페이지를 작성하고 사용법을 공유했습니다.



프로젝트 협업

About Us

Notion

# Notion 협업 툴 활용, 프로젝트 진척도 관리

- 개발 중 겪은 문제을 기록하여 서로의 어려움과 해결책을 공유했습니다.
- Notion을 활용한 결과, 효율적으로 프로젝트 진척도 관리를 할 수 있었습니다.

## 트러블 슈팅

개발 중 겪은 문제를 공유하는 곳입니다.

(양식 )[이름] 트러블 슈팅 제목

(양식 )[이름] 트러블 슈팅 제목

2021-10-27

[주은] git add . 에러 (LF will be replaced by CRLF)

2021-11-1

[재윤] DB연동중 한 실수

[재윤] PostgreSQL예약어 관련 에러

[재윤] 중복 에러

[재윤] 객체 관계 맵핑

2021-11-2

[재윤] SQL Order키워드 예약어 관련 에러

[주은] Next.js & 타입스크립트 스타일 적용

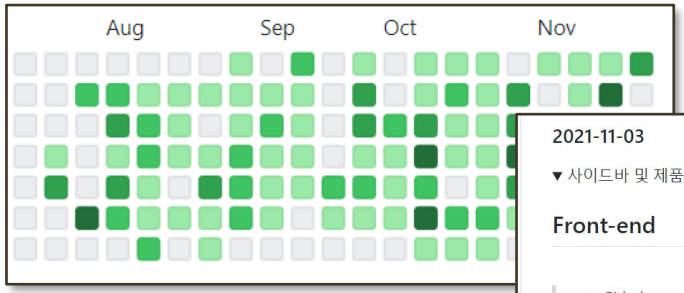
프로젝트 협업

About Us

Github

# Github repository의 Readme 파일 활용

- 마크업 문법으로 readme 파일을 작성하여 프로젝트 진행 과정을 기록했습니다.




2021-11-03

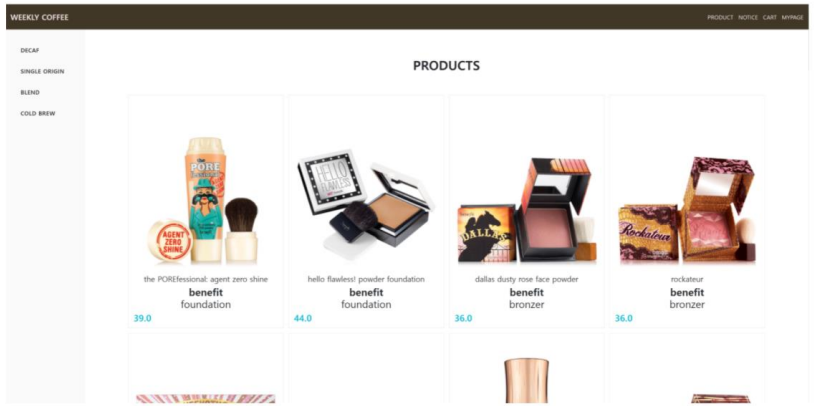
▼ 사이드바 및 제품목록 그리드 구현 [click]

## Front-end

> - Github commit
> https://github.com/happyjane210/weekly-coffee/commit/1912320b8a43f501b10bc21c8cd2af977acd5abb
> - Develop Issue
> https://www.notion.so/2021-11-03-d56fcedf92d142f9b61657e95a4ce045

### 진행 내용

- 왼편 사이드바를 구현했습니다.
- 각 컴포넌트에 import 했던 상단 nav바를 app.tsx 한 곳에 추가해 전역으로 적용되도록 변경하였습니다.
- 상품목록 (products)를 구현했습니다.
  - 그리드 구현 구현을 위해 화장품 오픈 API를 사용해서 제품 목록을 구현했습니다.
  - Next.js 에서 SSR(ServerSideRendering) 방식을 통해 오픈 API 데이터를 화면으로 가져오는 방법을 익혔습니다




### 참고 자료

- http://makeup-api.herokuapp.com/api/v1/products.json?brand=dior

README.md

## weekly-coffee 주간커피

여러 로스터리 사업장과 소비자를 연결하는,
원두 정기 구독 서비스를 제공합니다.

### 담당 개발 서비스 - 구독 서비스 (client)

다양한 원두를 소비자가 원하는 옵션에 맞춰 정기 구독 할 수 있는 구독 서비스를 개발 합니다

- 판매 제품 등록
- 주문 관리
- 마이페이지, 공지사항, 장바구니

### 개발 진행 기록

2021-11-01

▶ 개발 환경 구성 [click]

2021-11-02

▶ 홈화면 구현 [click]

2021-11-03

▶ 사이드바 및 제품목록 그리드 구현 [click]

★ JUGAN COFFEE ★

# PROJECT LINK BIO

https://github.com/happyjane210/weekly-coffee

https://www.notion.so/JUGAN-COFFEE-HOME-11c388513ab74fd4bda5e6478c22abc8

KAKAO OVEN

https://ovenapp.io/view/nEe8JaGEsCVCmiyLLlkkmkgamjo0M14R/

http://ec2-15-165-17-21.ap-northeast-2.compute.amazonaws.com/

★ JUGAN COFFEE ★

# 감사합니다