Swagger를 활용한 REST API 문서화

user-api-controller

> 💡 **사용자 등록**

- 요청 URL : `/api/users`
- HTTP 메서드 : `POST`
- 설명 : 새로운 사용자를 등록합니다.
- 요청 헤더 :
  - Content-Type : multipart/form-data
- Reqeust Body :

필드 이름	타입	필수 여부	설명
loginId	string	선택	사용자 로그인 ID
password	string	필수	사용자 비밀번호
email	string	필수	사용자 이메일 주소
phone	string	선택	사용자 전화번호
birthday	string(date)	선택	사용자 생년월일(yyyy-MM-dd 형식)
regionId	integer	필수	사용자 지역 ID
profileImage	binary	선택	사용자 프로필 이미지 파일

- Request Body 예제 :

```
{
  "loginId": "user123",
  "password": "password123",
  "email": "user@example.com",
  "phone": "010-1234-5678",
  "birthday": "1990-01-01",
  "regionId": 1,
  "profileImage": "(파일 첨부)"
}
```

- 성공 시 응답 (201 Created) :

```
{
  "loginId": "user123",
  "profileImageUrl": "profile_default.png",
  "region": "서울시",
  "status": "SUCCESS",
  "message": "회원가입 성공",
  "email": "user@example.com",
  "phone": "010-1234-5678",
  "manner": 36.5
}
```

- 필수 입력값 처리 : 필수 입력값이 누락된 경우, 서버는 오류를 반환하지 않고 누락된 필드 정보를 응답합니다. 사용자가 누락된 필드로 돌아가 입력을 완료하도록 유도합니다.

💡 사용자 조회

- 요청 URL : `/api/users`
- HTTP 메서드 : `GET`
- 설명 : `userSession` 객체를 `Query Parameter`로 전달받아 현재 사용자 정보를 반환합니다.
- 요청 파라미터 :

파라미터 이름	타입	필수 여부	설명
id	integer	필수	사용자 고유 ID
loginId	string	필수	사용자 로그인 ID
profileImageUrl	string	선택	사용자 프로필 이미지 URL
region	string	필수	사용자 지역 코드

- 요청 예제 :

```
{
  "id": 1,
  "loginId": "user123",
  "profileImageUrl": "string",
  "region": "서울시"
}
```

- 성공 시 응답 (200 OK) :

```
{
  "id": 1,
  "loginId": "user123",
  "profileImageUrl": "profile_default.png",
  "region": "서울시",
  "status": "SUCCESS",
  "message": "회원 조회 성공",
  "email": "user@example.com",
  "phone": "010-1234-5678",
  "createdAt": "2025-01-07T16:28:05.289378",
  "manner": 36.5,
  "failedLoginAttemptsCount": 0
}
```

💡 사용자 정보 수정

- 요청 URL : `/api/users`
- HTTP 메서드 : `PUT`
- 설명 : 사용자 정보를 업데이트합니다. `userSession` 객체로 현재 사용자 세션을 확인하고, `data` 객체로 업데이트할 정보를 제공합니다.
- 요청 헤더 :
  - Content-Type : multipart/form-data
- Request Body : `userSession`과 `data` 두 개의 주요 객체를 포함합니다.

  1. `userSession` 객체

  : 현재 사용자 세션 정보를 포함하는 객체로, 서버가 사용자를 식별하는 데 사용합니다.

필드 이름	타입	예제 값
id	integer	1
loginId	string	user123
profileImageUrl	string	string
region	string	서울시

  2. `data` 객체

  : 업데이트할 사용자 정보를 포함하는 객체입니다.

필드 이름	타입	설명
password	string	새로운 비밀번호

email	string	새로운 이메일 주소
phone	string	새로운 전화번호
regionId	integer	새로운 지역 ID

- 요청 예제 :

```
{
  "userSession": {
    "id": 1,
    "loginId": "user123",
    "profileImageUrl": "string",
    "region": "서울시"
  },
  "data": {
    "password": "newpassword",
    "email": "user@newmail.com",
    "phone": "010-1234-5678",
    "regionId": 2
  }
}
```

- 성공 시 응답 (200 OK) :

```
{
  "id": 1,
  "loginId": "user123",
  "profileImageUrl": "profile_default.png",
  "region": "강북구",
  "status": "SUCCESS",
  "message": "회원수정 성공",
  "email": "user@newmail.com",
  "phone": "010-1234-5678",
  "createdAt": "2025-01-07T16:28:05.289378",
  "manner": 36.5,
  "failedLoginAttemptsCount": 0
}
```

💡 **사용자 정보 삭제**

- 요청 URL : `/api/users`
- HTTP 메서드 : `DELETE`
- 설명 : 사용자 세션 정보를 기반으로 특정 사용자를 삭제합니다.

- 요청 예제 :

```
{
  "id": 1,
  "loginId": "user123",
  "profileImageUrl": "string",
  "region": "강북구"
}
```

- 성공 시 응답 (200 OK) :

```
{
  "status": "true"
}
```

💡 loginId로 사용자 조회

- URL : `/api/users/{loginId}`
- HTTP 메서드 : `GET`
- 설명 : `loginId`를 기반으로 사용자 정보를 조회합니다.
- 요청 파라미터 :

파라미터 이름	타입	설명
loginId	string	조회할 사용자 로그인 ID

- 성공 시 응답 (200 OK) :

```
{
  "id": 1,
  "loginId": "user123",
  "profileImageUrl": "profile_default.png",
  "region": "강북구",
  "status": "SUCCESS",
  "message": "회원 조회 성공",
  "email": "user@newmail.com",
  "phone": "010-1234-5678",
  "createdAt": "2025-01-07T17:55:31.756674",
  "manner": 36.5
}
```

- 예외 처리 : 사용자 정보 없음
  - 제공된 `loginId` 에 해당하는 사용자가 존재하지 않을 경우 사용자를 찾을 수 없다는 메세지를 반환합니다.

```
{
  "status": "FAILURE",
  "message": "사용자를 찾을 수 없음"
}
```

> 💡 **id로 사용자 조회**

- URL : `/api/users/id/{id}`
- HTTP 메서드 : `GET`
- 설명 : 사용자 고유 `id` 를 기반으로 사용자 정보를 조회합니다.
- 요청 파라미터 :

파라미터 이름	타입	설명
id	integer	조회할 사용자 ID

- 성공 시 응답 (200 OK) :

```
{
  "id": 1,
  "loginId": "user123",
  "profileImageUrl": "profile_default.png",
  "region": "강북구",
  "status": "SUCCESS",
  "message": "회원 조회 성공",
  "email": "user@newmail.com",
  "phone": "010-1234-5678",
  "createdAt": "2025-01-07T17:55:31.756674",
  "manner": 36.5
}
```

- 예외 처리 - 사용자 정보 없음
  - 제공된 `Id` 에 해당하는 사용자가 존재하지 않을 경우 사용자를 찾을 수 없다는 메세지를 반환합니다.

```
{
  "status": "FAILURE",
  "message": "사용자를 찾을 수 없음"
}
```

auth-api-controller

💡 로그인

- URL : `/api/login`
- HTTP 메서드 : `POST`
- 설명 : 사용자 로그인 요청을 처리합니다. 유효한 `loginId`와 `password`를 Body에 포함해야 합니다.
- Request Body :

필드 이름	타입	필수 여부	설명
loginId	string	필수	사용자의 로그인 ID
password	string	필수	사용자의 비밀번호

- 요청 예제 :

```
{
  "loginId": "user123",
  "password": "password123"
}
```

- 성공 시 응답 (200 OK) :

```
{
  "id": 1,
  "loginId": "user123",
  "profileImageUrl": "profile_default.png",
  "region": "강북구",
  "status": "SUCCESS",
  "message": "로그인 성공",
  "email": "user@newmail.com",
  "phone": "010-1234-5678",
  "createdAt": "2025-01-07T17:55:31.756674",
  "manner": 36.5,
  "failedLoginAttemptsCount": 0
}
```

- 예외 처리 - 인증 실패 (400 오류)
  - 잘못된 `loginId` 또는 `password`가 제공된 경우, 400 상태 코드를 반환합니다.

- 예외 처리 - 입력값 누락 (400 오류)
  - `loginId` 또는 `password`가 요청 Body에 누락된 경우, 400 상태 코드를 반환합니다.

💡 로그아웃

- URL : `/api/logout`
- HTTP 메서드 : `POST`
- 설명 : 현재 사용자를 로그아웃합니다. 요청 시 추가적인 파라미터는 필요하지 않습니다.
- 요청 파라미터 : 없음
- 성공 시 응답 (200 OK)

---

product-controller

💡 상품 등록

- URL : `/products`
- HTTP 메서드 : `POST`
- 설명 : 새로운 상품을 등록합니다.
- 요청 헤더 :
  - Content-Type : multipart/form-data
- 요청 Body

필드 이름	타입	필수 여부	설명
title	string	필수	상품 제목
description	string	필수	상품 설명
price	integer	필수	상품 가격
userId	integer	필수	상품 등록 사용자 ID
categoryId	integer	필수	카테고리 ID
status	string	필수	상품 상태
images	array(string)	선택	상품 이미지 파일 배열
province	string	선택	상품 위치(도)
city	string	선택	상품 위치(시)
town	string	선택	상품 위치(구/읍/면)
village	string	선택	상품 위치(동/리)

- 요청 예제 :

```
curl -X 'POST' \
  'http://localhost:8080/products' \
  -H 'accept: */*' \
  -H 'Content-Type: multipart/form-data' \
  -F 'userId=3' \
  -F 'price=1000' \
  -F 'city=수원시' \
  -F 'village=영통동' \
  -F 'province=경기도' \
  -F 'town=영통구' \
  -F 'status=ON_SALE' \
  -F 'title=당근' \
  -F 'images=@carrot.jpeg;type=image/jpeg' \
  -F 'categoryId=1' \
  -F 'description=맛있는 당근 팝니다'
```

- 성공 시 응답 (200 OK) :

```
Product created with ID: 1
```

- 예외 처리 - 입력값 누락 (400 오류)
  - 필수 필드가 누락된 경우, 400 상태 코드와 함께 오류 메시지를 반환합니다.
- 예외 처리 - 서버 오류 (500 오류)
  - 데이터베이스 또는 서버 오류가 발생한 경우, 500 상태 코드와 함께 오류 메시지를 반환합니다.

💡 상품 관심 등록

- URL : `/products/{productId}/favorite`
- HTTP 메서드 : `POST`
- 설명 : 특정 상품을 관심 등록합니다. 사용자 ID와 상품 ID를 전달해야 합니다.
- 요청 파라미터 :

필드 이름	타입	설명
userId	integer	관심 등록하는 사용자의 ID
productId	integer	관심 등록할 상품의 ID

- 요청 예제 :

```
curl -X 'POST' \
  'http://localhost:8080/products/1/favorite?userId=3' \
  -H 'accept: */*'
```

- 성공 시 응답 (200 OK) :

```
관심 상품으로 등록되었습니다.
```

- 예외 처리 - 상품 또는 사용자 ID가 유효하지 않은 경우
  - “존재하지 않는 상품 또는 사용자입니다.” 메세지를 반환합니다.
- 예외 처리 - 이미 관심 등록된 상품인 경우
  - “이미 관심 등록된 상품입니다.” 메세지를 반환합니다.

💡 관심 상품 해제

- URL : `/products/{productId}/favorite`
- HTTP 메서드 : `DELETE`
- 설명 : 특정 상품의 관심 등록을 해제합니다. 사용자 ID와 상품 ID를 전달해야 합니다.
- 요청 파라미터 :

필드이름	타입	설명
userId	integer	관심 해제 요청을 하는 사용자의 ID
productId	integer	관심 등록 해제할 상품의 ID

- 요청 예제 :

```
curl -X 'DELETE' \
  'http://localhost:8080/products/1/favorite?userId=3' \
  -H 'accept: */*'
```

- 성공 시 응답 (200 OK) :

```
관심 상품이 정상적으로 해제되었습니다.
```

- 예외 처리 - 관심 등록된 상품이 아닌 경우
  - “해당 상품은 관심 상품으로 등록되어 있지 않습니다.” 메세지를 반환합니다.

💡 상품 조회

- URL : `/products/{id}`
- HTTP 메서드 : `GET`
- 설명 : 특정 상품의 상세 정보를 조회합니다. `id`는 조회하려는 상품의 고유 ID입니다.
- 요청 파라미터 :

필드 이름	타입	설명
id	integer	조회할 상품의 ID

- 요청 예제 :

```
curl -X 'GET' \
  'http://localhost:8080/products/1' \
  -H 'accept: */*'
```

- 성공 시 응답 :

```
{
  "id": 1,
  "title": "당근",
  "description": "맛있는 당근 팝니다",
  "price": 1000,
  "userId": 3,
  "regionId": null,
  "category": {
    "id": 1,
    "name": "동네거래",
    "description": null,
    "enabled": false,
    "children": []
  },
  "status": "ON_SALE",
  "imageUrls": [
    "c86cd2a8-2567-4fc7-bb26-009f42d5f9cd.jpeg"
  ],
  "favoriteCount": 1,
  "viewCount": 2,
  "image": null,
  "viewed": false
}
```

💡 상품 삭제

- URL : `/products/{id}`
- HTTP 메서드 : `DELETE`
- 설명 : 특정 상품을 삭제합니다. `id`는 삭제하려는 상품의 고유 ID입니다.
- 요청 파라미터 :

필드 이름	타입	설명
id	integer	삭제할 상품의 ID

- 요청 예제 :

```
curl -X 'DELETE' \
  'http://localhost:8080/products/1' \
  -H 'accept: */*'
```

- 성공 시 응답 (200 OK) :

```
해당 상품이 삭제되었습니다.
```

💡 상품 수정

- URL : `/products/{id}`
- HTTP 메서드 : `PATCH`
- 설명 : 특정 상품의 정보를 수정합니다.
- 요청 파라미터 :

필드 이름	타입	설명
id	integer	수정할 상품의 ID

- 요청 예제 :

```
{
  "title": "당근",
  "price": 2000,
  "description": "맛있는 당근 팔아요",
  "status": "ON_SALE"
}
```

- 성공 시 응답 (200 OK) :


```
{
  "id": 1,
  "title": "당근",
  "description": "맛있는 당근 팔아요",
  "price": 2000,
  "userId": 3,
  "regionId": null,
  "category": {
    "id": 1,
    "name": "동네거래",
    "description": null,
    "enabled": false,
    "children": []
  },
  "status": "ON_SALE",
  "imageUrls": [
    "9b89b04f-1082-4350-9990-900bb714fb79.jpeg"
  ],
  "favoriteCount": 0,
  "viewCount": 0,
  "image": null,
  "viewed": false
}
```

💡 상품 상태 수정

- URL : `/products/{id}/status`
- HTTP 메서드 : `PATCH`
- 설명 : 특정 상품의 상태를 수정합니다.
- Request Path Parameter :

필드 이름	타입	설명
id	integer	수정할 상품의 ID

- Request Query Parameter :

필드 이름	타입	설명
status	string	상품 상태 (ON_SALE, SOLD_OUT, RESERVED, HIDDEN, SOLD)

- 요청 예제 :

```
/products/1/status?status=SOLD
```

- 성공 시 응답 (200 OK) :

```
{
  "id": 1,
  "title": "당근",
  "description": "맛있는 당근 팔아요",
  "price": 2000,
  "userId": 3,
  "regionId": null,
  "category": {
    "id": 1,
    "name": "동네거래",
    "description": null,
    "enabled": false,
    "children": []
  },
  "status": "RESERVED",
  "imageUrls": [
    "9b89b04f-1082-4350-9990-900bb714fb79.jpeg"
  ],
  "favoriteCount": 0,
  "viewCount": 0,
  "image": null,
  "viewed": false
}
```

💡 관심 등록한 상품 목록 조회

- URL : `/products/{userId}/favorites`
- HTTP 메서드 : `GET`
- 설명 : 사용자가 관심 등록한 상품 목록을 조회합니다.
- 요청 파라미터 :

필드 이름	타입	설명
userId	integer	관심 상품 조회를 요청한 사용자의 ID

- 성공 시 응답 (200 OK) :


```
{
  "id": 4,
  "title": "당근",
  "description": "맛있는 당근 팔아요",
  "price": 2000,
  "userId": 3,
  "regionId": null,
  "category": {
    "id": 1,
    "name": "동네거래",
    "description": null,
    "enabled": false,
    "children": []
  },
  "status": "RESERVED",
  "imageUrls": [
    "9b89b04f-1082-4350-9990-900bb714fb79.jpeg"
  ],
  "favoriteCount": 1,
  "viewCount": 0,
  "image": null,
  "viewed": false
}
```


- 예외 처리 - 사용자가 아무 상품도 관심 상품으로 등록하지 않은 경우


```
{
  "message": "해당 유저의 관심 상품이 등록되어 있지 않습니다."
}
```

💡 **상품 판매자 정보 조회**

- URL : `/products/{id}/seller-info`
- HTTP 메서드 : `GET`
- 설명 : 특정 상품의 판매자 정보와 해당 판매자가 등록한 다른 상품 목록을 조회합니다.
- 요청 파라미터 :

필드 이름	타입	설명
id	integer	조회하려는 상품 ID

- 성공 시 응답 (200 OK) :

```
{
  "otherProducts": [
    {
      "productId": 6,
      "title": "맛있는 토마토",
      "price": 3000
    }
  ],
  "sellerProfile": {
    "userId": 3,
    "loginId": "user123",
    "profileImageUrl": "profile_default.png",
    "mannerTemperature": 36.5
  }
}
```

💡 **상품 판매자 매너 온도 조회**

- URL : /products/{id}/manner-temperature
- HTTP 메서드 : GET
- 설명 : 특정 상품의 판매자 매너 온도를 조회합니다. 판매자가 상품 거래를 완료해 거래 상태가 `COMPLETED`로 바뀌면 매너 온도가 1씩 증가합니다. 초기 매너 온도는 36.5입니다.
- 요청 파라미터 :

필드 이름	타입	설명
id	integer	조회하려는 상품 ID

- 성공 시 응답 (200 OK) :

```
{
  "sellerId": 3,
  "mannerTemperature": 36.5
}
```

💡 특정 사용자가 등록한 모든 상품 조회

- URL : `/products/user/{userId}`
- HTTP 메서드 : `GET`
- 요청 파라미터 :

필드 이름	타입	설명
userId	integer	조회할 사용자 ID

💡 특정 사용자가 등록한 상품 중 특정 상태를 가진 상품 조회

- URL : `/products/user/status`
- HTTP 메서드 : `GET`
- 설명 : 특정 사용자가 등록한 상품 중 특정 상태(`ON_SALE`, `SOLD_OUT` 등)를 가진 상품을 조회합니다.
- 요청 파라미터 :

필드 이름	타입	설명
userId	integer	사용자 ID
status	string	상품 상태

💡 최근 등록된 상품 10개 조회

- URL : `/products/top`
- HTTP 메서드 : `GET`

💡 특정 상태의 모든 상품 조회

- URL : `/products/status`
- HTTP 메서드 : `GET`

- 요청 파라미터 :

필드 이름	타입	설명
status	string	상품 상태(`ON_SALE` , `SOLD_OUT` 등)

💡 특정 사용자가 등록한 상품 중 제목으로 검색

- URL : `/products/search`
- HTTP 메서드 : `GET`
- 설명 : 특정 사용자가 등록한 상품 중 제목 부분만으로 검색해도 그 부분 포함한 상품들을 검색해 조회합니다.
- 요청 파라미터 :

필드 이름	타입	설명
userId	integer	사용자 ID
title	string	검색할 제목 키워드

💡 상품 카테고리, 지역, 가격 범위 등 다양한 조건으로 필터링된 상품 조회

- URL : `/products/filter`
- HTTP 메서드 : `GET`
- 요청 파라미터 :

필드 이름	타입	설명
categoryId	integer	카테고리 ID
regionId	integer	지역 ID
categoryName	string	카테고리 이름
soryBy	string	정렬 기준
fixedMaxPrice	integer	최대 가격
minPrice	integer (기본 값 : 0)	최소 가격
maxPrice	integer (기본값: 100000000)	최대 가격

---

chat-api-controller

💡 **채팅 시작**

- URL : `/talk`
- HTTP 메서드 : `POST`
- 요청 파라미터 :
  1. userSession
     - 필수 여부 : 필수
     - 타입 : object
     - 요청 예제 :

```
{
  "id": 3,
  "loginId": "user123",
  "profileImageUrl": "string",
  "region": "수원시"
}
```

  2. productId
     - 필수 여부 : 필수
     - 타입 : integer
     - 설명 : 채팅을 시작할 상품의 ID
- 성공 시 응답 (200 OK) :

```
{
  "id": 1,
  "productId": 4,
  "productName": "당근",
  "productImageUrl": "user123",
  "participantLoginId": "profile_default.png",
  "productStatus": "RESERVED"
}
```

💡 사용자의 모든 채팅방 목록 조회

- URL : `/talk`
- HTTP 메서드 : `GET`
- 요청 파라미터 :
  1. userSession
     - 필수 여부 : 필수
     - 타입 : object
     - 요청 예제 :

```
{
  "id": 3,
  "loginId": "user123",
  "profileImageUrl": "string",
  "region": "수원시"
}
```

- 성공 시 응답 (200 OK) :

```
{
    "chatRoomId": 1,
    "productId": 4,
    "productTitle": "당근",
    "productPrice": 2000,
    "otherUserLoginId": "user123",
    "otherManner": 36.5,
    "partnerProfileImage": "profile_default.png",
    "lastMessage": "최근 대화가 없습니다.",
    "lastMessageTime": null,
    "unreadMessageCount": 0,
    "productStatus": "RESERVED",
    "productImageURL": "9b89b04f-1082-4350-9990-900bb714fb79.jpeg"
  }
```

💡 특정 채팅방의 채팅 히스토리 조회

- URL : `/talk/{roomId}`
- HTTP 메서드 : `GET`

- 요청 파라미터 :
    1. roomId
        - 필수 여부 : 필수
        - 타입 : integer
        - 설명 : 조회할 채팅방의 ID