프로젝트 보고서

데이터베이스설계04

깃허브 주소: https://github.com/Ykmykmkkk/Database_MusicDiary/tree/master

1. 설계 개요

MusicDiary 앱은 사용자가 음악을 들으며 느끼는 감정과 생각을 기록하고, 다른 사람들이 어떤 노래를 좋아하며 어떤 감상을 갖고 있는지를 알 수 있는 커뮤니티를 제공하는 플랫폼이다. 사용자들은 자신의 음악에 대한 감상을 기록하고 공유할 수 있으며, 다른 사람들의 음악적 경험을 통해 더 풍부한 감정적 교류를 할 수 있다. 이러한 기능을 실현하기 위해서는 사용자 정보와 리뷰 데이터가 체계적으로 저장되고, 손쉽게 관리될 수 있어야 한다. 데이터베이스는 데이터를 효율적으로 저장하고, 다양한 쿼리와 연산을 통해 빠르고 정확하게 검색할 수 있는 기능을 제공한다. 예를 들어 사용자가 설정한 공개/비공개 옵션에 따라 리뷰를 필터링하거나 내가 좋아요 한 노래 및 감상을 조회하는데 있어 데이터베이스의 역할은 매우 중요하다. 또한, 데이터의 확장성과 유지보수를 고려해야 하기 때문에 정교한 데이터베이스 설계를 진행하고 구현하였다.

2. 요구사항 분석 - 요구사항 명세서

① MusicDiary 어플리케이션에 회원으로 가입하려면 회원 아이디, 비밀번호, 이름, 이메일을 입력해야 한다.

② 회원은 회원 아이디를 통해 고유하게 식별된다.

③ 회원 탈퇴 시, 해당 회원의 활동 기록(리뷰, 좋아요 등)은 유지되지만 회원 정보는 비활성화 처리된다.

④ 노래는 노래명, 앨범명, 가수명, 발매일, 재생 시간 정보를 유지해야 한다.

⑤ 노래는 노래명과 가수명의 조합으로 고유하게 식별된다.

⑥ 회원은 노래를 추가하고, 노래에 대한 감상평(리뷰)을 작성할 수 있다.

⑦ 리뷰를 저장할 때 작성한 회원의 회원아이디, 노래명, 가수명, 리뷰 작성 날짜, 리뷰 내용, 공개여부 정보를 입력해야 한다.

⑧ 리뷰에 대한 작성 회원 식별자, 노래 식별자, 리뷰 작성 날짜, 리뷰 내용, 공개여부 정보를 유지해야 한다.

⑨ 리뷰는 회원 아이디과 리뷰 작성 날짜의 조합으로 고유하게 식별된다.

⑩ 하나의 회원은 여러 노래에 대해 리뷰를 작성할 수 있다.

⑪ 하나의 노래에 여러 회원이 리뷰를 작성할 수 있다.

⑫ 동일 회원이 동일 날짜에 두 개 이상의 리뷰를 작성할 수 없다.

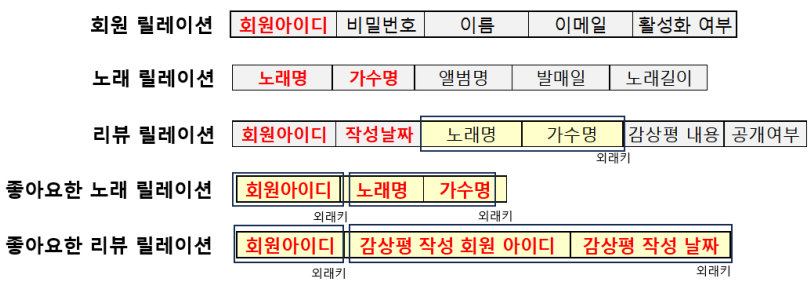
⑬ 회원과 노래 간 좋아요 관계는 회원 아이디와 노래명 및 가수명의 조합으로 고유하게 식별된다.

⑭ 회원과 리뷰 간 좋아요 관계는 회원 아이디와 리뷰 작성자 회원아이디 및 리뷰 작성 날짜의 조합으로 고유하게 식별된다.

⑮ 한 명의 회원은 여러 노래, 리뷰에 좋아요를 누를 수 있다.

16 하나의 노래, 리뷰는 여러 회원이 좋아요를 누를 수 있다.

3. 논리적 설계 - 릴레이션 스키마



4. 물리적 설계 및 구현

개발 환경 및 프레임워크

Front-end: Flutter (dart)

Back-end: Spring Boot (java 17) / Database: MySql (local)

OS: windows 11 / Test Simulator: Pixel 5 Api 34 (android studio emulator)

백 엔드는 객체 지향적 프로그래밍을 지향하는 프레임워크인 스프링 부트를 기반으로 RESTful API 서버를 설계 및 구현했다. 객체지향적 설계는 유지보수가 수월하고 개발자가 비즈니스 로직에 집중하여 생산성을 높일 수 있다는 장점이 있지만, 데이터 중심이 아닌 객체 중심의 설계는 비효율적인 SQL 쿼리를 생성할 가능성이 높고 데이터 조작의 제약이 있을 수 있다는 단점이 있다. 이러한 한계를 보완하기 위해, 복잡한 데이터 조회와 성능 최적화를 할 수 있는 Native Query와 JPQL을 활용해 필요한 쿼리를 추가로 작성했다. 데이터베이스는 MySQL을 채택하고 JPA/Hibernate를 활용해 객체-관계 매핑(ORM)을 구현했다. 프론트 엔드는 Flutter를 활용하여 iOS와 Android에서 동작하는 크로스플랫폼 모바일 어플리케이션을 개발했다

데이터 무결성 및 효율성

물리적 데이터베이스 설계 시, 각 테이블에 Long 타입의 ID를 기본 키로 설정하고, 이를 외래 키로 참조할 수 있도록 설계했다. 숫자형 기본 키(ID)는 검색, 조인, 및 인덱싱 작업에서 성능 향상을 제공하고 참조 무결성을 유지하는 데에도 유리하기 때문이다. 또한 데이터베이스 단에도 unique key constraint를 설정하여 중복된 데이터 입력을 방지하고, 데이터의 무결성을 강화했다. 백 엔드 서버단에서는 복잡한 데이터 조회와 성능 최적화를 위해 Native Query와 JPQL을 적절히 활용하여 데이터를 효율적으로 처리하였으며, 트랜잭션 무결성을 유지하기 위해 @Transactional을 사용하여 데이터 일관성을 보장했다. 특히, 트랜잭션 격리 수준을 Isolation.SERIALIZABLE로 설정하여, 여러 트랜잭션이 동시에 처리될 때 발생할 수 있는 교착 상태나 더티 리드를 예방하고자 노력했다.

```
@Transactional(readOnly = true)
public ReviewResponseDto getReviewDate(String username, LocalDate date)
```

읽기 전용 트랜잭션 설정을 통해 데이터베이스의 성능 최적화

```
@Transactional(isolation = Isolation.SERIALIZABLE)
public void setReviewLike(String username, SetReviewLikeRequestDto setReviewLikeRequestDto) {
```

Create, Update, Delete 작업 시, 동일한 데이터에 대해 여러 트랜잭션이 동시에 수정하려는 상황을 차단

```
@Query("SELECT r FROM Review r JOIN FETCH r.user JOIN FETCH r.song WHERE r.isPublic = true")
List<Review> findAllByIsPublicTrue();
```

JQPL 쿼리를 이용한 JOIN 연산으로 성능 최적화

```
@Table(name = "song", uniqueConstraints = {
        @UniqueConstraint(columnNames = {"title", "artist"})
})
```

UNIQUE KEY 제약 조건 설정

실제 실행 예제

요구사항 명세서에 작성 되어있는 모든 기능을 구현하였으나 보고서 분량이 제한적인 관계로 어플리케이션의 핵심 기능인 사용자가 리뷰를 작성하여 데이터베이스에 저장하는 부분만 살펴볼 것이다.

#리뷰 저장하기

```
{
  "title": "whiplash",
  "artist": "aespa",
  "album": "1thAlbum",
  "releaseDate": "2024-10-21",
  "durationTime": "3:15",
}
```

사용자가 리뷰 저장을 요청하며 보낸 JSON 데이터

```
@Modifying
@Query(value = """
INSERT INTO review (is_public, review_content, review_date, song_id, user_id)
SELECT :isPublic, :reviewContent, :reviewDate,
       (SELECT song_id FROM song WHERE title = :songTitle AND artist = :songArtist),
       (SELECT user_id FROM user WHERE username = :username)
WHERE NOT EXISTS (
    SELECT 1 FROM review r
    JOIN user u ON r.user_id = u.user_id
    WHERE u.username = :username AND r.review_date = :reviewDate
)
""", nativeQuery = true)
void createReviewbyNativequery(
        @Param("isPublic") boolean isPublic,
        @Param("reviewContent") String reviewContent,
        @Param("reviewDate") LocalDate reviewDate,
        @Param("songTitle") String songTitle,
        @Param("songArtist") String songArtist,
        @Param("username") String username
);
```

DB최적화를 위해 Native Query문으로 처리

```
INSERT
INTO
    review
    (is_public, review_content, review_date, song_id, user_id) SELECT
        ?,
        ?,
        ?,
        (SELECT
            song_id
        FROM
            song
        WHERE
            title = ?
            AND artist = ?),
        (SELECT
            user_id
        FROM
            user
        WHERE
            username = ?)
    WHERE
        NOT EXISTS (    SELECT
            1
        FROM
            review r
        JOIN
            user u
                ON r.user_id = u.user_id
        WHERE
            u.username = ?
            AND r.review_date = ? )
```

실행된 SQL 쿼리문

데이터베이스 결과

user_id	deleted	email	name	password	username
1	0	g@gmail.com	ykm	$2a$10$alsBt9eKSZzAyHXMQ1axd.06q.RIkDZiLiw48Tb…	test1
2	0	hi@gmail.com	ykmm	$2a$10$y9DOXT5aML1Av9tu0z5FBe9qHPeJLSDukMPKpdv…	test2

USER 테이블

song_id	album	artist	duration_time	release_date	title
9	1thAlbum	aespa	3:15	2024-10-21	whiplash
10	Fourever	DAY6	3:12	2024-03-18	Happy
11	LOSE Yourself	KISS OF LIFE	3:22	2024-10-15	Igloo
12	12	빈지노	3:04	2016-05-31	Break
13	향	능인스님	2:58	2017-10-30	아, 인생이여

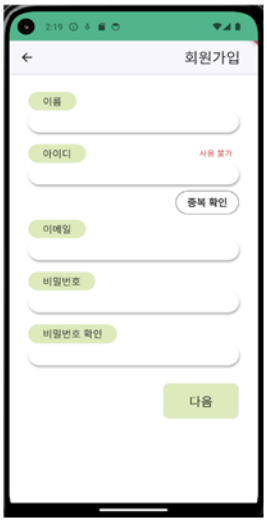
SONG 테이블

id	is_public	review_content	review_date	song_id	user_id
2	false	This song makes me high	2024-11-28	9	1
3	true	it makes me happy	2024-11-29	10	1
4	true	this song makes me free	2024-11-29	12	2
10	true	아, 인생 쓰다	2024-11-30	13	1

작성한 리뷰가 삽입된 REVIEW 테이블

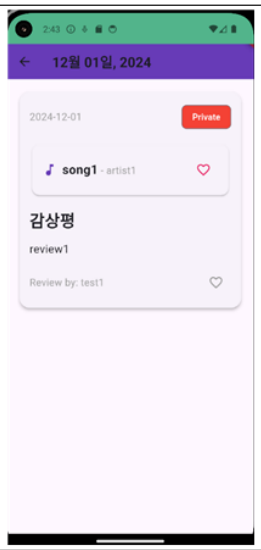
데이터베이스에 리뷰가 문제 없이 저장되어 있는 것을 확인할 수 있다.

5. 실제 어플리케이션 구현 화면

<table>
<tr>
<td> 
</td>
<td> 
</td>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td>로그인 화면</td>
<td>로그인 실패</td>
<td>회원가입</td>
<td>오늘의 음악 기본화면</td>
</tr>
<tr>
<td> 
</td>
<td> 
</td>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td>음악 설정 화면</td>
<td>노래가 존재하지 않을시</td>
<td>노래 추가 화면</td>
<td>감상평 작성 화면</td>
</tr>
</table>

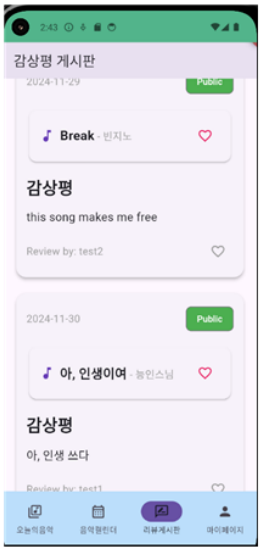
감상평 저장

음악캘린더 기본화면

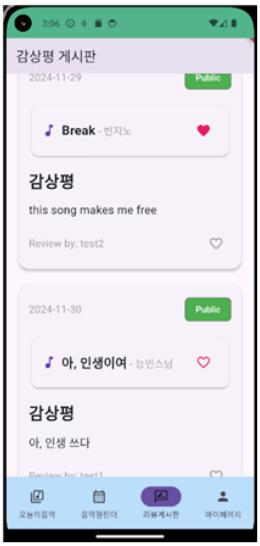


12월 1일 감상평 조회

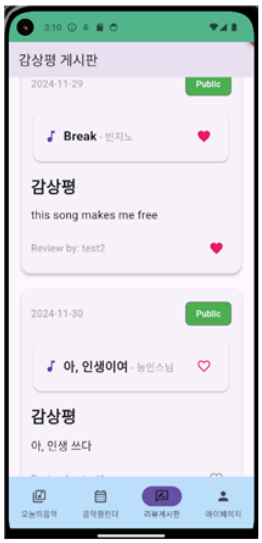
작성한 감상평이 없을 시



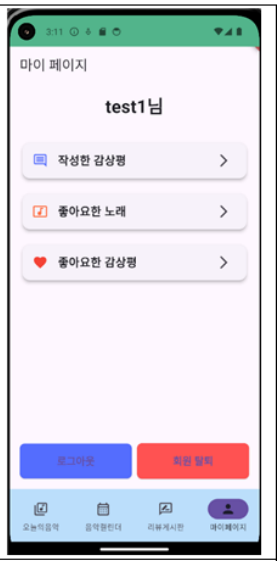
감상평 게시판 화면



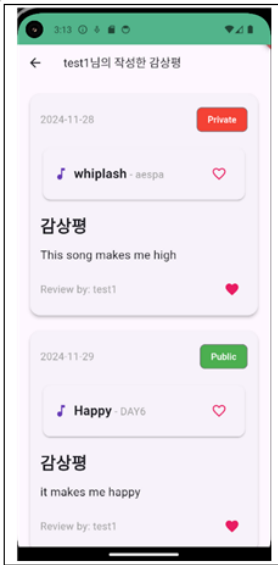
노래 좋아요



리뷰 좋아요



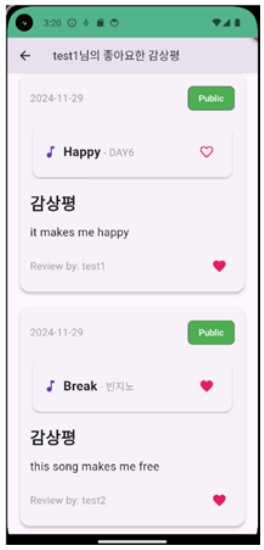
마이페이지 기본화면



작성한 감상평 조회

좋아요한 노래 조회

노래 좋아요 취소



좋아요한 감상평 조회