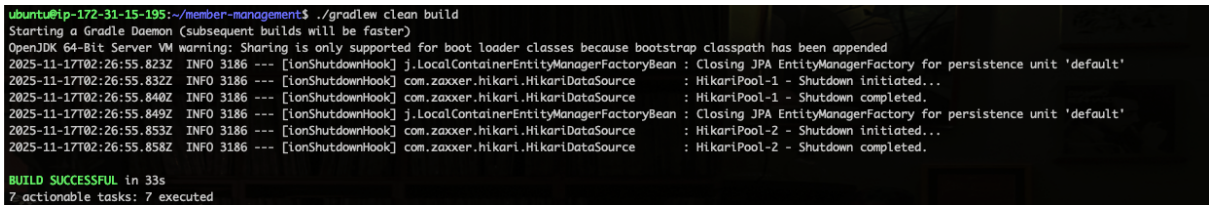
문제 1. 회원관리 REST API

1. 엔티티 테이블 설계

```sql
DROP TABLE IF EXISTS `member`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8mb4 */;
CREATE TABLE `member` (
                      `id` int(11) NOT NULL AUTO_INCREMENT,
                      `created_at` datetime(6) NOT NULL,
                      `email` varchar(100) NOT NULL,
                      `password` varchar(100) NOT NULL,
                      `username` varchar(50) NOT NULL,
                      PRIMARY KEY (`id`),
                      UNIQUE KEY `UKmbmcqelty0fbrvxp1q58dn57t`
(`email`),
                      UNIQUE KEY `UKgc3jmn7c2abyo3wf6syln5t2i`
(`username`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

2. 빌드 성공 화면



3. 포트 정보

1. 애플리케이션 포트

- Spring Boot 기본 포트: 8080
- (nginx 리버스 프록시 사용시) 외부 접속 포트: 80

2. DB 정보

- DB 종류: MariaDB
- 호스트: localhost
- 포트: 3306
- DB 이름: member_management

4. 실행가이드

1. 빌드 환경 및 사전 준비

```
- JDK: OpenJDK 21 이상
- 빌드 도구: Gradle (프로젝트 내부 `./gradlew` 사용)
- DB: MariaDB 10.x
- 기본 실행 프로필: local
```

2. 소스코드 클론

```
```bash
git clone https://github.com/lmw7414/member-management.git
cd member-management
./gradlew clean build
```

3. DB 설정

```
CREATE DATABASE member_management CHARACTER SET utf8mb4;

CREATE USER 'member'@'localhost' IDENTIFIED BY '1234';
GRANT ALL PRIVILEGES ON member_management.* TO 'member'@'localhost';

FLUSH PRIVILEGES;
```

4. 서비스 실행

```
java -jar build/libs/app.jar ₩
      --spring.profiles.active=local ₩
      --spring.datasource.username=member ₩
      --spring.datasource.password=1234
```

(2 번까지 진행 후) Docker 사용시 (Docker, Docker-compose 설치 필요)

.env 파일 추가(docker-compose 파일과 동일한 디렉토리에)

```
SPRING_DATASOURCE_USERNAME=member
SPRING_DATASOURCE_PASSWORD=1234
SPRING_PROFILES_ACTIVE=prod
```

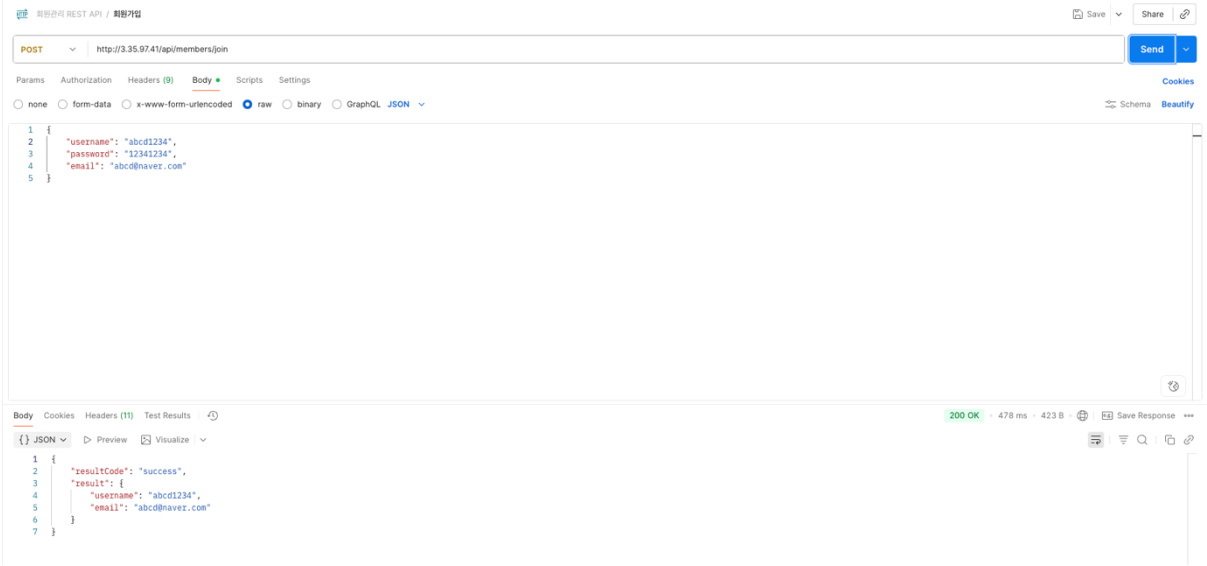
```
docker-compose up -d --build
```
```

5. 기능 설명

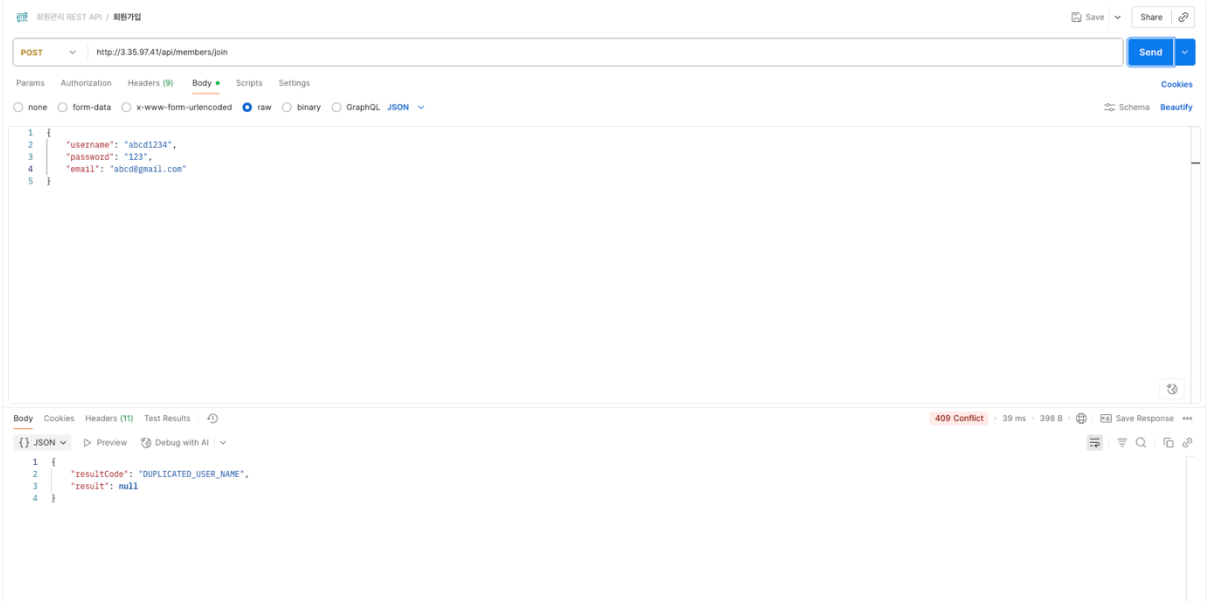
1. 회원가입(POST /api/members/join)

클라이언트로부터 사용자이름(username), 비밀번호(password), 이메일(email) 정보를 전달받아 새로운 회원을 생성하고 DB에 저장합니다. 회원가입이 완료되면 생성된 회원정보(사용자 이름, 이메일)를 응답으로 반환합니다.

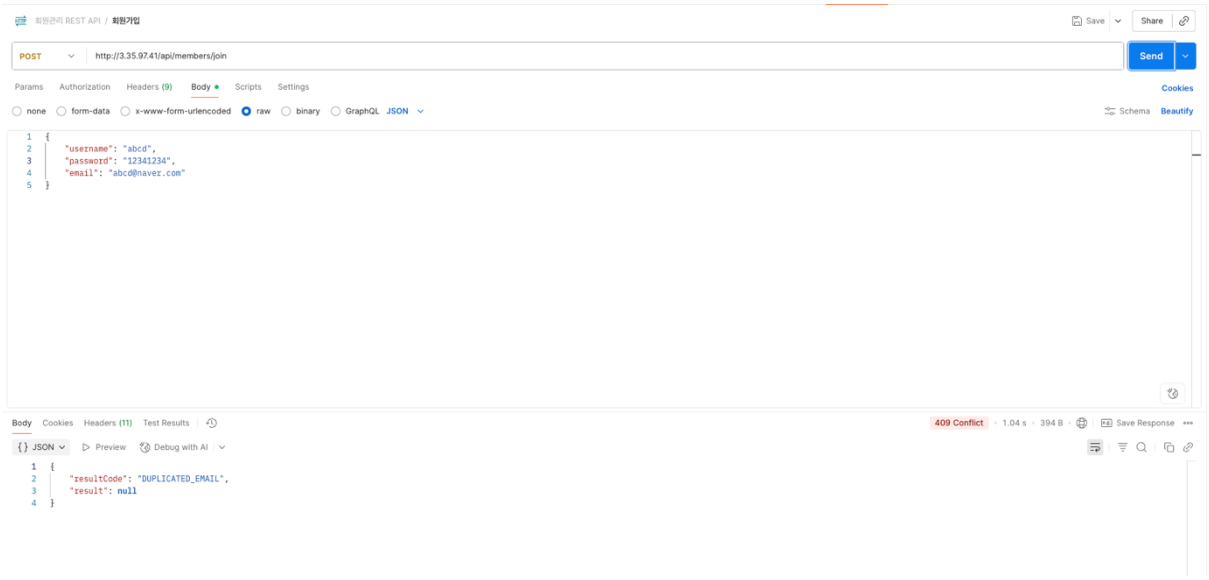
```
{
    "username": "test",
    "password": "12341234",
    "email": "test@naver.com"
}
```



예외 상황1. 이미 존재하는 username의 경우(409 Conflict)



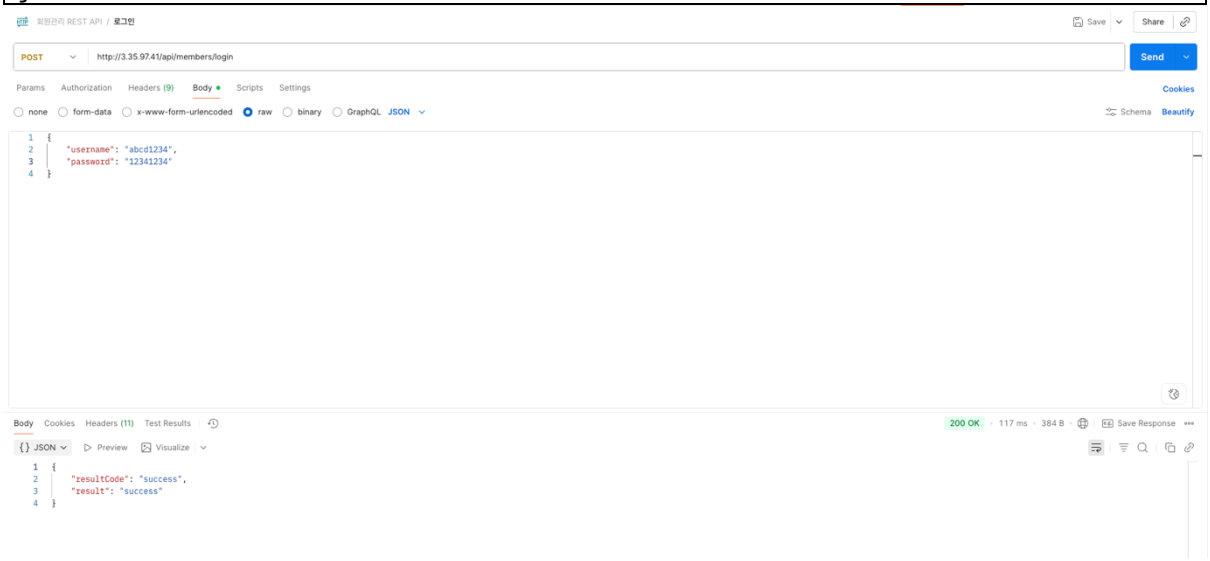
예외상황2. 이미 존재하는 email의 경우(409 Conflict)



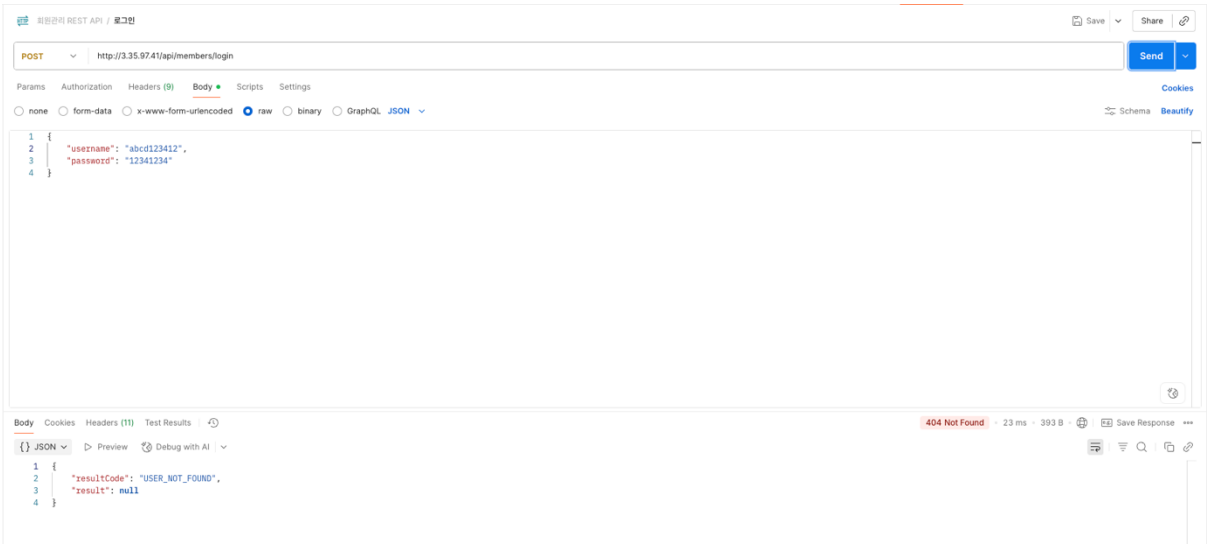
2. 로그인(POST /api/members/login)

기존에 가입된 사용자가 아이디/비밀번호를 이용해 서비스에 로그인하는 기능입니다. 전달받은 정보가 유효한지 확인 후, 성공 시 success를 반환합니다.(세션, 토큰 등 실제 구현 방식에 따라 달라짐)

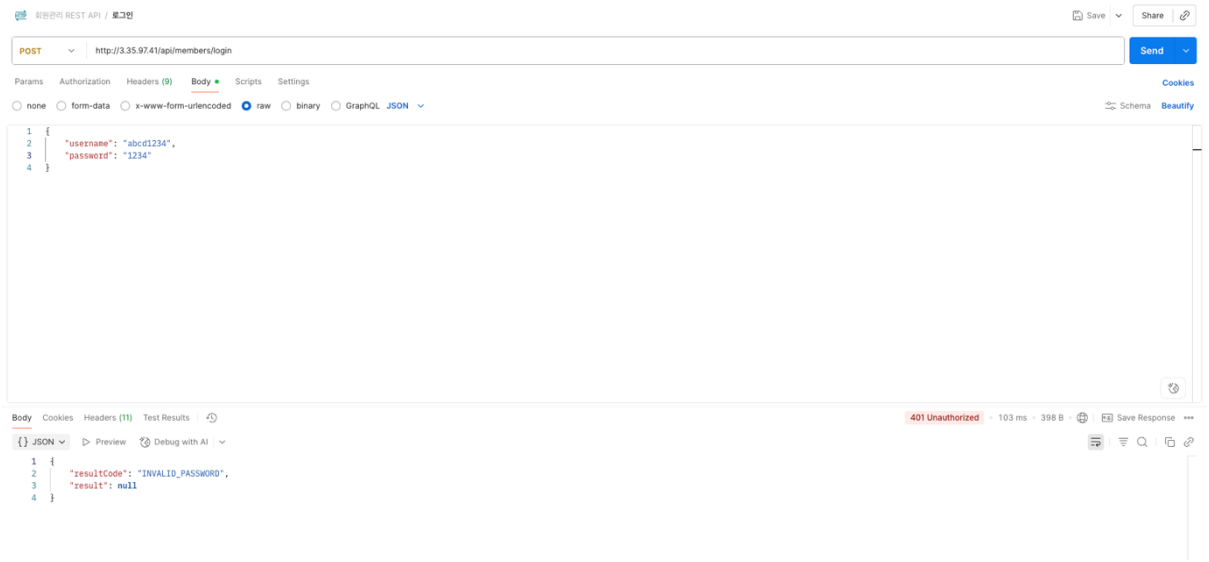
```
{
    "username": "test",
    "password": "12341234"
}
```



예외상황1. 없는 username의 경우(404 Not Found)

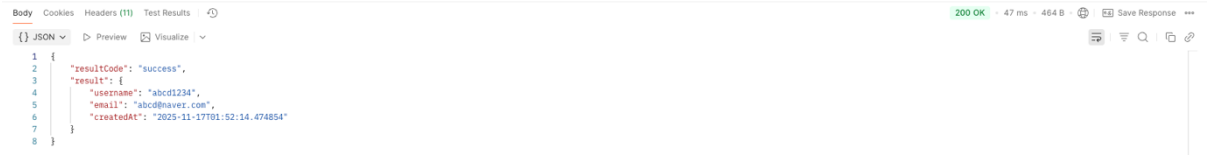


예외상황2. 잘못된 비밀번호의 경우(401 Unauthorized)

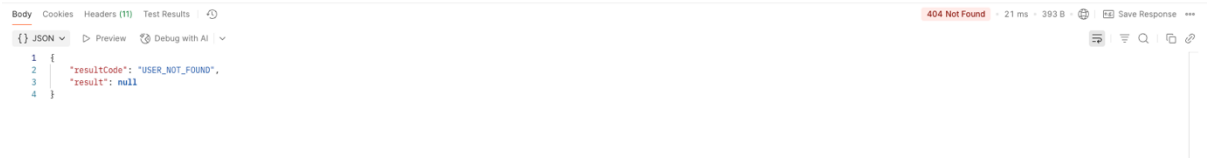


3. 단건조회(GET /api/members/{username})

특정 회원의 상세 정보를 조회하는 기능입니다. URL 경로에 전달된 `username` 값을 기준으로 회원을 조회하고, 해당 회원의 기본정보(username, email, createdAt)를 반환합니다.



예외상황1. 없는 username의 경우(404 Not Found)



4. 전체조회(GET /api/members)

현재 시스템에 등록된 모든 회원 목록을 조회하는 기능입니다.