

# 우리샵

---

온라인 도소매업 사이트 프로젝트

---

남현우 2000

# CONTENTS

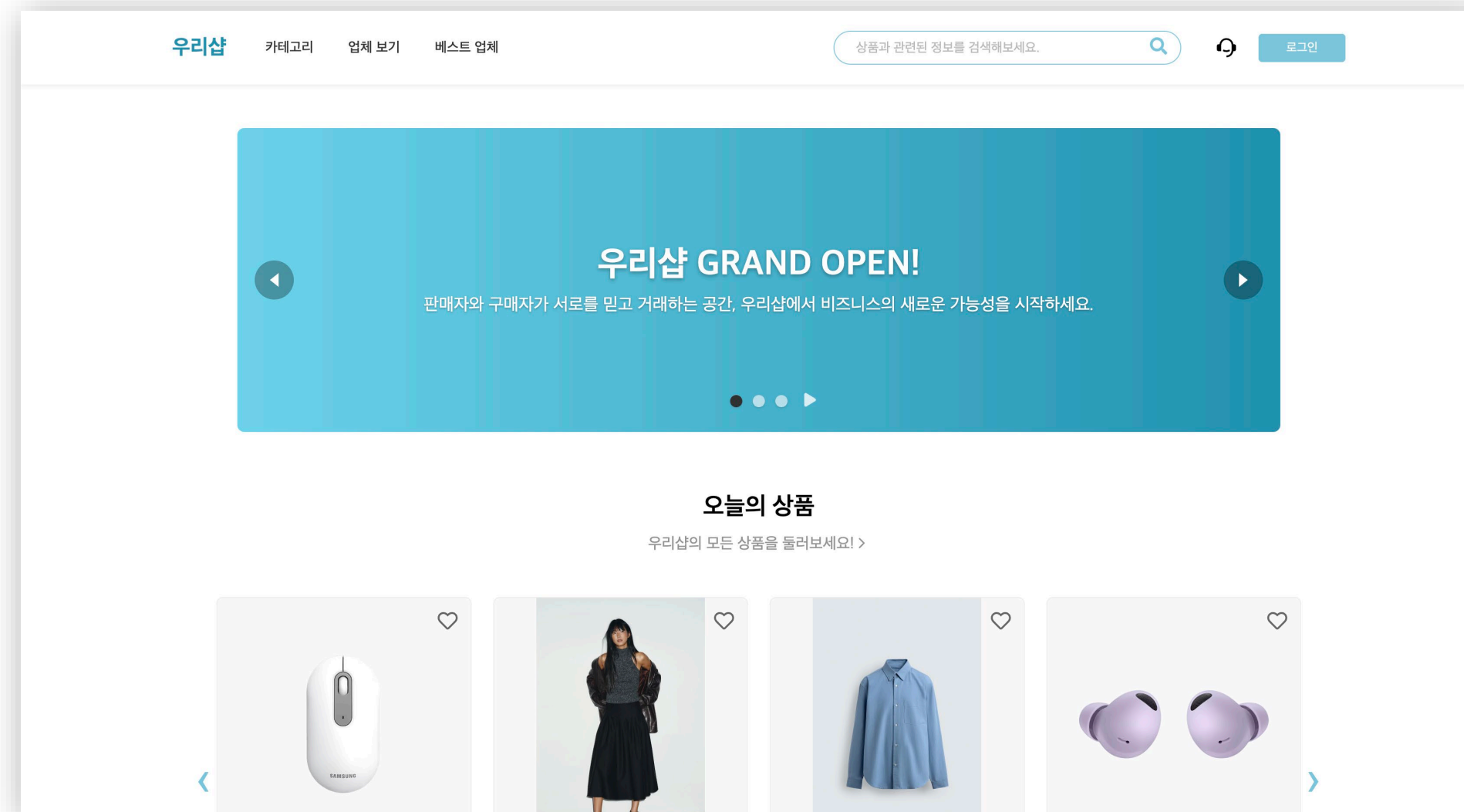
---

프로젝트 기획	진행 과정	기술 스펙	프로젝트 구조	프로젝트 설계	시현 및 회고
주제 및 배경 차별점 목적	역할 진행 일정	프론트엔드 백엔드 데이터베이스 서버	주요 페이지 주요 기능	데이터 베이스 설계 화면 설계	회고 시현

# 프로젝트 기획

---



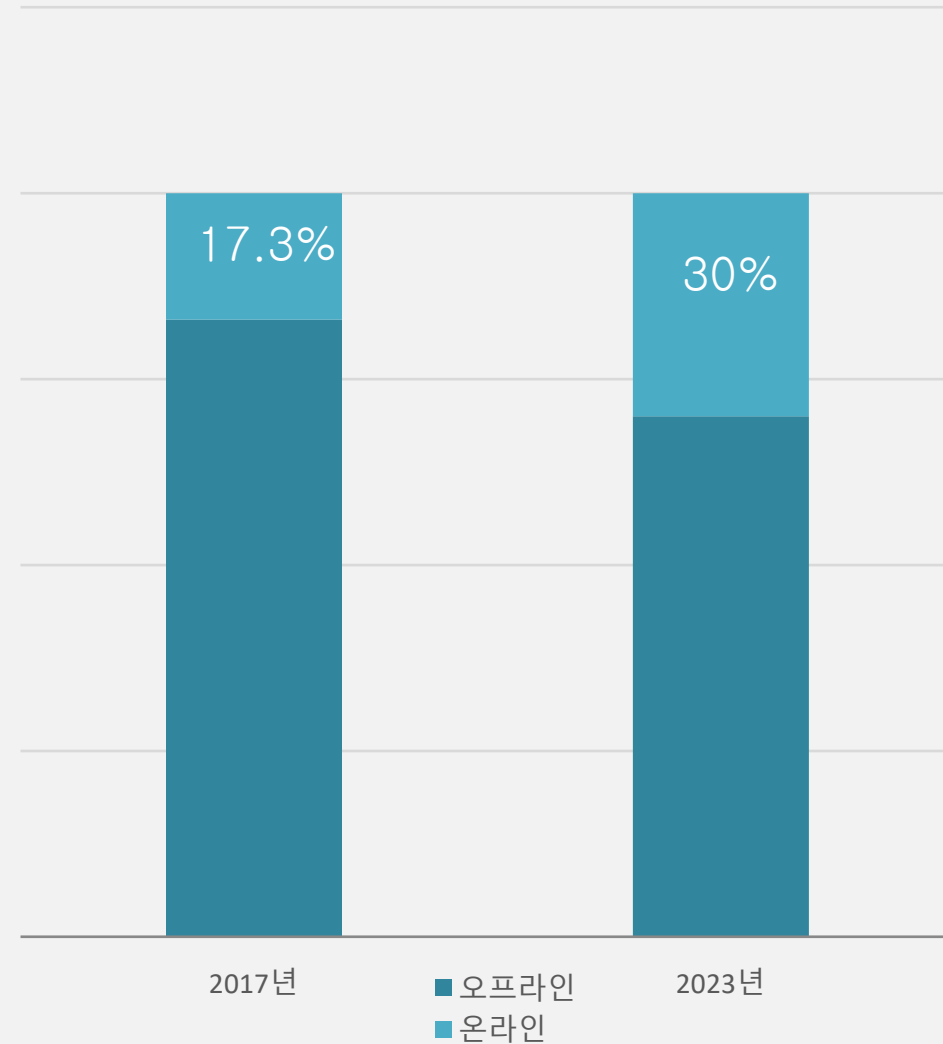


## 우리샵

### C2C 기반 도소매업 플랫폼

단순한 상품 나열을 넘어,  
업체 정보와 추천을 통한 새로운 도소매 생태계

도소매업 온오프라인 거래 비중



오프라인 중심 도·소매업은 임대료·인건비 부담

→ 온라인/모바일 거래 비중이 소매 판매의 약 30% 차지

하지만 소규모 업체는 온라인 입점/홍보 역량이 부족

→ “업체 단위의 온라인 플랫폼”의 필요성 ↑

“소상공인·자영업자 온라인 진입 필요성 증가”



## 신뢰하기 어려운 업체

단순 상품보다 “믿을 수 있는 업체” 인지 알고 싶어요.



## 업체 큐레이션에 대한 니즈

타플랫폼은 상품 나열과 가격 비교가 중심이라,  
사용자 맞춤 업체 추천 기능이 있으면 좋을 것 같아요.



## 업체 브랜드화 & 지속적 관계 구축에 대한 니즈

단골을 만들고 싶은데, 타플랫폼에서는 상품 위주의 정보만 제공하니  
저희 업체와 고객간의 지속적인 관계를 구축하기에는 한계가 있어요.

“업체 단위 정보 열람 + 업체 추천 시스템에 대한 니즈 증가”

## 경쟁 플랫폼

### 상품 개별 단위 나열

상품 정보는 알기 쉽지만  
해당 상품을 판매하는 업체에 대해서 알기 어려움

### 업체 추천 기능 X

도소매 사이트에서는 사용자별 맞춤 추천 기능 없음  
있더라도 광고 기반 노출인 경우 다수

## 우리샵 플랫폼

### 업체 단위

업체 단위로 열람이 가능하고 해당 업체가 판매하는 상품리스트 제공  
업체와 구매자 간의 지소적인 거래 관계 구축 가능

### 업체 추천 기능 O

사용자 니즈에 맞는 업체 추천으로 고객 경험 개선



소상공인·자영업자의  
온라인 진입 지원



상세 업체 정보를 제공  
하여 신뢰할 수 있는 거  
래 환경 조성



업체별 상품 리스트와 프  
로필을 제공하여 장기적  
거래 관계 구축 가능



플랫폼은 소상공인 매출  
증대와 소비자 선택권 확  
대라는 사회적 가치 실현



# 진행 과정

---





남현우

**데이터베이스 설계**

**백앤드**

Auth / User API

Buyer / Seller / Admin API

FAQ / 광고 API



박OO

**백앤드**

Product API

Category API

Review API

Product Image API



오OO

**프론트앤드**

메인페이지

상품 리스트 페이지

상품 상세페이지

**백앤드**

Cart API

Order API

Inquiry(QnA) API

Delivery API



추OO

**프론트앤드**

로그인 페이지

회원가입 페이지

업체 리스트 페이지

업체 상세 페이지

장바구니 페이지

결제 페이지

관리자 페이지

마이페이지(구매자)

마이페이지(판매자)



남현우

**데이터베이스 설계**

**백앤드**

- 로그인 api
- Buyer / Seller / Admin API
- FAQ / 구매자 문의 / 판매자 문의
- 카테고리
- 장바구니
- 주문 / 리뷰 / 좋아요
- 상품 / 상품 상세 / 상품 이미지



추OO

**프론트앤드**

- |            |            |
|------------|------------|
| 로그인 페이지    | 상품 상세 페이지  |
| 회원가입 페이지   | 장바구니 페이지   |
| 메인 페이지     | 결제 페이지     |
| 업체 리스트 페이지 | 관리자 페이지    |
| 업체 상세 페이지  | 마이페이지(구매자) |
| 상품 리스트 페이지 | 마이페이지(판매자) |



# 기술 스펙





## Frontend

React  
Axios  
Zustand+Immer+persist  
React Query  
CSS, CSS Modules  
React Hook Form,  
React-Quill-New  
React-daum-postcode



## Backend

Spring Boot  
Java  
Spring Security + JWT  
JPA (Hibernate)  
MariaDB Gradle  
REST API  
Lombok



## Database

MariaDB



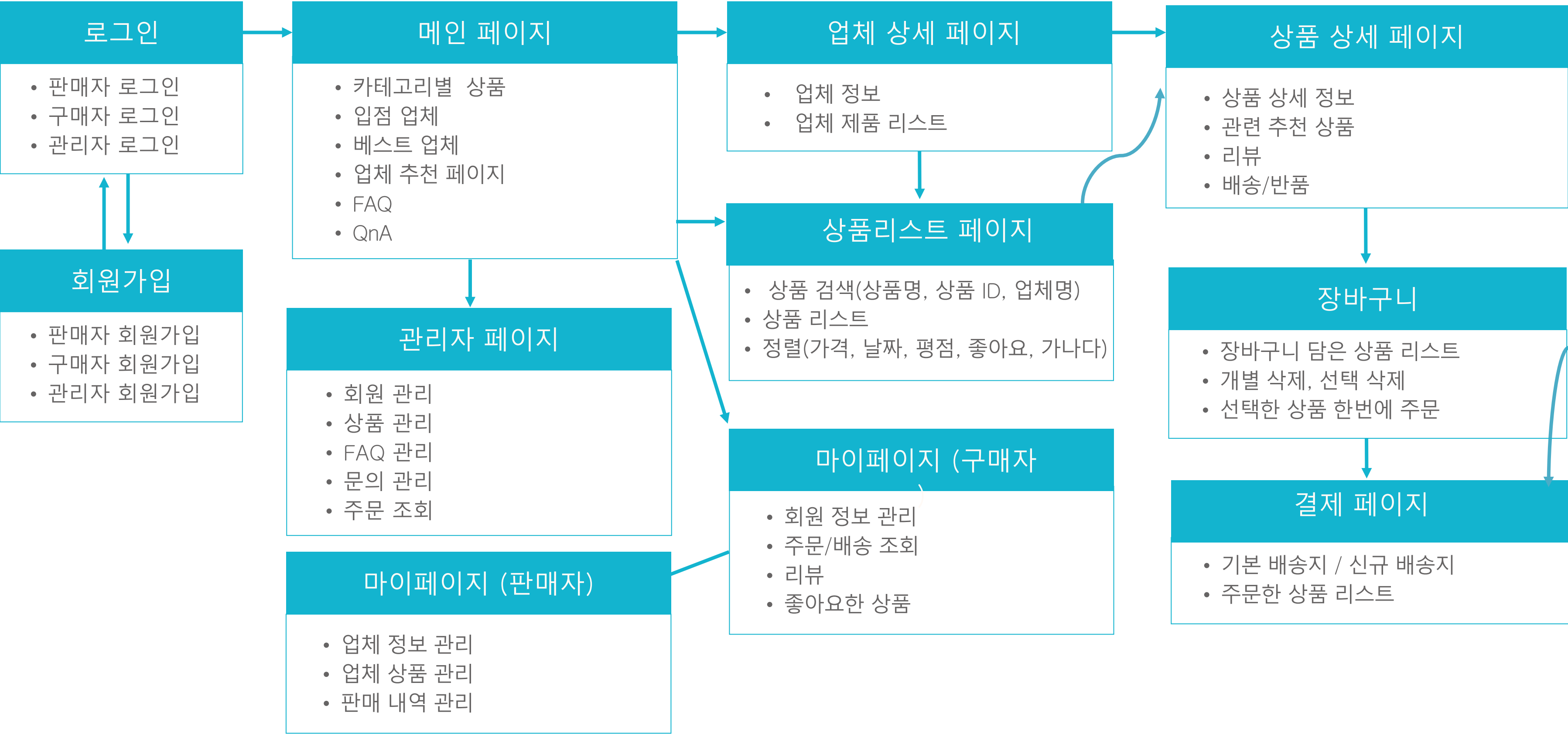
## Server

AWS

# 프로젝트 구조

---









### 공통 기능

- 로그인 / 로그아웃
- 회원가입/ 탈퇴/ 정보수정 (판매자/ 구매자)
- 상품 및 업체 검색
- FAQ 조회
- QnA 등록/수정/삭제/조회 (판매자/구매자)



### 관리자 기능

- 사용자 정보 조회/수정/삭제
- 상품 정보 조회/삭제
- 문의 답변 등록
- 판매자 판매인증 처리
- FAQ 등록
- 회원 거래 조회



### 판매자 기능

- 상품 등록/수정/삭제
- 배송 상태 수정 (결제/취소/배송중/배송완료)



### 쇼핑 기능

- 정렬
- 페이지네이션
- 필터 및 검색 결과 내 재 검색
- 관련 상품 추천
- 카테고리별 상품 조회
- 리뷰 평점
- 좋아요 수



### 구매자 기능

- 상품 좋아요/좋아요 삭제
- 상품 장바구니 추가/삭제
- 주문
- 리뷰 등록/수정/삭제



### 업체 보기 기능

- 정렬
- 페이지네이션
- 필터 및 검색 결과 내 재 검색
- 상품 좋아요 기반 업체 추천
- 업체 전체 상품 리뷰 평점
- 업체 전체 상품 좋아요 평점
- 판매 인증 마크 표시

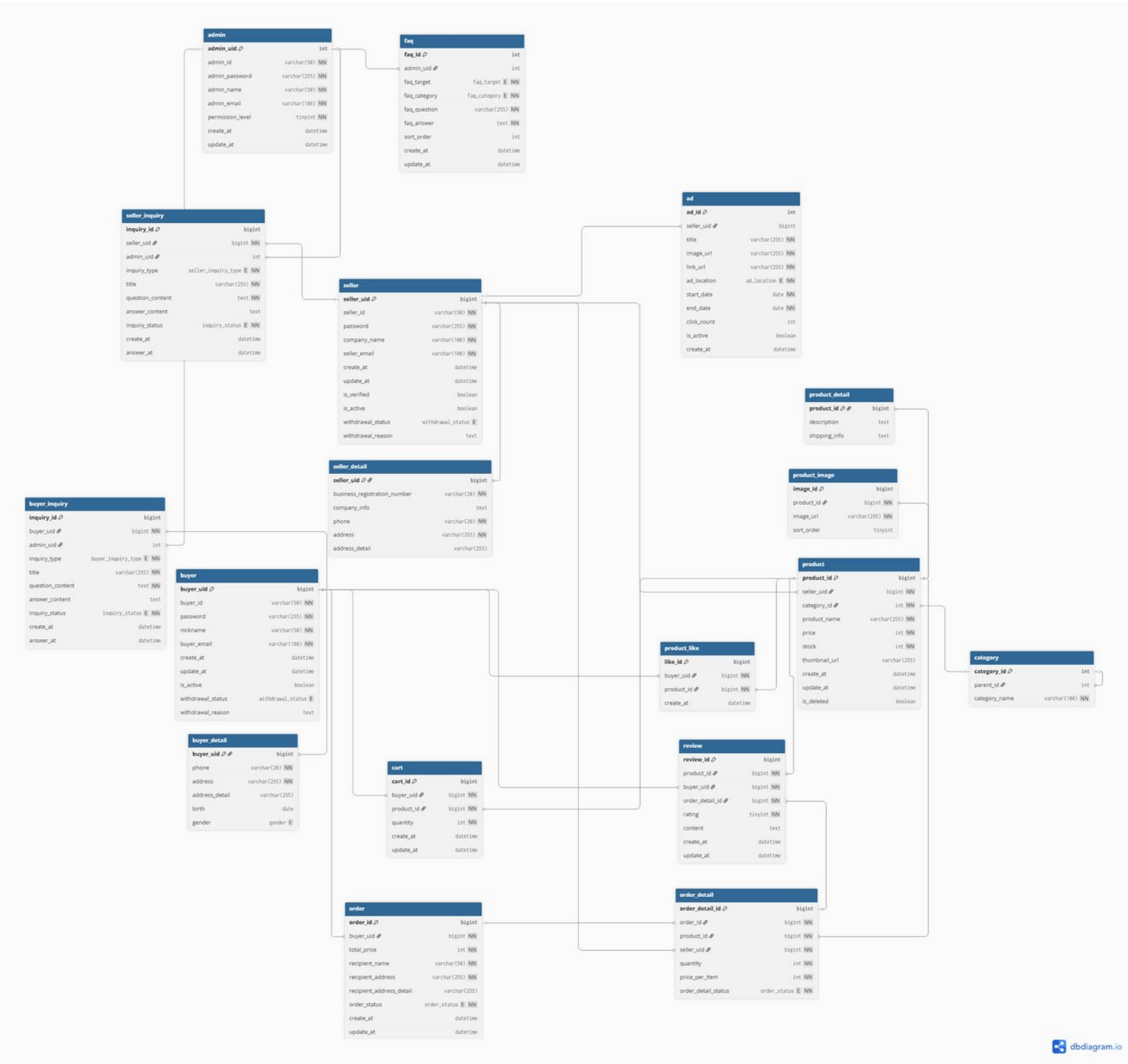
# 프로젝트 설계

---



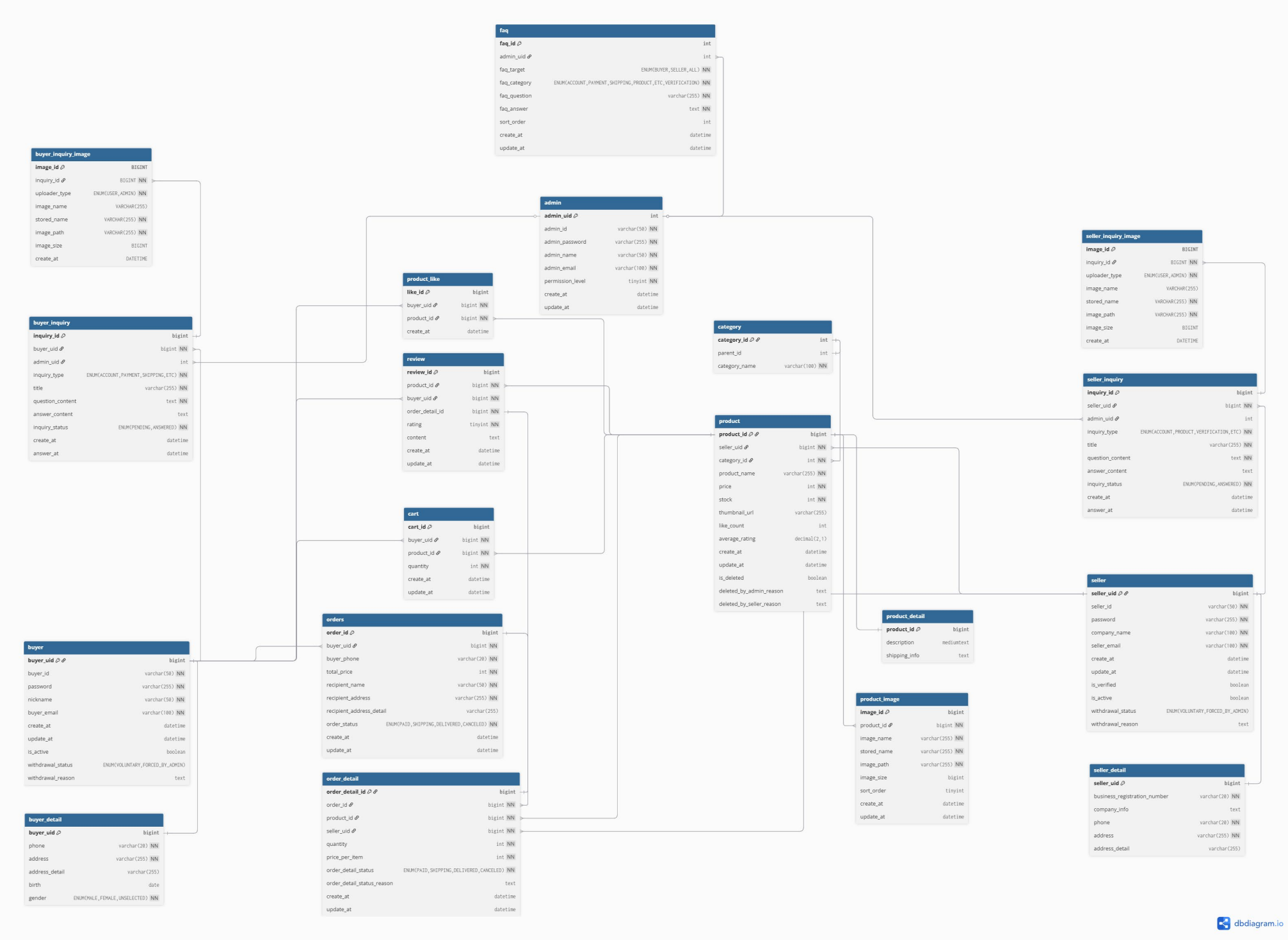
# 데이터 베이스 설계 (ERD) AS-IS

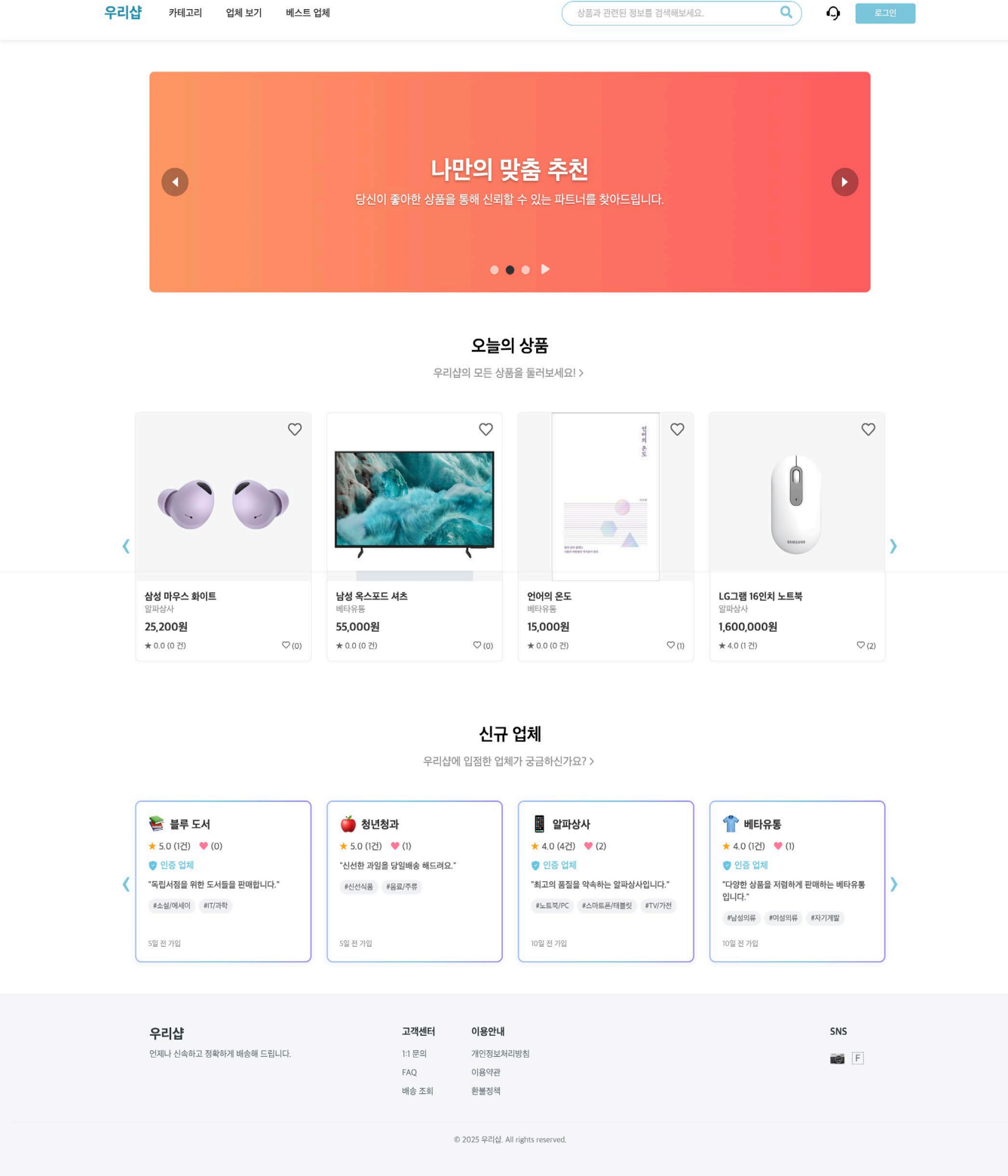
프로젝트 설계



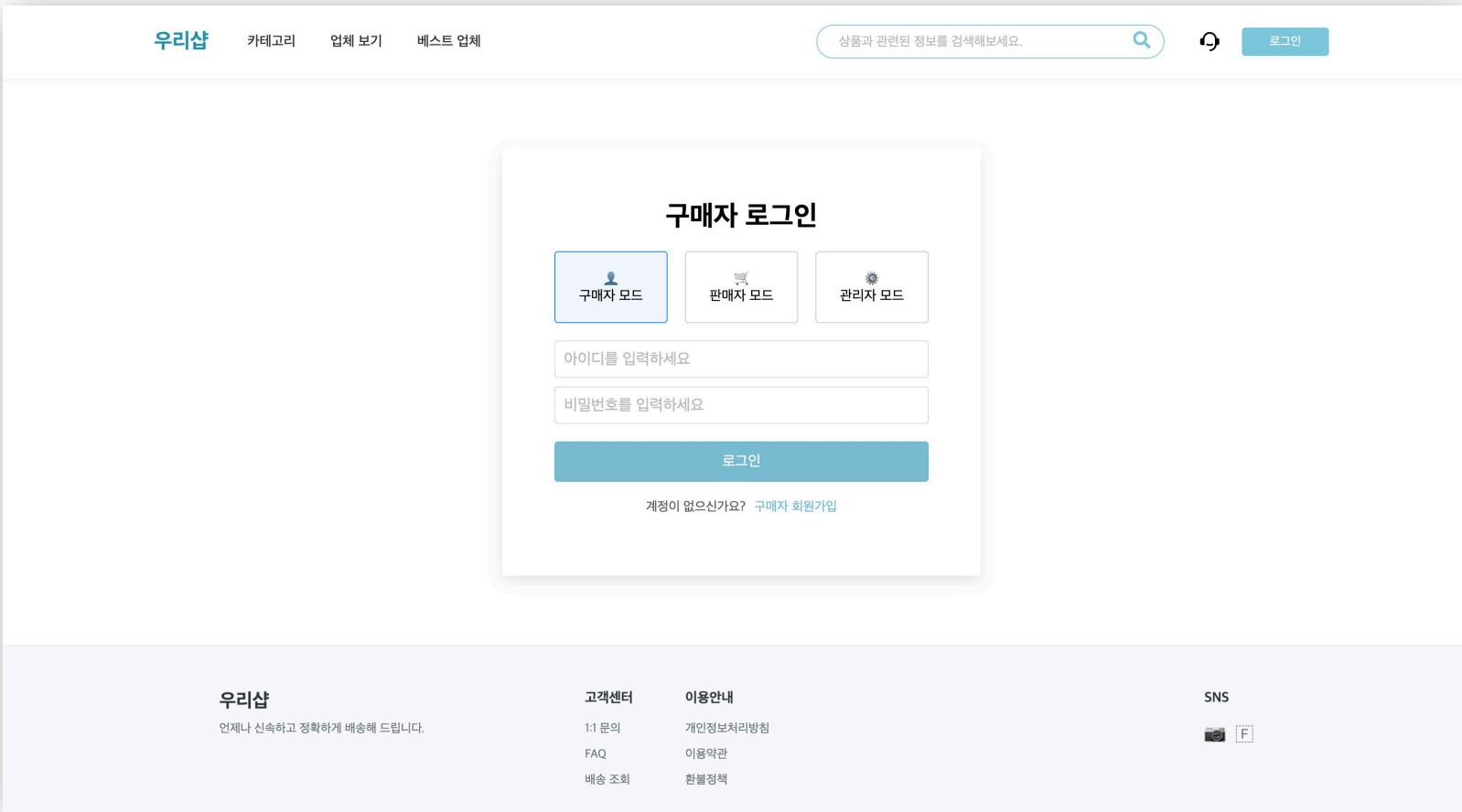
# 데이터 베이스 설계 (ERD) TO-BE

프로젝트 설계





번호	기능
1	[검색] `GET /api/v1/product/list` API가 `productName` 등의 검색 파라미터와 함께 호출 1. 사용자가 입력한 검색어의 양쪽 끝에 있는 불필요한 공백을 제거 공백을 기준으로 검색어를 분리하여 여러 개의 키워드로 인식 예: " 파란 셔츠 " -> [ " 파란 " , " 셔츠 " ] 2. 백엔드 검색 로직 (다중 키워드 AND 조건 분리된 모든 키워드가 상품명 또는 상품 설명에 포함된 상품을 검색 이를 위해 각 키워드에 대해 AND 조건으로 데이터베이스 쿼리를 실행 3. 공백 무시 검색 (띄어쓰기 없는 검색어 처리 "파란셔츠"와 같이 띄어쓰기 없이 검색하더라도, "파란 셔츠"가 검색 결과에 포함 구현 방법: 검색 시 DB의 상품명에서 공백을 제거한 후, 사용자의 검색어(공백 제거)와 비교 4. 부분 일치 검색 (축약어 및 일부 단어 검색) 기존의 `LIKE '%키워드%'` 검색 방식을 통해, 사용자가 “파란 셔츠 ” 에서 " 파 셔 " 와 같이 단어의 일부분만 입력해도 "파란 셔츠"가 검색 결과에 포함. 이는 다중 키워드 검색 로직과 결합되어, "파 셔"로 검색 시 '파'와 '셔'를 모두 포함하는 상품을 찾는다.
2	[오늘의 상품] getmapping /api/v1/product/list 로 상품 id, 상품 이름, 가격, 재고, 좋아요 수, 평균 평점, 생성일, 수정일로 asc, desc 정렬 가능하게 해 프론트에서 원하는 방식으로 상품들 출력 가능
3	[신규 업체] `GET /api/v1/seller/public/list` 를 사용해, 판매자 UID, 판매자가 가진 상품들의 총 좋아요 수, 상품들의 총 평균 평점, 총 리뷰 수, 회사명, 가입일 이 정렬 가능해 프론트에서 원하는 대로 리스트 출력 가능



번호	기능
1	[로그인 모드 설정] 사용자 유형(구매자 / 판매자 / 관리자)에 따라 로그인 경로를 구분하여, 각 역할에 맞는 권한과 화면을 제공
2	[로그인] 조회된 사용자의 `isActive()` 상태를 확인하여, 비활성화된 계정(`false`)의 로그인을 방지. 로그인 성공 시, 사용자의 고유 정보(`uid`, `userId`, `userNickname`, `userRole`)와 만료 시간(코드상 10시간)이 담긴 JWT 토큰 발급

우리샵

카테고리

업체 보기

베스트 업체

상품과 관련된 정보를 검색해보세요.

로그인

구매자 회원가입

아이디

아이디

아이디 중복확인

아이디 중복체크 필수

비밀번호

비밀번호

비밀번호 확인

입력한 비밀번호와 동일한 비밀번호로 입력하세요.

닉네임

닉네임

이메일

이메일 아이디

@

도메인 선택

이메일 중복확인

이메일 중복체크 필수

전화번호

휴대폰 번호

전화번호 중복확인

전화번호 중복체크 필수

생년월일

연도, 월, 일

성별

성별 선택

주소

주소

주소 검색

상세 주소

상세 주소

가입하기

우리샵

안녕!나 신속하고 정확하게 배송해 드립니다.

고객센터

1:1 문의

FAQ

배송 조회

이용안내

개인정보처리방침

이용약관

환불정책

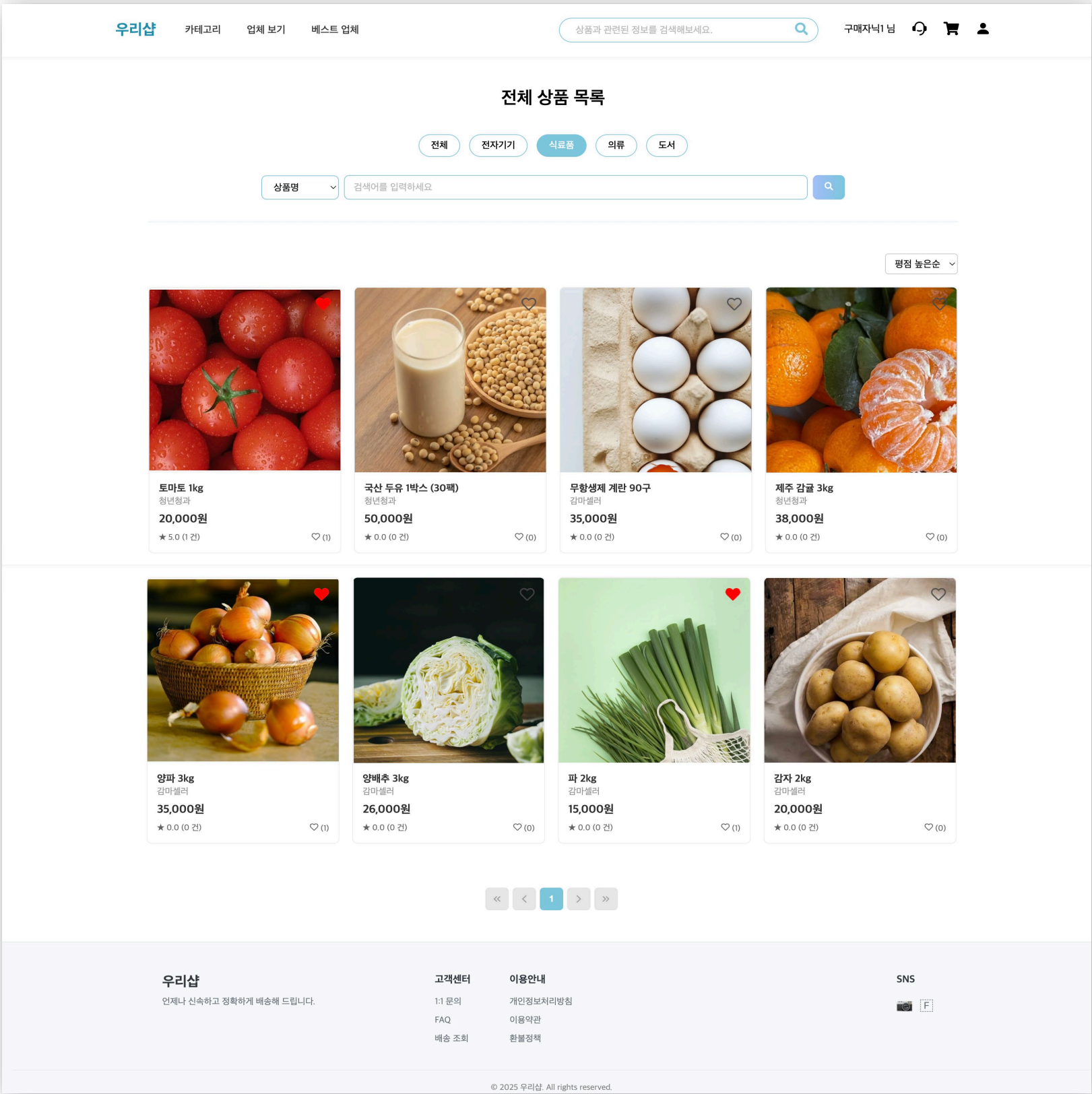
SNS

번호	기능
1	[중복체크] 실시간 중복 검사를 위해 `@PostMapping("/check-buyerId")`, `"/check-email"`, `"/check-phone"` 엔드포인트를 통해 회원 가입을 위한 유효성 체크 기능 제공. 프론트엔드에서 회원 가입 유효성 체크를 위한 기능들 제공



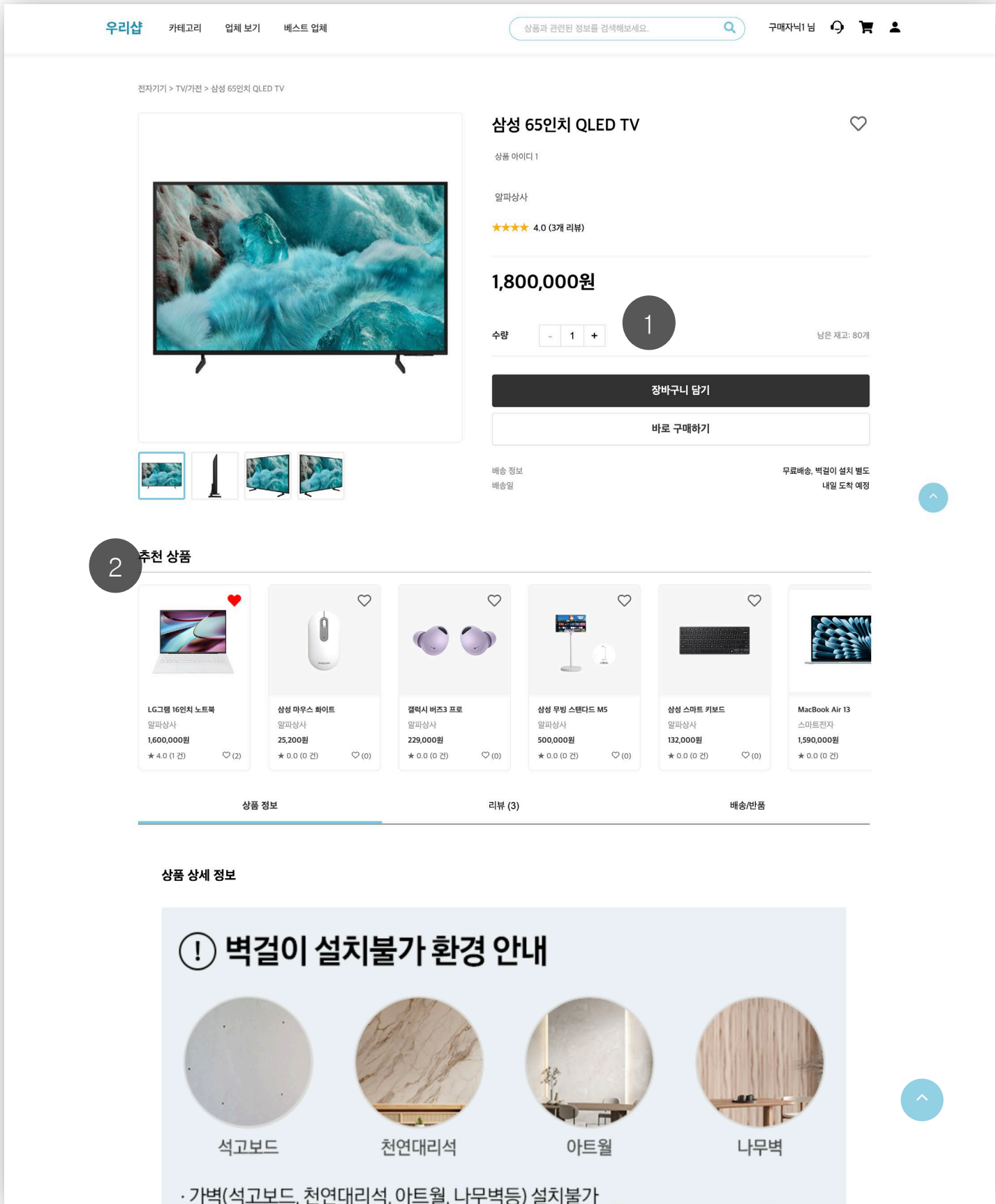
# 화면 설계 - 상품 리스트 페이지

프로젝트 설계

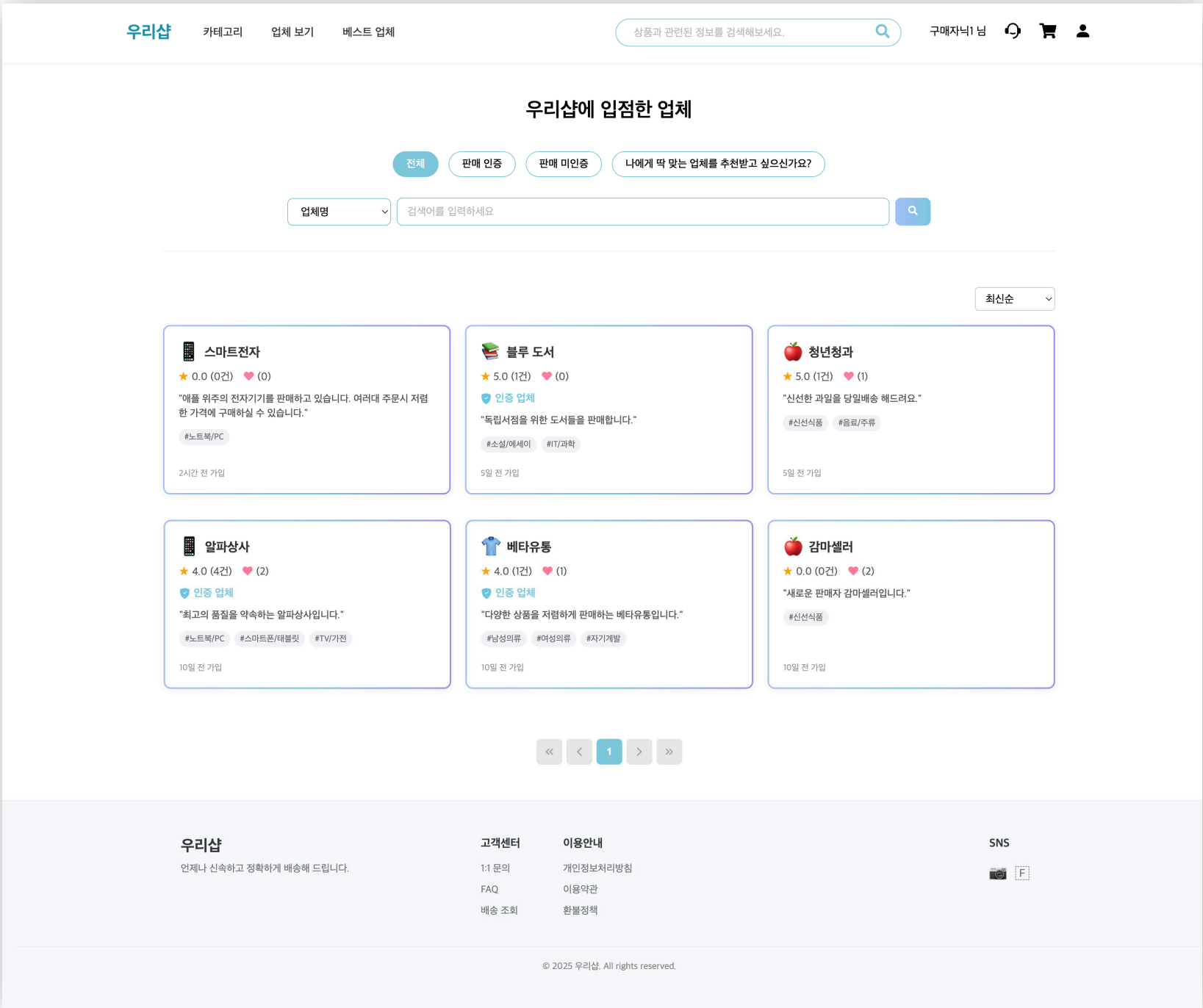


번호	기능
1	1. 카테고리 필터: 페이지 상단에 위치하며, 전체 상품 목록에서는 최상위 카테고리를, 특정 카테고리 진입 시에는 해당 카테고리의 하위 카테고리를 동적으로 보여줌. 이는 `GET /api/v1/category/list` API 응답을 기반으로 프론트엔드에서 렌더링되며, 필터 선택 시 해당 카테고리 파라미터와 함께 상품 목록 API를 재호출.
2	1. 결과 내 재검색: 현재 노출된 상품 목록 내에서 추가적인 검색을 지원. 사용자는 Select box를 통해 '상품명', '상품아이디', '업체명' 중 검색 기준을 선택하고 키워드를 입력 가능. 이 기능은 기존 조회 API(`GET /api/v1/product/list`)에 검색 기준과 키워드를 쿼리 파라미터로 추가하여 서버에서 효율적으로 데이터를 필터링.

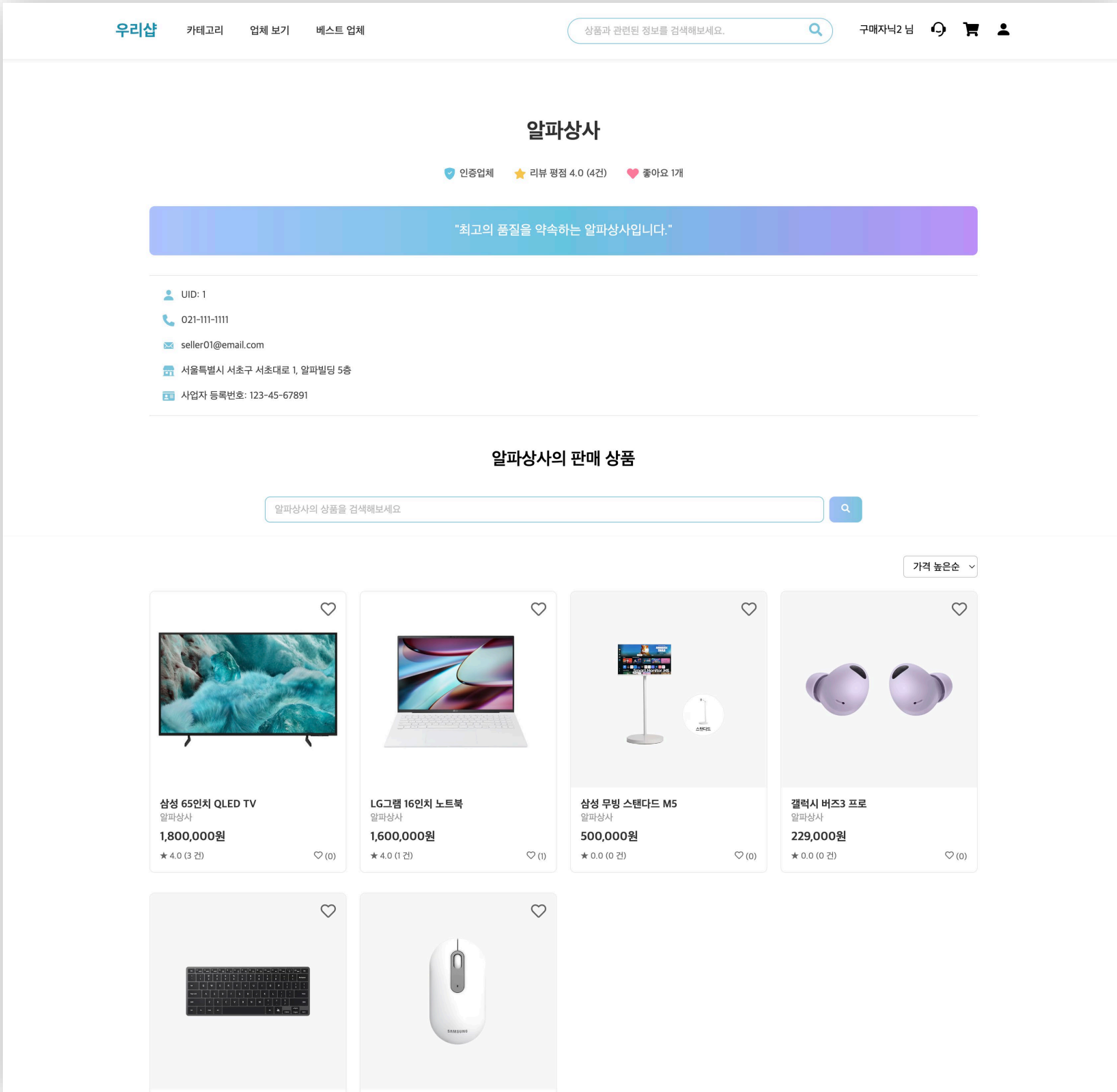




번호	기능
1	[수량 조절 및 구매] 사용자는 수량을 직접 입력하거나 버튼을 통해 조절 가능. 재고보다 많은 수량을 입력하거나 최소/최대 주문 수량을 벗어날 경우, 프론트엔드에서 유효성 검사를 통해 사용자에게 즉시 피드백 제공. '장바구니 담기' 또는 '바로 구매하기' 버튼 클릭 시, 백엔드의 `POST /api/v1/cart` 또는 `POST /api/v1/order` API를 호출하여 다음 단계를 진행.
2	[연관 상품 추천] 현재 보고 있는 상품과 동일한 카테고리에 속한 다른 상품들을 추천 목록으로 보여줌. 이는 `GET /api/v1/product/list` API에 현재 상품의 카테고리 ID를 파라미터로 전달하고, '좋아요'가 많은 순(`sort=likeCount,desc`)으로 정렬하여 가져온 데이터를 기반으로 함.



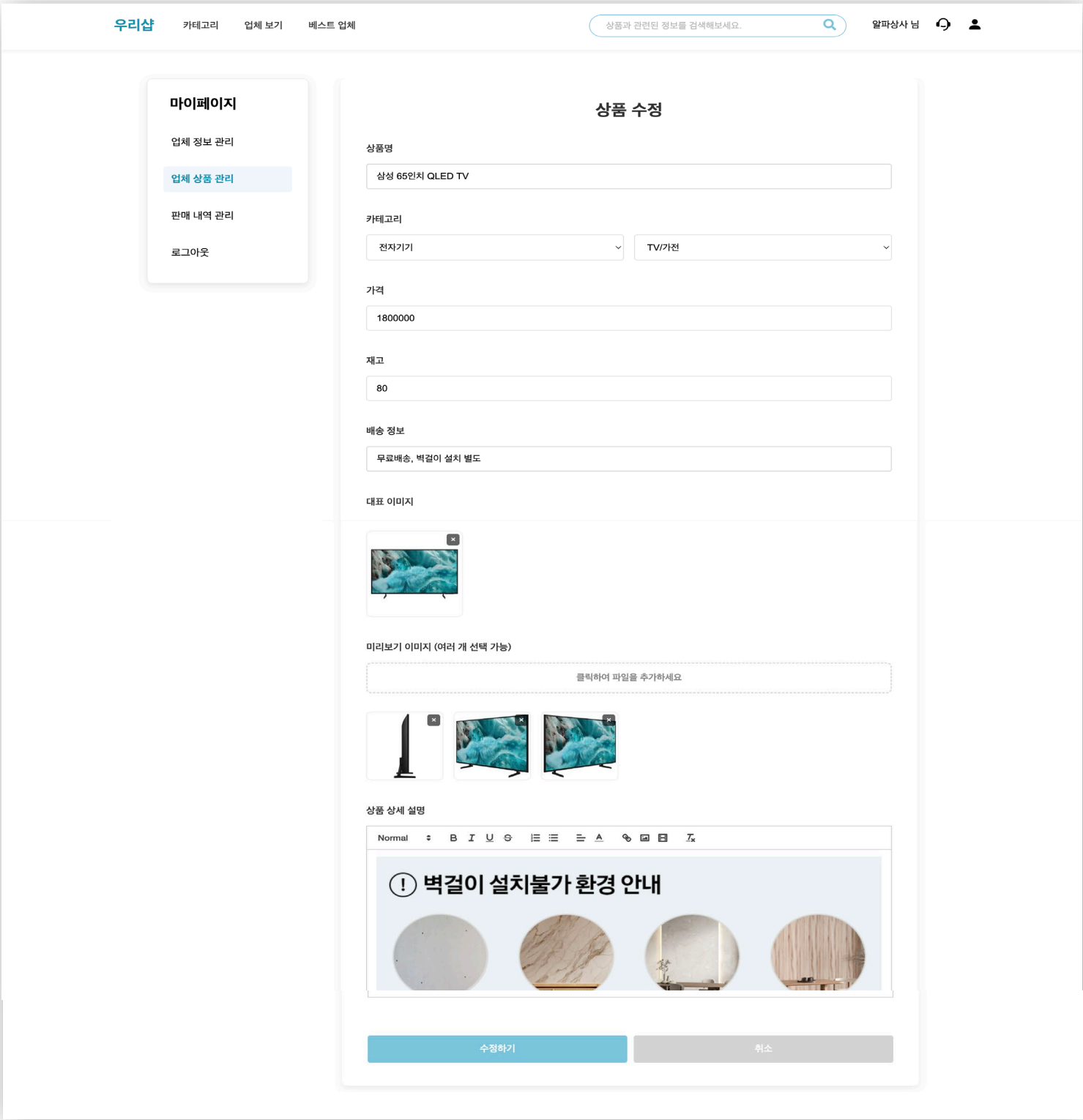
번호	기능
1	[업체 검색] 업체명 또는 사업자 등록번호를 기준으로 특정 업체를 검색. 이는 '결과 내 재검색'과 유사하게, `getSellerPublicList` API에 `sellerName={keyword}` 또는 `businessNumber={keyword}`와 같은 파라미터를 추가하여 서버 측에서 검색을 수행.
2	[업체 정보 카드] 각 업체의 신뢰도를 판단할 수 있는 핵심 정보를 요약하여 표시. 업체의 전체 상품에 대한 평균 평점, 총 리뷰 수, 총 '좋아요' 수를 표시. 이 데이터는 백엔드에서 `Seller` 엔티티와 관련된 `Review`, `ProductLike` 테이블을 조인하고 집계(Aggregation)하여 계산된 후, 업체 목록 조회 시 함께 반환.
3	[판매 인증 필터] '판매 인증' 여부에 따라 업체를 필터링하는 기능. 사용자가 '인증 업체' 필터를 활성화하면, 프론트엔드에서는 `GET /api/v1/seller/public/list` API 호출 시 `isVerified=true`와 같은 쿼리 파라미터를 추가하여, 백엔드에서 인증된 업체만 필터링하여 반환하도록 요청.



번호	기능
1	[업체 평가 정보 제공 ] 해당 업체의 신뢰도를 판단할 수 있는 정보 최상단 배치
2	[업체 상세 정보] 업체와 관련된 상세 정보 제공.
3	[해당 업체가 판매하는 상품 조회] 업체 단위로 상품 리스트 조회 가능. 해당 업체가 판매하는 상품내에서 검색 가능

# 화면 설계 - 상품 등록 및 수정 페이지(판매자 only)

프로젝트 설계



번호	기능
1	[상품 등록 또는 삭제] 등록 버튼 클릭 시 빈 input 창. 수정 버튼 클릭 시 기존 상품 input 값 보임.
2	[카테고리] 상위 카테고리 클릭 시 하위 카테고리 선택 가능.
3	[대표 이미지 및 서브 이미지 업로드] 사용자가 업로드한 이미지는 `multipart/form-data` 형식으로 서버에 전송. 서버는 이미지를 `product/{productId}/main` 및 `product/{productId}/sub` 폴더에 저장하여 상품별로 관리. UI에서 이미지 미리보기, 순서 변경, 삭제 기능 제공.
4	[상품 상세 설명] `React-Quill` 에디터를 사용하여 텍스트 서식, 이미지 삽입 등 풍부한 콘텐츠 작성을 지원.
5	[상세 설명 내 이미지 처리 (Base64 변환 및 파일 경로 치환)]  상품 ID 생성: 기본 정보로 상품을 먼저 저장하여 고유 `productId`를 확보. 이 ID는 모든 이미지 경로에 사용.  상세 설명 이미지 처리: `product/{productId}/description` 폴더를 먼저 생성. 그 후, 설명(HTML)에 포함된 Base64 이미지를 파일로 변환하여 해당 폴더에 저장하고, HTML의 `src` 속성을 파일 경로로 교체.  대표/추가 이미지 저장: `product/{productId}/main`, `product/{productId}/sub` 폴더를 생성하고, 전달받은 대표(main) 및 추가(sub) 이미지를 각각 저장.

시현 및 회고



## 1: 상품 좋아요 리뷰 고민

- 문제: 상품의 좋아요 와 리뷰 평점을 실시간으로 계산해 값을 불러와 dto 로 보여줄 때, 좋아요 개수랑 리뷰 평점 높은 순의 정렬이 불가능
- 해결: 상품 테이블에 likeCount, averageRating 컬럼을 추가해 값 변동 있을 때 마다 실시간으로 변동 시켜 해결

## 2: 배송 상태 변경

- 문제: order 테이블에 배송 상태가 있고, order\_detail 테이블에 배송 상태가 있을 때 처리 방법
- 해결: 지불, 배송 중, 배송 완료, 취소 상태 중 order\_detail 에서 취소가 하나라도 있으면 취소, 모두가 배송 완료면 배송 완료, 이외에 지불이 하나라도 있으면 지불, 배송 중 과 배송 완료가 있다면 배송 중이 뜨게 했습니다.

## 3: 로그인 계정 유형 3개 분리

- 처음에는 securityconfig.java 수정하기에 복잡할거 같아 로그인 계정을 buyer, seller, admin 타입 별로 분리를 했고, 각각의 개인 정보를 해당 detail 테이블에 넣었습니다. 그래서 controller 별로 /api/v1/admin, /api/v1/buyer, /api/v1/seller 식으로 각 권한을 설정하기가 용이 했으나, user 테이블로 통합하고 역할 별로 detail 테이블을 따로 설정하는게 낫지 않았을까 하는 복잡한 생각이 남기도 했습니다.

## 4: 개선 목표

- 중간 점검 때 많은 오류 들이 나타났고, 리눅스 서버에서 백엔드를 배포 했을 때, 일부 부분에서 하드 코딩 했던 곳들이 문제가 있었고, 오류들을 해결하고, 프론트엔드를 배포해 다시 실행 했을 때, 상품 이미지들이 나타나지 않는다든가, 여러 문제가 발생한 걸 보고, 다음엔 이 오류들을 서버 테스트를 좀 자주해야 할 거 같다고 생각했습니다.

## 1단계: 초기 목표

- 컴포넌트 구조화로 유지보수성과 확장성 확보
- 리렌더링 최소화 및 성능 최적화
- 반응형 & 사용자 경험 개선

## 2단계: 실제 결과

- UI/기능 단위로 모듈화된 컴포넌트 설계 및 중복 제거. BUT 일부 기능 분리 미흡
- Zustand + React Query로 상태 관리 분리 및 렌더링 제어. BUT 캐싱 처리에서 동작 미흡
- 반응형 시도. BUT 고정형으로 변경

## 3단계: 차이 및 발생 원인

- 초기 설계 단계에서 필터 및 결과 내 재검색 기능을 고려하지 않아, 동일한 로직이 여러 컴포넌트에서 중복되었음에도 공통 컴포넌트화하지 못함.
- React Query 사용 과정에서 쿼리키 설계에 대한 이해가 부족해, 캐싱 및 데이터 동기화 처리에 반복적인 문제가 발생함.
- 화면 구성이 테이블 형태로 확장될 가능성을 충분히 고려하지 못하고 반응형 UI를 우선 적용했다가, 개발 효율과 유지보수를 위해 고정형 UI로 방향을 전환함.

→ 설계 부족에 의해 발생한 이슈로 판단됨.

## 4단계: 개선 목표

- 공통 검색 및 필터 로직을 서비스/커스텀 혹은 단위로 분리하여 재사용성과 유지보수성 확보
- React Query 쿼리키 및 캐싱 전략 재설계로 데이터 일관성 및 성능 개선
- UI 특성에 맞는 반응형/고정형 전략을 사전에 정의하여 리소스 낭비 예방
- 기능 추가를 고려한 구조 설계 및 초기 설계 단계 강화를 통한 변경 비용 최소화

# 시현 링크

<https://ourshop.monster/>

---

<https://github.com/LuReich/shoppingMall>

---



감사합니다.

---

남현우, 추OO