



우리가 함께 만들어갈 단 하나의 감성 이커머스

≡ Tag

BE_L2

schenario

배경

좋아요 누르고, **쿠폰** 쓰고, 포인트로 **결제**하는 감성 이커머스.

내가 좋아하는 브랜드의 상품들을 한 번에 담아 주문하고,

유저 행동은 랭킹과 추천으로 연결돼요.

우린 이 흐름을 하나씩 직접 만들어갈 거예요.

서비스 흐름 예시

1. 사용자가 **회원가입**을 하고, **포인트를 충전**해요.
2. 여러 브랜드의 상품을 둘러보고, 마음에 드는 상품엔 **좋아요**를 누르죠.
3. 유저의 행동은 모두 기록되고, 그 데이터는 이후 다양한 기능으로 확장될 수 있어요.

요구사항


유저 (Users)

1. 사용자는 **쿠폰을 발급**받고, 여러 상품을 **한 번에 주문하고 결제**합니다.

METHOD	URI	설명
POST	/api/v1/users	회원가입
GET	/api/v1/users/me	내 정보 조회

인증/인가는 여기서 구현하지 않습니다.


X-USER-ID 헤더로 유저를 식별합니다.

 **회원가입**은 ID 및 성별, 생년월일, 이메일 주소를 받습니다.

 **X-USER-ID** 는 회원가입 시 가입한 ID 입니다.

포인트 (Points)

METHOD	URI	설명
POST	<code>/api/v1/points/charge</code>	포인트 충전
GET	<code>/api/v1/points</code>	보유 포인트 조회

 이 시스템은 실제 결제(PG) 대신, 사전 충전된 포인트로 상품을 결제합니다.


포인트 사용은 주문 과정에서 자동으로 처리되며, 별도의 사용 API는 제공되지 않습니다.

브랜드 & 상품 (Brands / Products)

METHOD	URI	설명
GET	<code>/api/v1/brands/{brandId}</code>	브랜드 정보 조회
GET	<code>/api/v1/products</code>	상품 목록 조회
GET	<code>/api/v1/products/{productId}</code>	상품 정보 조회

※ 상품 데이터는 사전 등록되어 있다고 가정합니다.

파라미터	예시	설명
<code>brandId</code>	1	특정 브랜드의 상품만 필터링
<code>sort</code>	<code>latest</code> / <code>price_asc</code> / <code>likes_desc</code>	정렬 기준
<code>page</code>	0	페이지 번호 (기본값 0)
<code>size</code>	20	페이지당 상품 수 (기본값 20)

 정렬 기준은 선택 구현입니다.

상품 목록 조회 쿼리 파라미터

필수는 `latest`, 그 외는 `price_asc`, `likes_desc` 정도로 제한해도 충분합니다.

좋아요 (Like)

METHOD	URI	설명
POST	/api/v1/like/products/{productId}	상품 좋아요 등록
DELETE	/api/v1/like/products/{productId}	상품 좋아요 취소
GET	/api/v1/like/products	내가 좋아요 한 상품 목록 조회

사용자는 각 상품에 한 번만 좋아요 를 누를 수 있습니다.
 상품에 대한 좋아요 추가/해제 기능은 멍등하게 동작하여야 합니다.
 상품 목록, 상품 상세 정보 조회 시 총 좋아요 수를 표기해야 합니다.



주문 / 결제 (Orders)

METHOD	URI	설명
POST	/api/v1/orders	주문 요청
GET	/api/v1/orders	유저의 주문 목록 조회
GET	/api/v1/orders/{orderId}	단일 주문 상세 조회

요청 예시:

```
{
  "items": [
    { "productId": 1, "quantity": 2 },
    { "productId": 3, "quantity": 1 }
  ]
}
```

주문 시 필요한 처리:

- * 상품 재고 확인 및 차감
- * 포인트 확인 및 차감
- * 주문 정보 외부 시스템 전송 (**Mock 처리 가능**)



나아가며

모든 API는 별도의 인증 없이 X-USER-ID 헤더로 동작합니다.

모든 기능의 동작을 개발한 후에 동시성, 멍등성, 일관성, 느린 조회, 동시 주문 등 실제 서비스에서 발생하는 문제들을 해결하게 됩니다.

