비디오 게임 평점, 정보 검색 서비스 개발 포트폴리오

김재구
Tel.010-7247-7646
kimwithglasses@kakao.com

목차

소스코드/앱 링크

- 소스코드 :
  - Front End : https://github.com/JaeguKim/GameTodoey
  - Back End : https://github.com/JaeguKim/GameTodoey-Backend
- 앱 링크
  - https://apps.apple.com/kr/app/gametodoey/id1507663102?l=en

Language/Framework

Front End

Swift

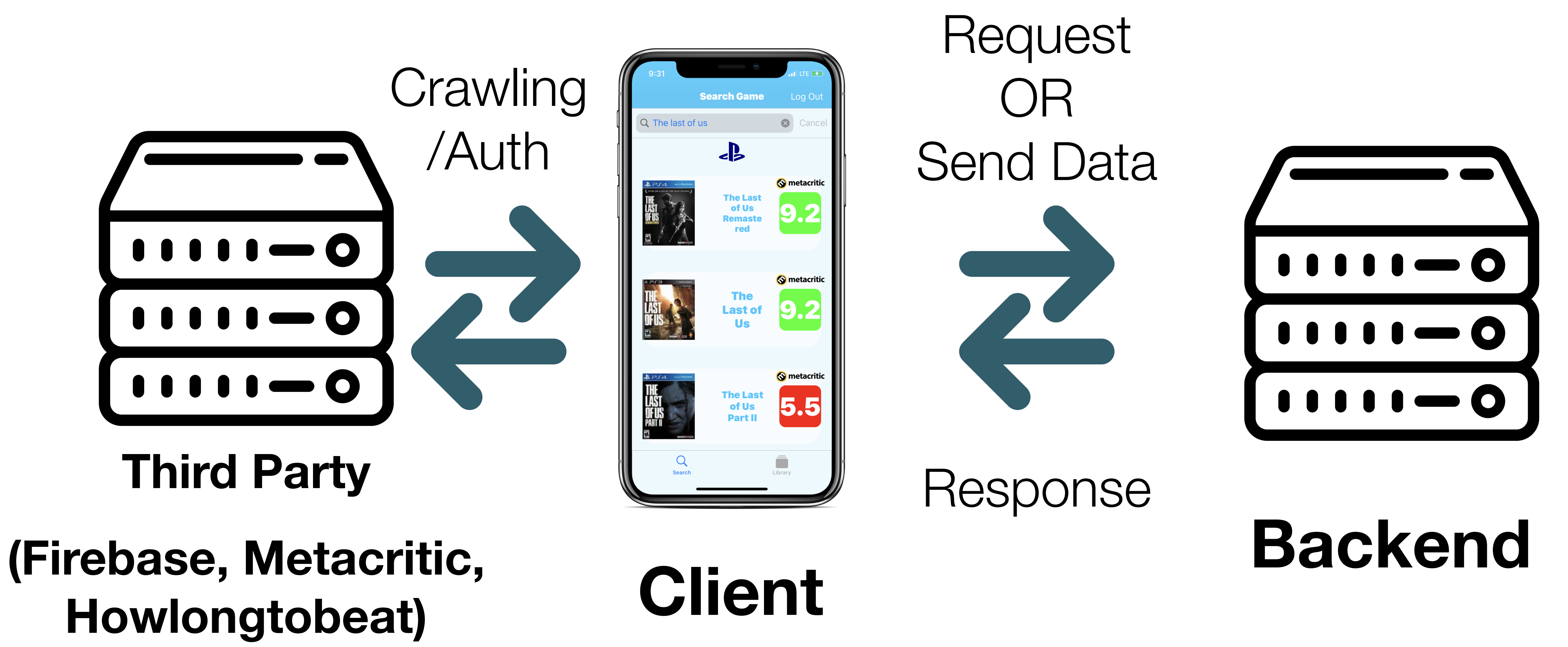
Back End

spring

MySQL™

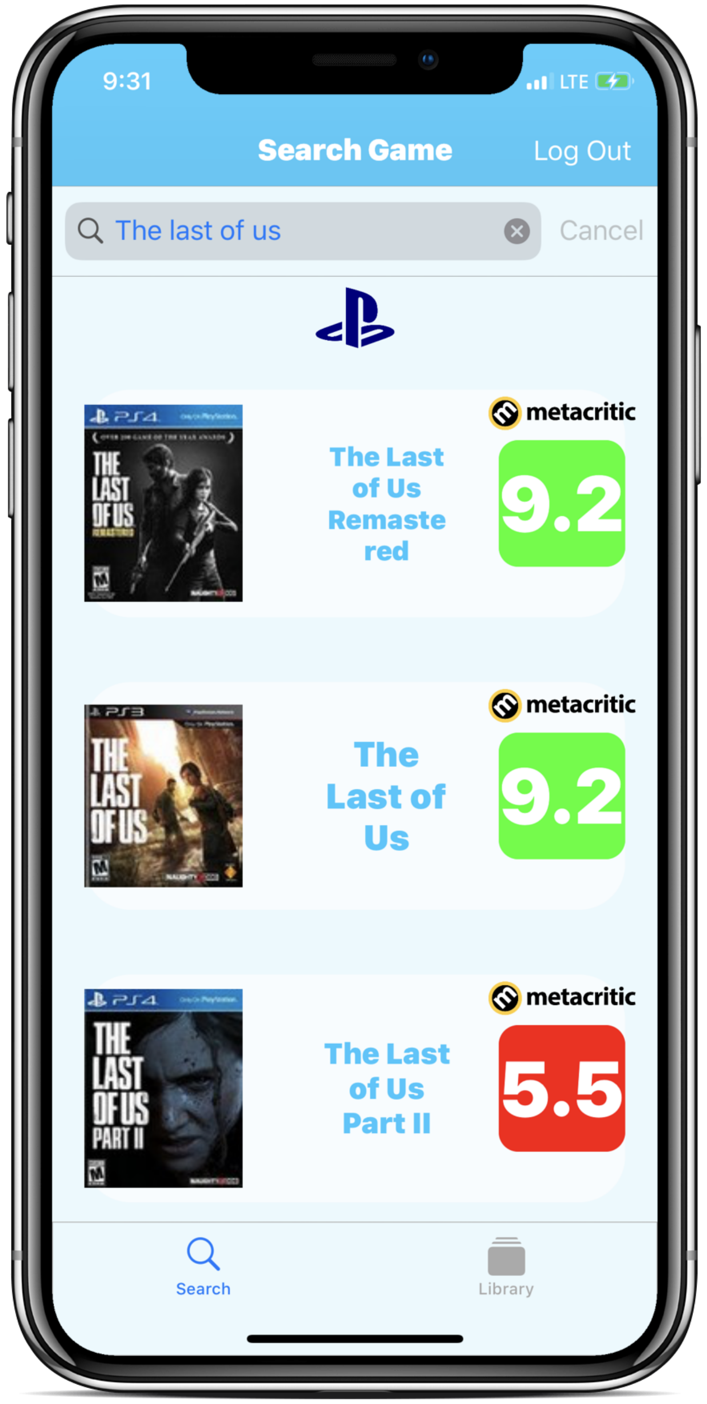
Java™

전체 서비스 아키텍쳐

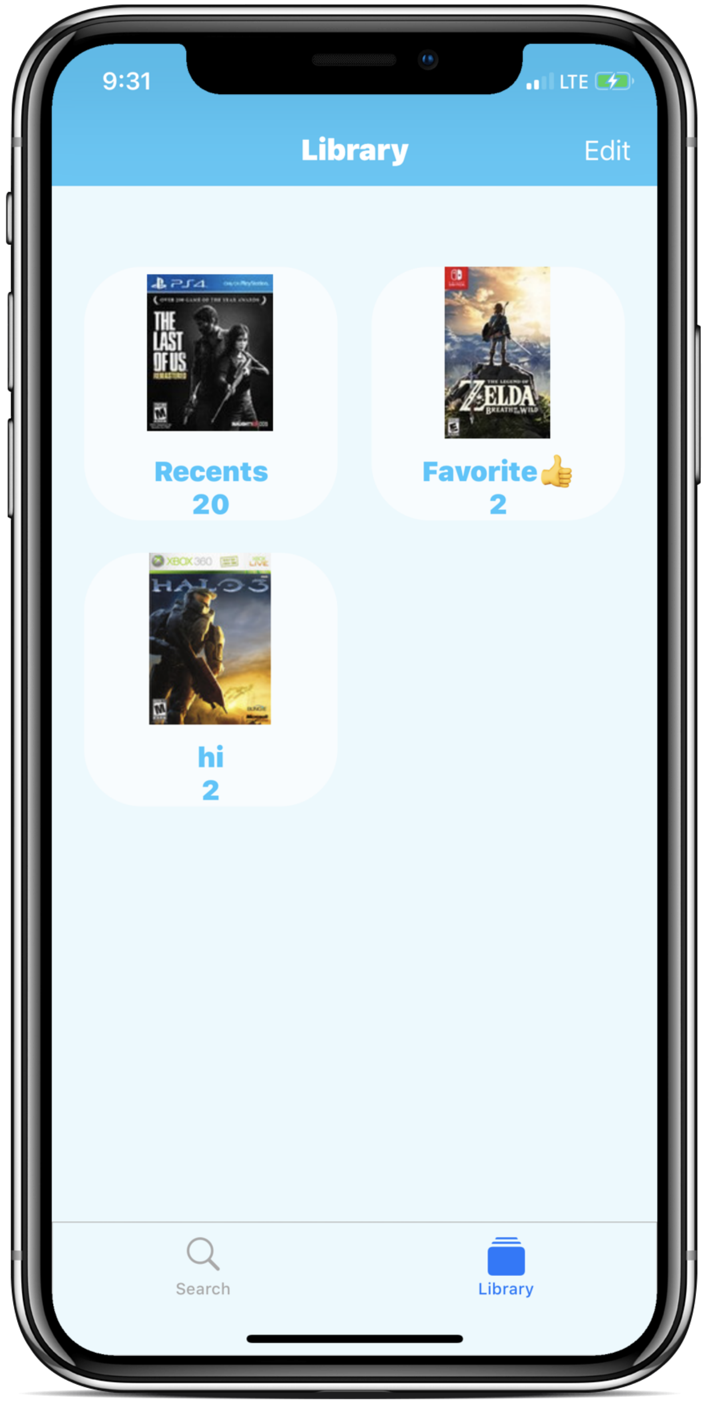

Crawling
/Auth
Request
OR
Send Data
Response
Third Party
(Firebase, Metacritic, Howlongtobeat)
Client
Backend
Search Game
Log Out
The last of us
Cancel
The Last of Us Remastered
9.2
The Last of Us
9.2
The Last of Us Part II
5.5
metacritic
Search
Library

Frontend 실행화면

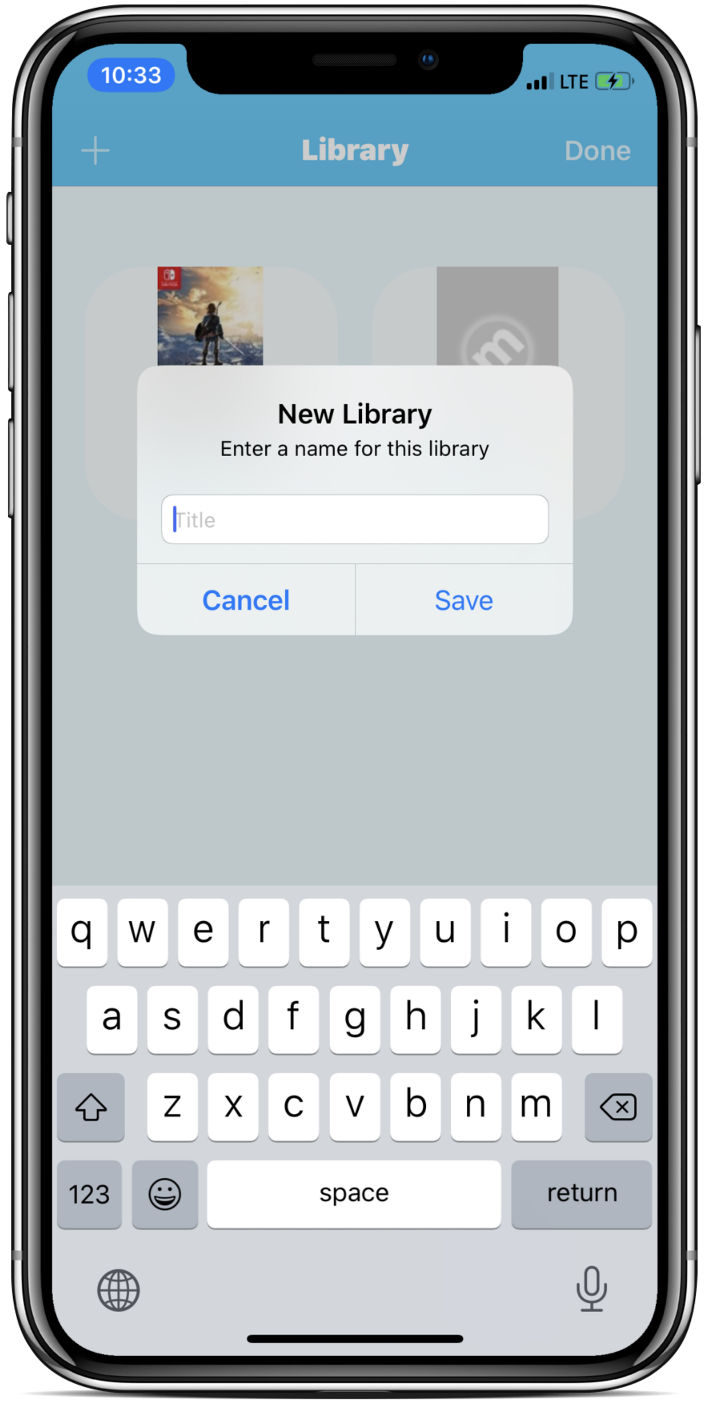
게임검색



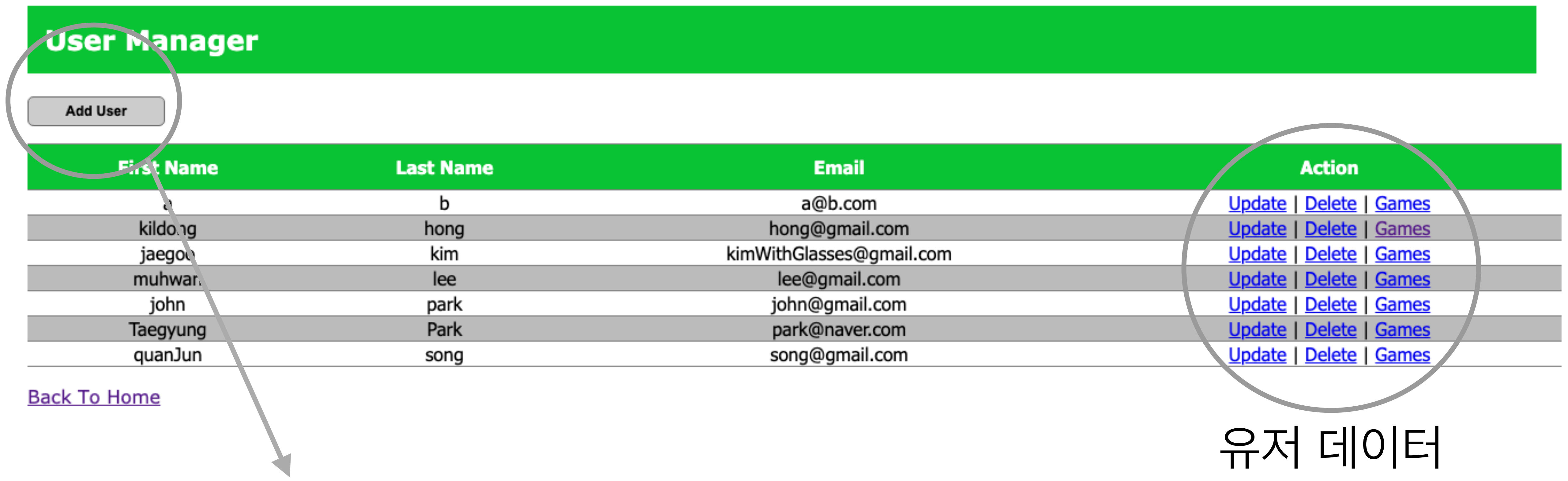
라이브러리



라이브러리 생성



Backend Admin 실행화면 (유저 조회)



User Manager

Add User

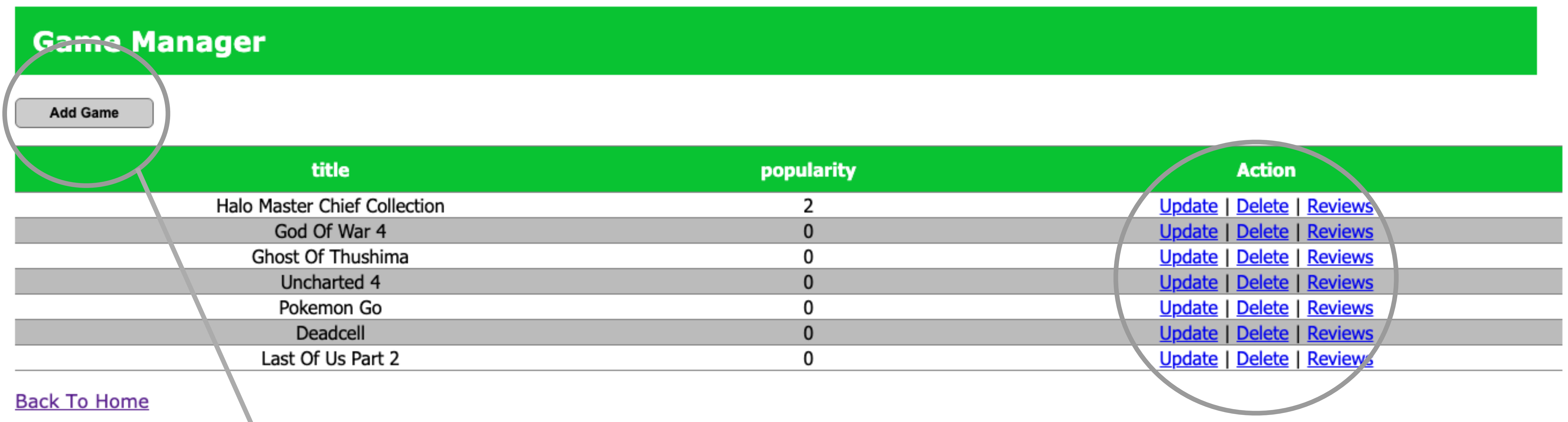
First Name	Last Name	Email	Action
a	b	a@b.com	Update | Delete | Games
kildong	hong	hong@gmail.com	Update | Delete | Games
jaegoo	kim	kimWithGlasses@gmail.com	Update | Delete | Games
muhwan	lee	lee@gmail.com	Update | Delete | Games
john	park	john@gmail.com	Update | Delete | Games
Taegyung	Park	park@naver.com	Update | Delete | Games
quanJun	song	song@gmail.com	Update | Delete | Games

Back To Home

새로운 유저 추가

유저 데이터
변경/삭제/게임 조회

Backend Admin 실행화면 (게임 조회)



Game Manager

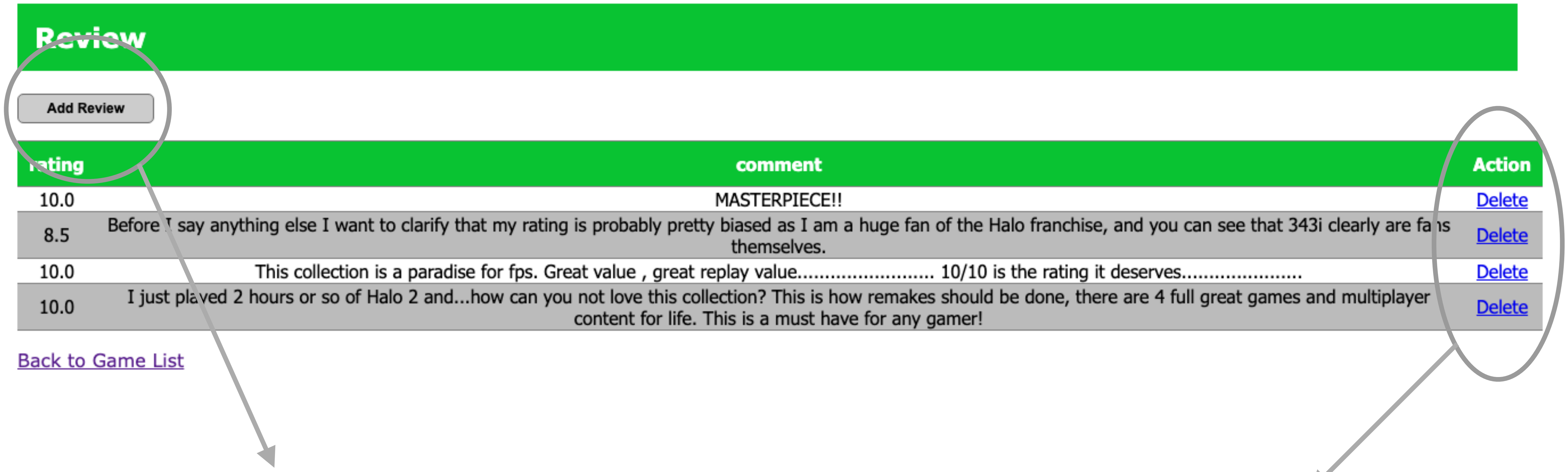
title	popularity	Action
Halo Master Chief Collection	2	Update | Delete | Reviews
God Of War 4	0	Update | Delete | Reviews
Ghost Of Thushima	0	Update | Delete | Reviews
Uncharted 4	0	Update | Delete | Reviews
Pokemon Go	0	Update | Delete | Reviews
Deadcell	0	Update | Delete | Reviews
Last Of Us Part 2	0	Update | Delete | Reviews

Back To Home

새게임 추가

게임 데이터
변경/삭제/리뷰 조회

Backend Admin 실행화면 (특정 게임에 대한 리뷰 조회)

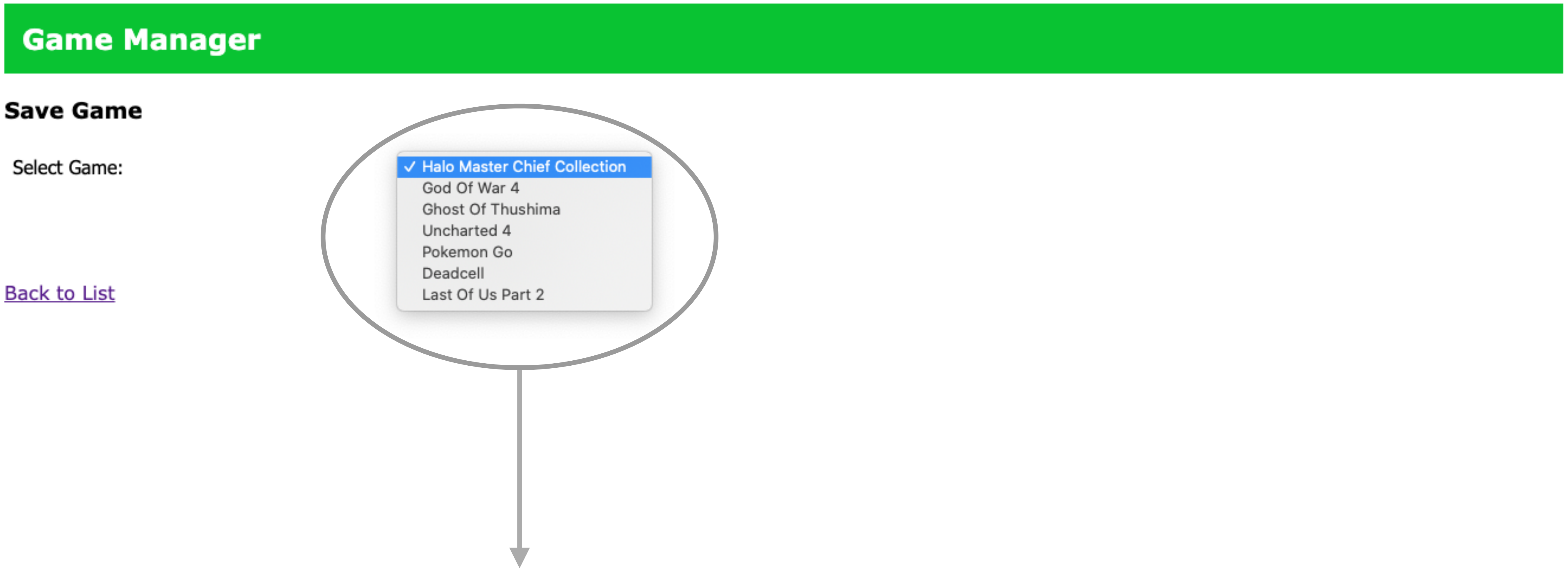


rating	comment	Action
10.0	MASTERPIECE!!	Delete
8.5	Before I say anything else I want to clarify that my rating is probably pretty biased as I am a huge fan of the Halo franchise, and you can see that 343i clearly are fans themselves.	Delete
10.0	This collection is a paradise for fps. Great value , great replay value........................ 10/10 is the rating it deserves.....................	Delete
10.0	I just played 2 hours or so of Halo 2 and...how can you not love this collection? This is how remakes should be done, there are 4 full great games and multiplayer content for life. This is a must have for any gamer!	Delete

새리뷰 추가

리뷰 삭제

Backend Admin 실행화면 (보유 게임 추가)



Game Manager

Save Game

Select Game:

✓ Halo Master Chief Collection
God Of War 4
Ghost Of Thushima
Uncharted 4
Pokemon Go
Deadcell
Last Of Us Part 2

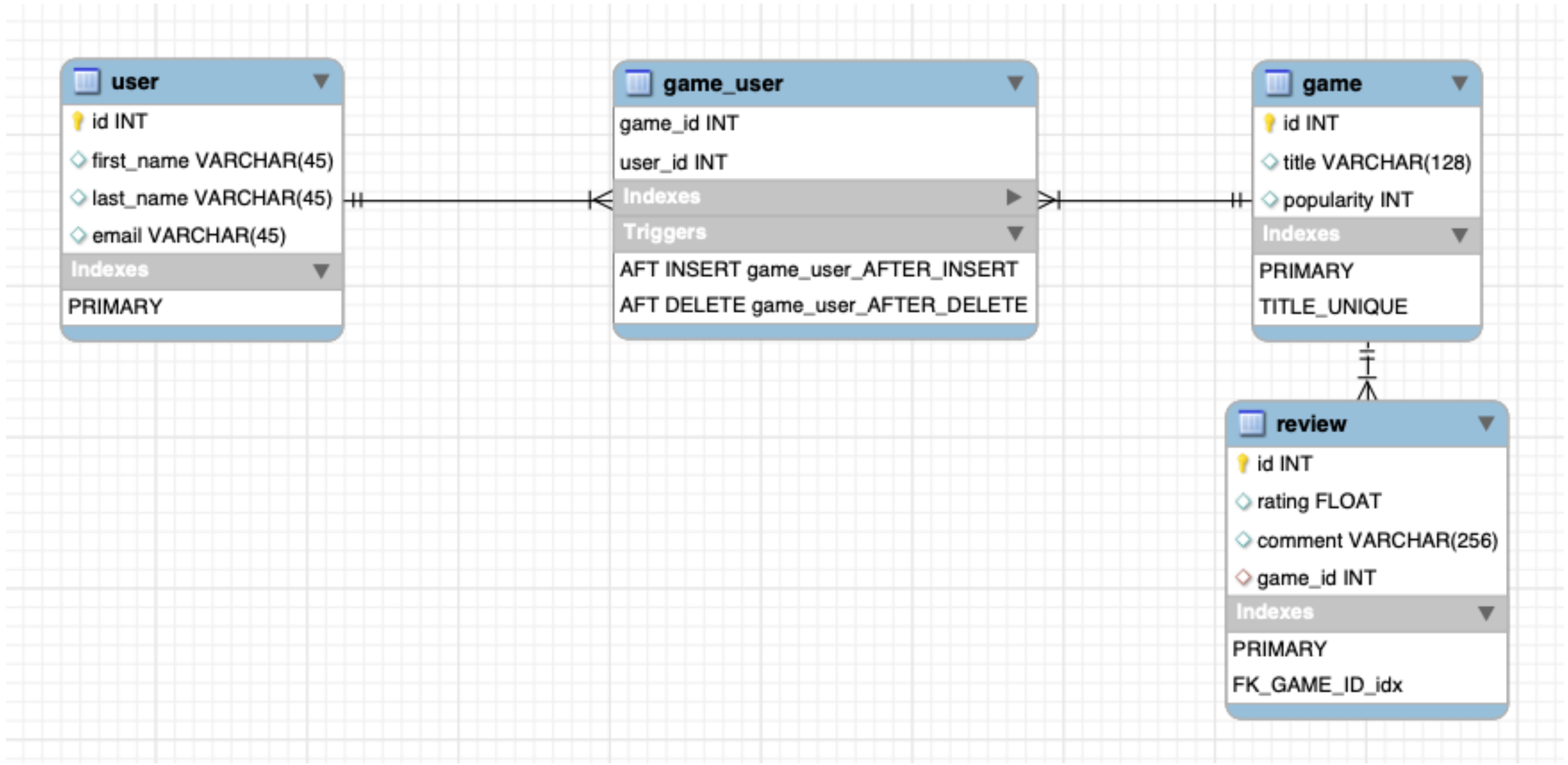
Back to List

추가할 게임 선택

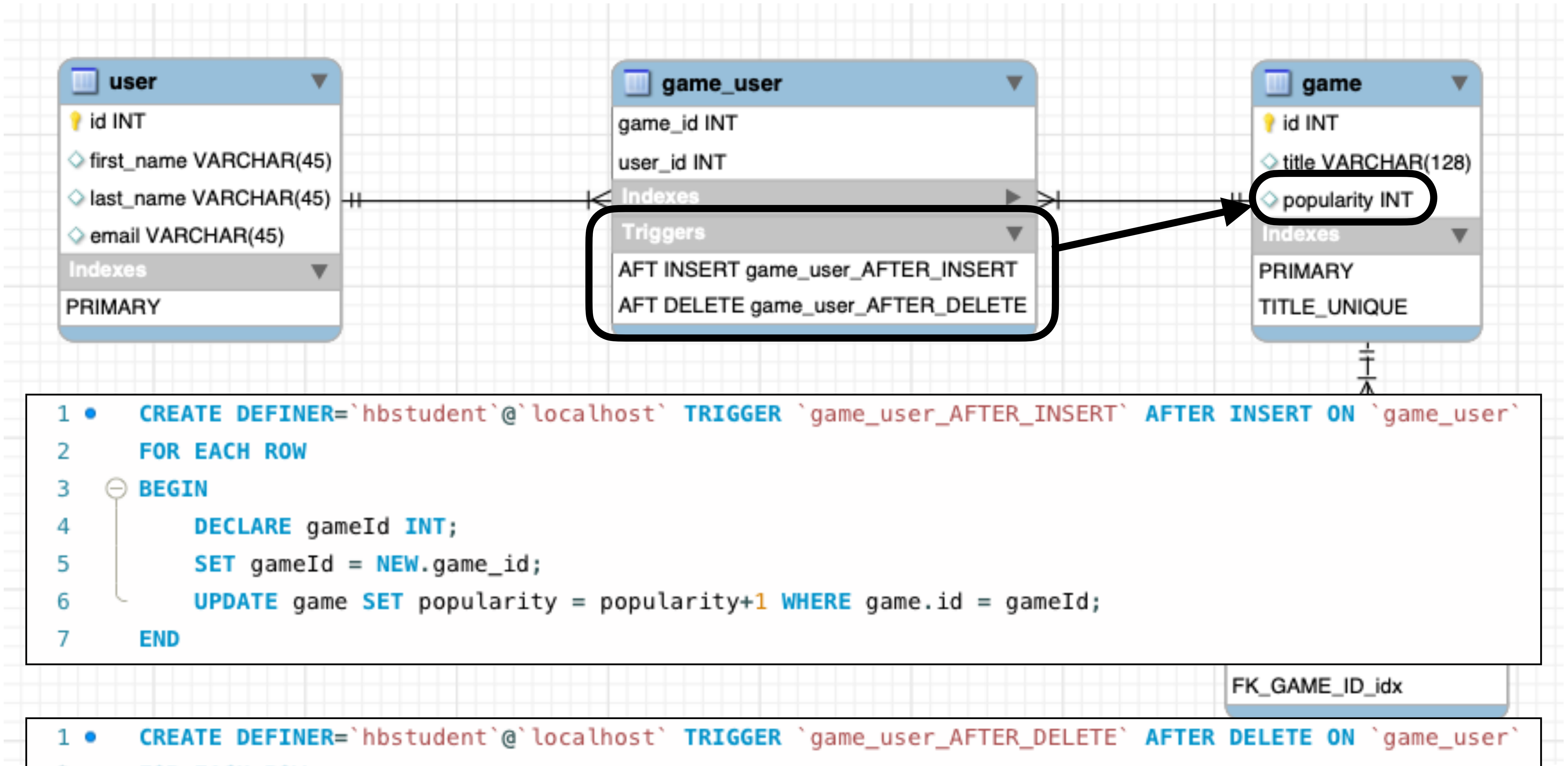
사용된 Spring 버전 및 dependency

- **Spring Framework - 5.0.6 RELEASE**
- **Hibernate - 5.4.1 Final**
- **MySQL - 5.1.45**
- **c3Po - 0.9.5.2**
- **maven - 1.8**
- **Java - 1.8**

ER Diagram


user
id INT
first_name VARCHAR(45)
last_name VARCHAR(45)
email VARCHAR(45)
Indexes
PRIMARY
game_user
game_id INT
user_id INT
Indexes
Triggers
AFT INSERT game_user_AFTER_INSERT
AFT DELETE game_user_AFTER_DELETE
game
id INT
title VARCHAR(128)
popularity INT
Indexes
PRIMARY
TITLE_UNIQUE
review
id INT
rating FLOAT
comment VARCHAR(256)
game_id INT
Indexes
PRIMARY
FK_GAME_ID_idx

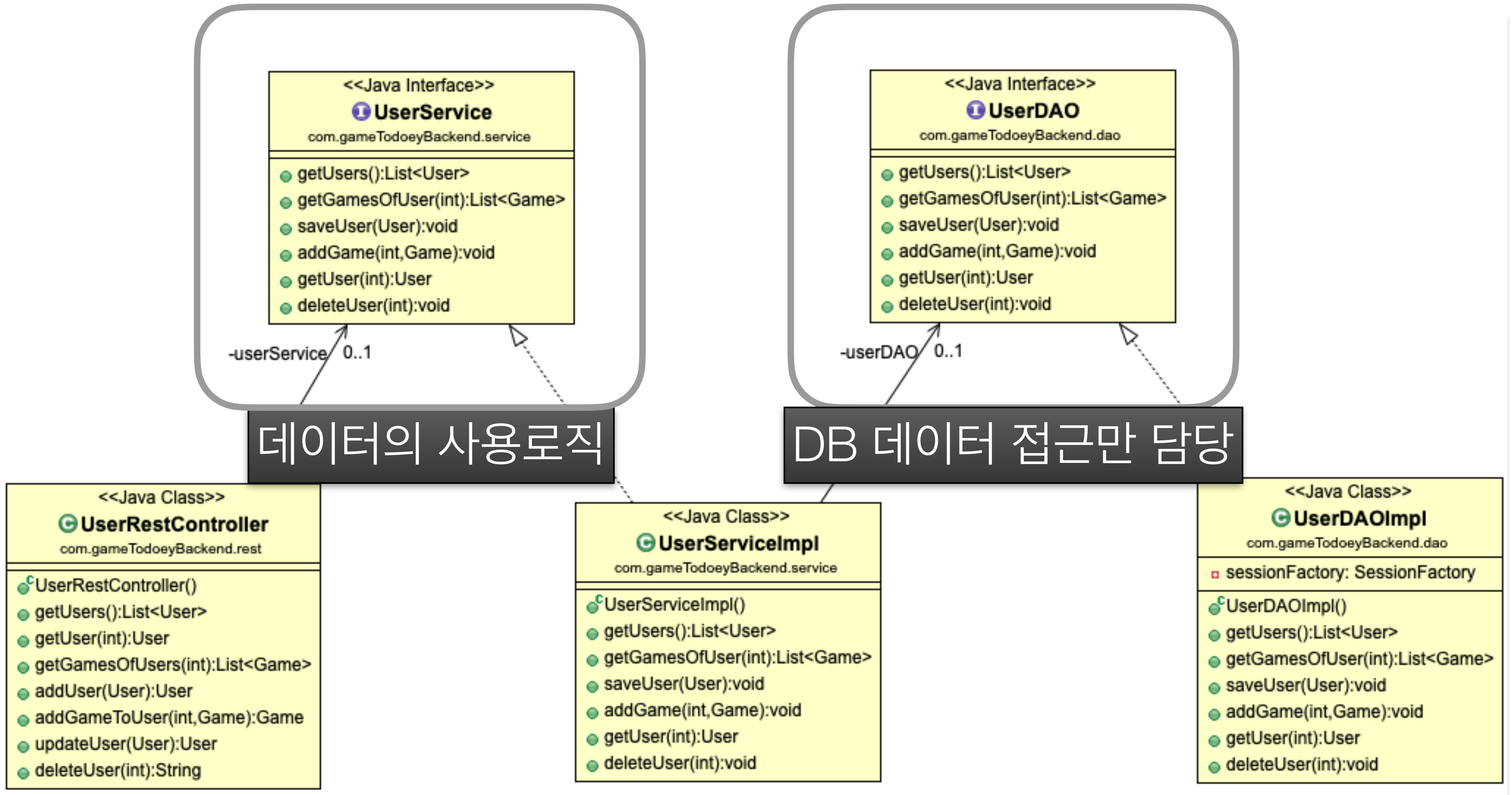
Trigger
(유저가 게임을 추가/삭제할시 자동으로 인기도 업데이트)



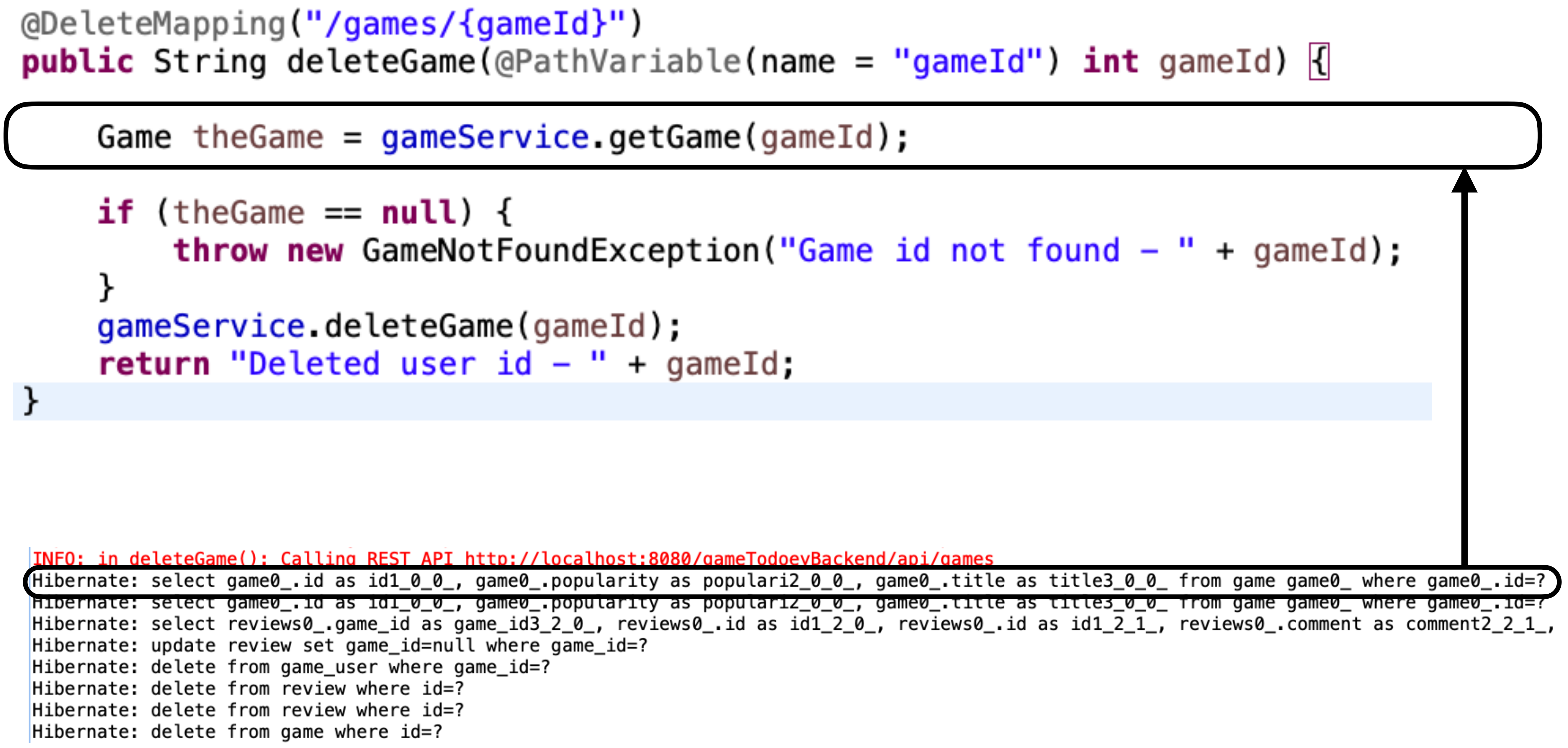
```sql
CREATE DEFINER=`hbstudent`@`localhost` TRIGGER `game_user_AFTER_INSERT` AFTER INSERT ON `game_user`
FOR EACH ROW
BEGIN
    DECLARE gameId INT;
    SET gameId = NEW.game_id;
    UPDATE game SET popularity = popularity+1 WHERE game.id = gameId;
END
```

```sql
CREATE DEFINER=`hbstudent`@`localhost` TRIGGER `game_user_AFTER_DELETE` AFTER DELETE ON `game_user`
FOR EACH ROW
BEGIN
    DECLARE gameId INT;
    SET gameId = OLD.game_id;
    UPDATE game SET popularity = popularity-1 WHERE game.id = gameId;
END
```

UML Diagram (유저 예시)



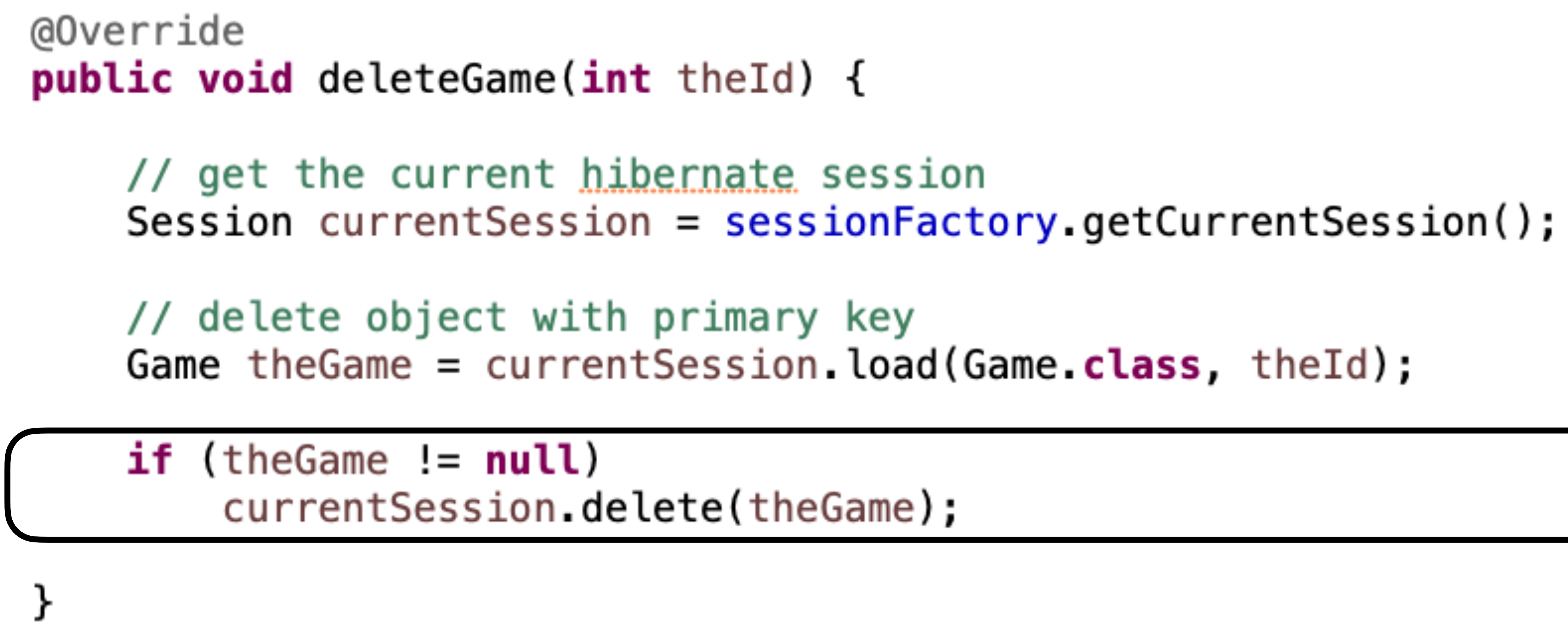
Game 레코드 삭제시 Hibernate 내부 동작



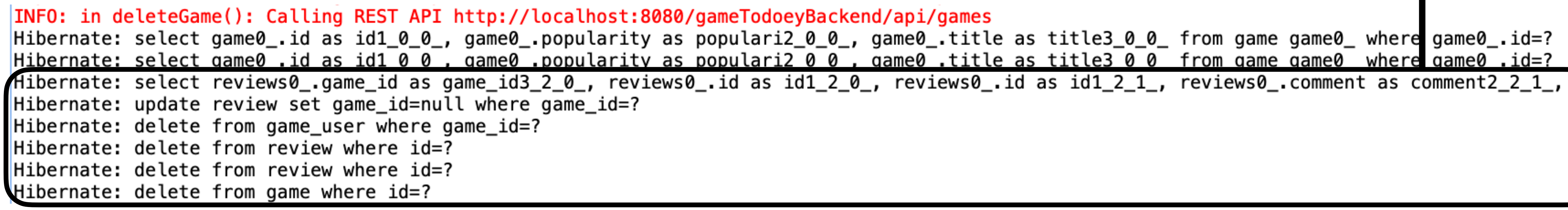
```java
@DeleteMapping("/games/{gameId}")
public String deleteGame(@PathVariable(name = "gameId") int gameId) {

    Game theGame = gameService.getGame(gameId);

    if (theGame == null) {
        throw new GameNotFoundException("Game id not found - " + gameId);
    }
    gameService.deleteGame(gameId);
    return "Deleted user id - " + gameId;
}
```

```
INFO: in deleteGame(): Calling REST API http://localhost:8080/gameTodoeyBackend/api/games
Hibernate: select game0_.id as id1_0_0_, game0_.popularity as populari2_0_0_, game0_.title as title3_0_0_ from game game0_ where game0_.id=?
Hibernate: select game0_.id as id1_0_0_, game0_.popularity as populari2_0_0_, game0_.title as title3_0_0_ from game game0_ where game0_.id=?
Hibernate: select reviews0_.game_id as game_id3_2_0_, reviews0_.id as id1_2_0_, reviews0_.id as id1_2_1_, reviews0_.comment as comment2_2_1_,
Hibernate: update review set game_id=null where game_id=?
Hibernate: delete from game_user where game_id=?
Hibernate: delete from review where id=?
Hibernate: delete from review where id=?
Hibernate: delete from game where id=?
```

Game 레코드 삭제시 Hibernate 내부 동작



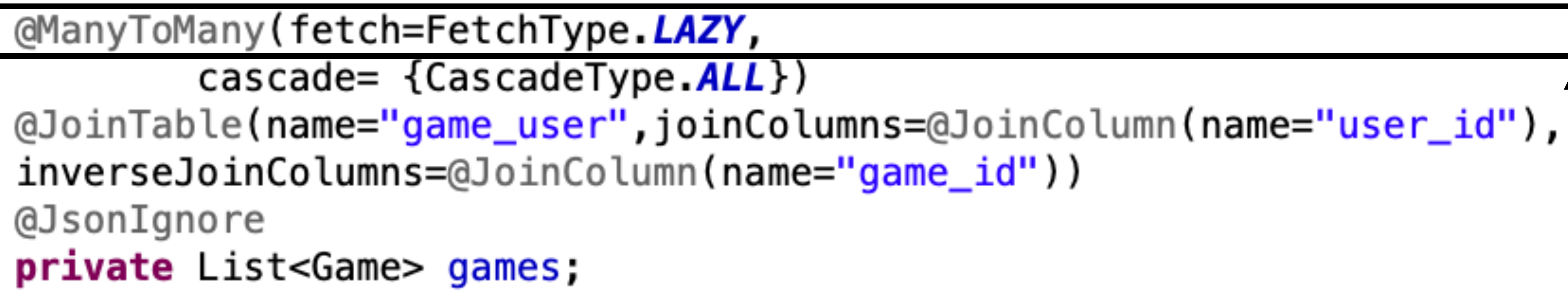
```java
@Override
public void deleteGame(int theId) {

    // get the current hibernate session
    Session currentSession = sessionFactory.getCurrentSession();

    // delete object with primary key
    Game theGame = currentSession.load(Game.class, theId);

    if (theGame != null)
        currentSession.delete(theGame);

}
```



```
INFO: in deleteGame(): Calling REST API http://localhost:8080/gameTodoeyBackend/api/games
Hibernate: select game0_.id as id1_0_0_, game0_.popularity as populari2_0_0_, game0_.title as title3_0_0_ from game game0_ where game0_.id=?
Hibernate: select game0_.id as id1_0_0_, game0_.popularity as populari2_0_0_, game0_.title as title3_0_0_ from game game0_ where game0_.id=?
Hibernate: select reviews0_.game_id as game_id3_2_0_, reviews0_.id as id1_2_0_, reviews0_.id as id1_2_1_, reviews0_.comment as comment2_2_1_,
Hibernate: update review set game_id=null where game_id=?
Hibernate: delete from game_user where game_id=?
Hibernate: delete from review where id=?
Hibernate: delete from review where id=?
Hibernate: delete from game where id=?
```

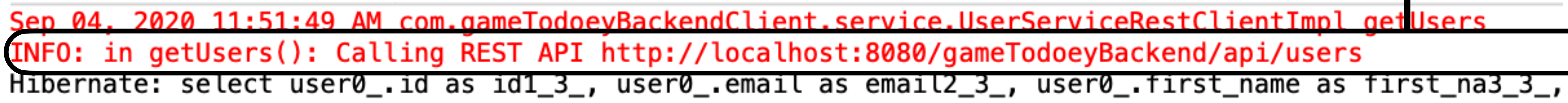
연관된 Review 삭제, User와의 JoinTable 데이터 삭제 쿼리 생성

User 게임 레코드 조회시 Hibernate 내부 동작



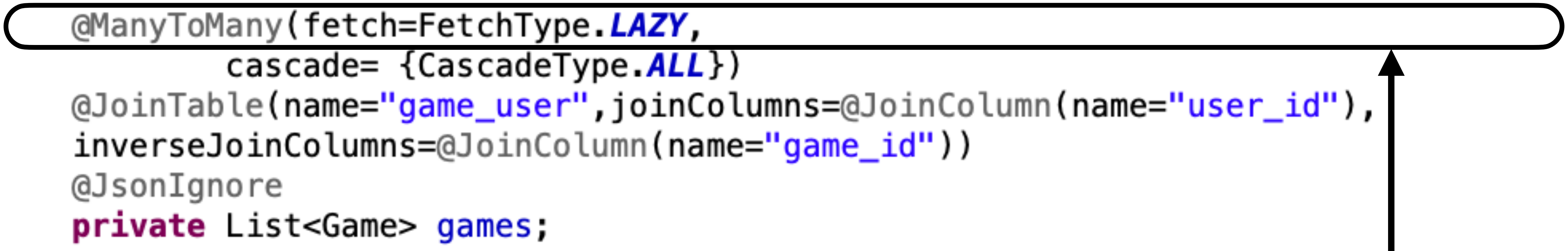
```
@ManyToMany(fetch=FetchType.LAZY,
        cascade= {CascadeType.ALL})
@JoinTable(name="game_user",joinColumns=@JoinColumn(name="user_id"),
inverseJoinColumns=@JoinColumn(name="game_id"))
@JsonIgnore
private List<Game> games;
```

fetch 타입이 LAZY 이므로
유저정보에 대한 조회가 일어날때, 게임정보를 조회하기 위한 별도의 쿼리가 발생하지 않는다.



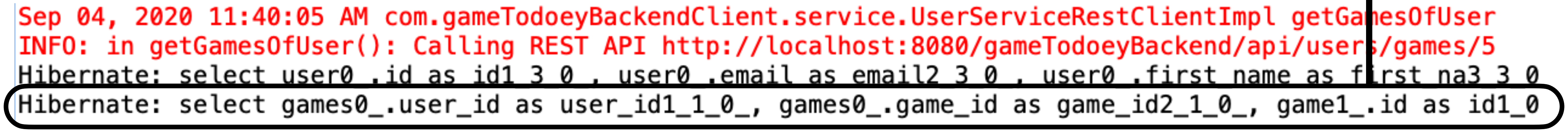
```
Sep 04, 2020 11:51:49 AM com.gameTodoeyBackendClient.service.UserServiceRestClientImpl getUsers
INFO: in getUsers(): Calling REST API http://localhost:8080/gameTodoeyBackend/api/users
Hibernate: select user0_.id as id1_3_, user0_.email as email2_3_, user0_.first_name as first_na3_3_,
```

User 게임 레코드 조회시 Hibernate 내부 동작



```
@ManyToMany(fetch=FetchType.LAZY,
        cascade= {CascadeType.ALL})
@JoinTable(name="game_user",joinColumns=@JoinColumn(name="user_id"),
inverseJoinColumns=@JoinColumn(name="game_id"))
@JsonIgnore
private List<Game> games;
```

fetch 타입이 LAZY 이므로
유저게임 정보에 대한 조회가 일어날때,
game_join 테이블과 game 테이블의 inner join이 발생



```
Sep 04, 2020 11:40:05 AM com.gameTodoeyBackendClient.service.UserServiceRestClientImpl getGamesOfUser
INFO: in getGamesOfUser(): Calling REST API http://localhost:8080/gameTodoeyBackend/api/users/games/5
Hibernate: select user0_.id as id1_3_0_, user0_.email as email2_3_0_, user0_.first_name as first_na3_3_0
Hibernate: select games0_.user_id as user_id1_1_0_, games0_.game_id as game_id2_1_0_, game1_.id as id1_0
```