4조 Spring 프로젝트

목차

1. 개발배경 및 개요

2. 개발 구성

3. 화면 구현 및 기능 구현

후기 및 Q&A

stage.1

개발 배경 및 개요





개발 배경발배경

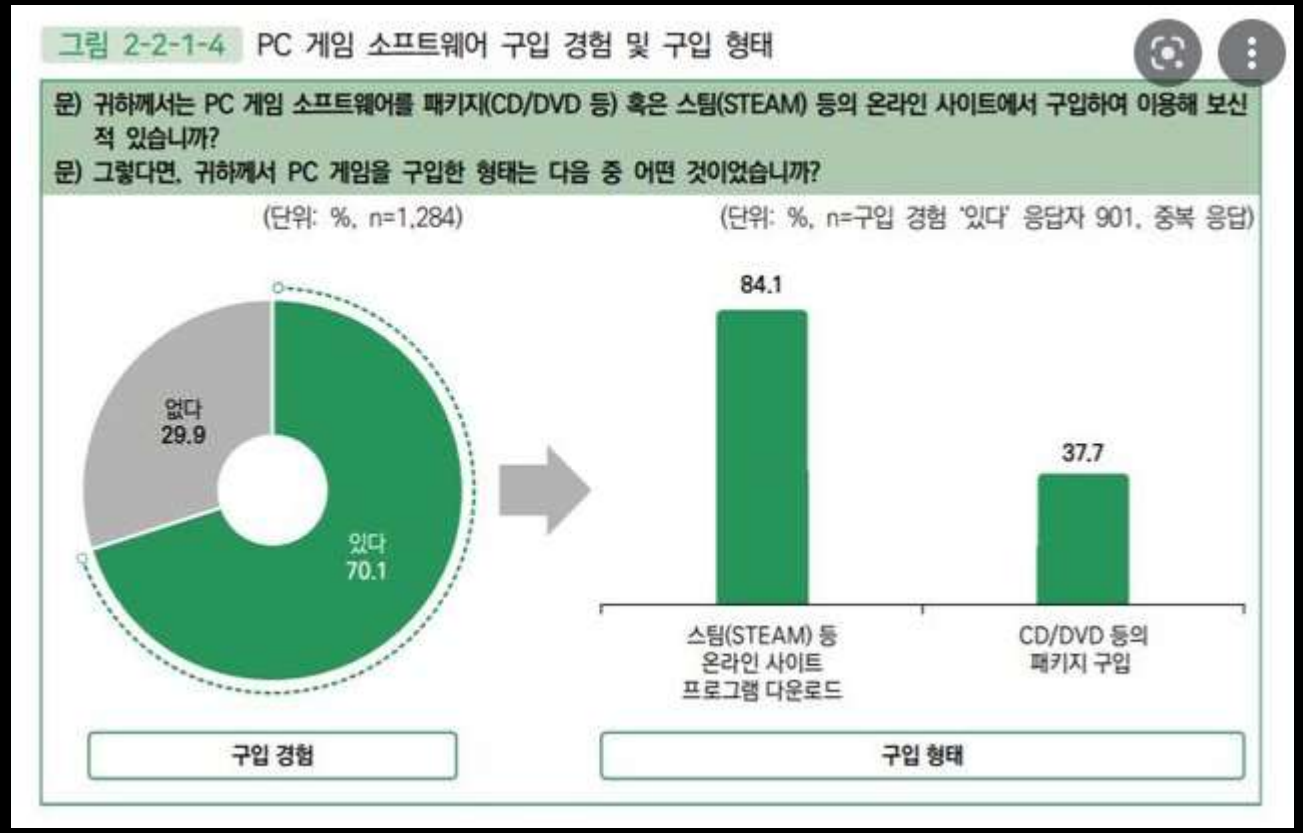
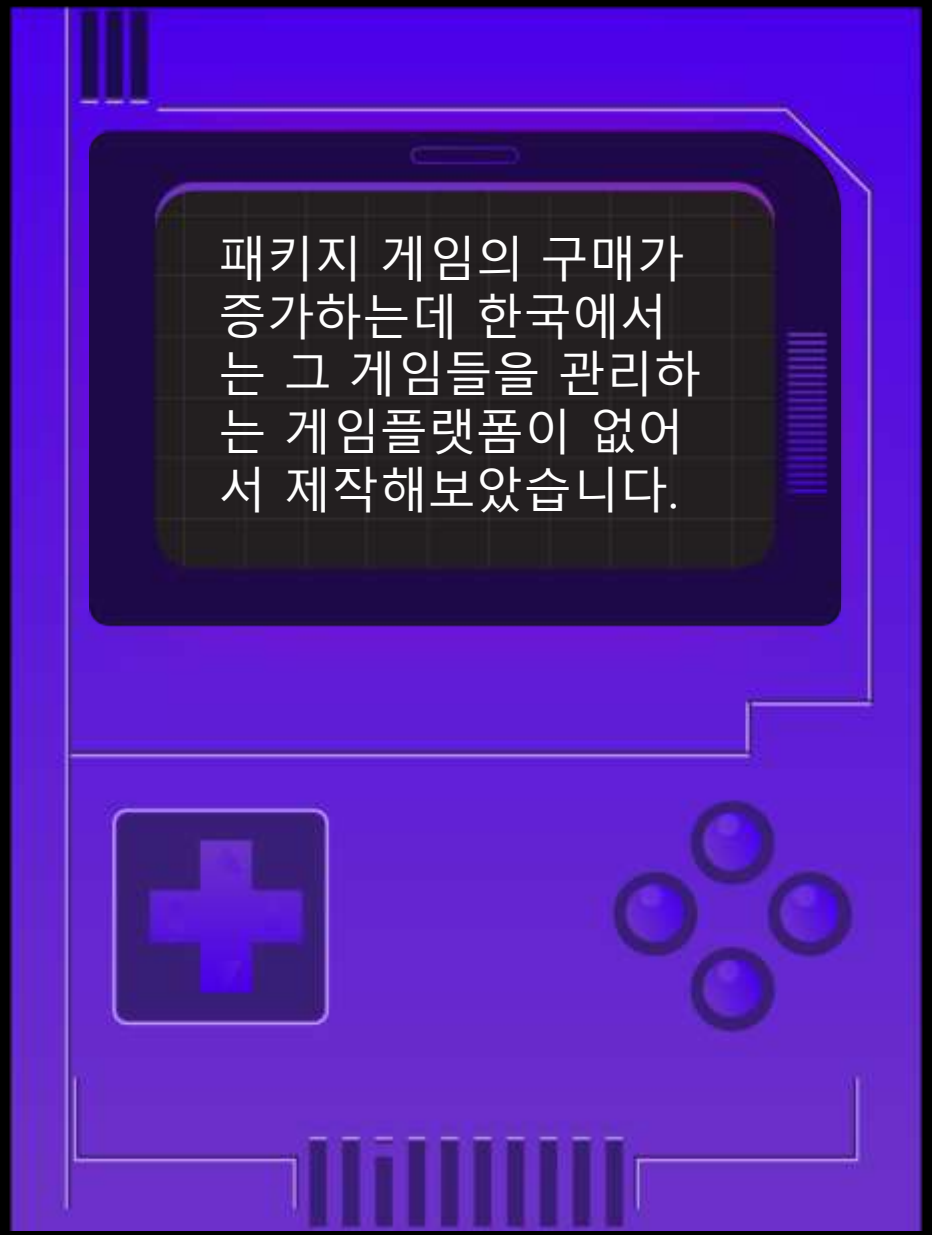


그림 2-2-1-4 PC 게임 소프트웨어 구입 경험 및 구입 형태
문) 귀하께서는 PC 게임 소프트웨어를 패키지(CD/DVD 등) 혹은 스팀(STEAM) 등의 온라인 사이트에서 구입하여 이용해 보신 적 있습니까?
문) 그렇다면, 귀하께서 PC 게임을 구입한 형태는 다음 중 어떤 것이었습니까?
(단위: %, n=1,284)
(단위: %, n=구입 경험 '있다' 응답자 901, 중복 응답)
없다 29.9
있다 70.1
84.1
37.7
스팀(STEAM) 등 온라인 사이트 프로그램 다운로드
CD/DVD 등의 패키지 구입
구입 경험
구입 형태


패키지 게임의 구매가 증가하는데 한국에서는 그 게임들을 관리하는 게임플랫폼이 없어서 제작해보았습니다.

개요

프로젝트	GAME(게임 플랫폼 사이트)	
개발기간	2022. 03. 28 ~ 2022. 04. 08 (14일)	
개발인원	3명	
참여자	정재규, 구주용, 김경래	
프로젝트 요약	게임플랫폼 제작 및 관리	
개발환경	운영체제	Window 10 64Bit
	개발언어	JAVA, HTML, CSS, JavaScript, JQuery, JSON
	개발도구	SpringToolSuite4, MySQL, Maven
	데이터베이스	MySQL
	프레임워크	SpringBoot, JPA

stage.2

개발 구성

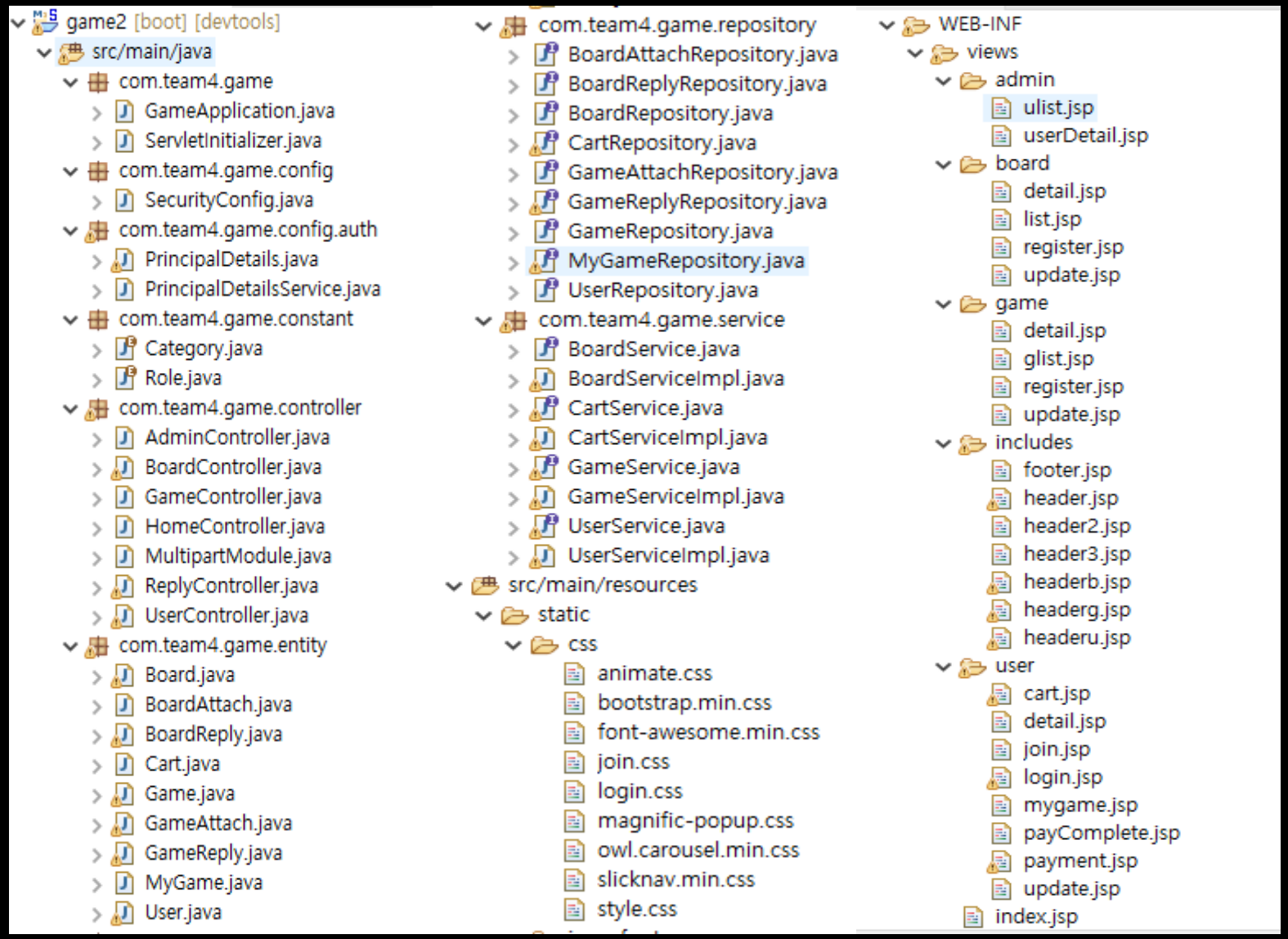




구조

비회원	회원가입 게시판(보기, 검색) 게임(보기, 검색)
회원	로그인 게시판(보기, 검색, 댓글, 글쓰기, 수정, 삭제) 게임(보기, 검색, 평점리뷰, 장바구니, 구매, 내 게임[구매] 목록) 회원정보(보기, 수정, 탈퇴)
관리자	회원관리(목록, 보기, 삭제) 게시글삭제 게임관리(등록, 수정, 삭제)

구조



프로젝트를 구현하기 위해 사용한 패키지와 폴더배치 구조입니다.
java , jsp, css, js
pom.xml 파일입니다.

작업환경

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-tomcat</artifactId>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-test</artifactId>
            <scope>test</scope>
        </dependency>
                <!-- https://mvnrepository.com/artifact/org.apache.tomcat.embed/tomcat-embed-jasper -->
        <dependency>
            <groupId>org.apache.tomcat.embed</groupId>
            <artifactId>tomcat-embed-jasper</artifactId>
        </dependency>
        <!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jstl</artifactId>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.springframework.security/spring-security-taglibs -->
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-taglibs</artifactId>
        </dependency>

        <!-- querydsl -->
        <dependency>
            <groupId>com.querydsl</groupId>
            <artifactId>querydsl-jpa</artifactId>
        </dependency>

        <dependency>
            <groupId>com.querydsl</groupId>
            <artifactId>querydsl-apt</artifactId>
        </dependency>
```

New SPring Start Project

작업환경을 설정해주었으며 , Lombok,MySQl Driver Spring Security 등 사용.

작업환경

```
server.port=8081
spring.mvc.view.prefix=/WEB-INF/views/
spring.mvc.view.suffix=.jsp

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/gamedb?useSSL=false&serverTimezon=Asia/Seoul&&characterEncoding=UTF-8
spring.datasource.username=koo
spring.datasource.password=1234

#LAZY loading erreor
spring.jackson.serialization.fail-on-empty-beans=false

spring.jpa.properties.hibernate.show_sql=true

#spring.jpa.properties.hibernate.format_sql=true
#logging.level.org.hibernate.type.descriptor.sql=trace

spring.jpa.hibernate.ddl-auto=update
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
```

1. MySQL DB와 연결해주는 작업
2. hibernate 사용 - 자바 언어를 위한 ORM 프레임워크이며, JPA의 구현체 JPA 인터페이스를 구현하며, 내부적으로 JDBC API를 사용합니다.

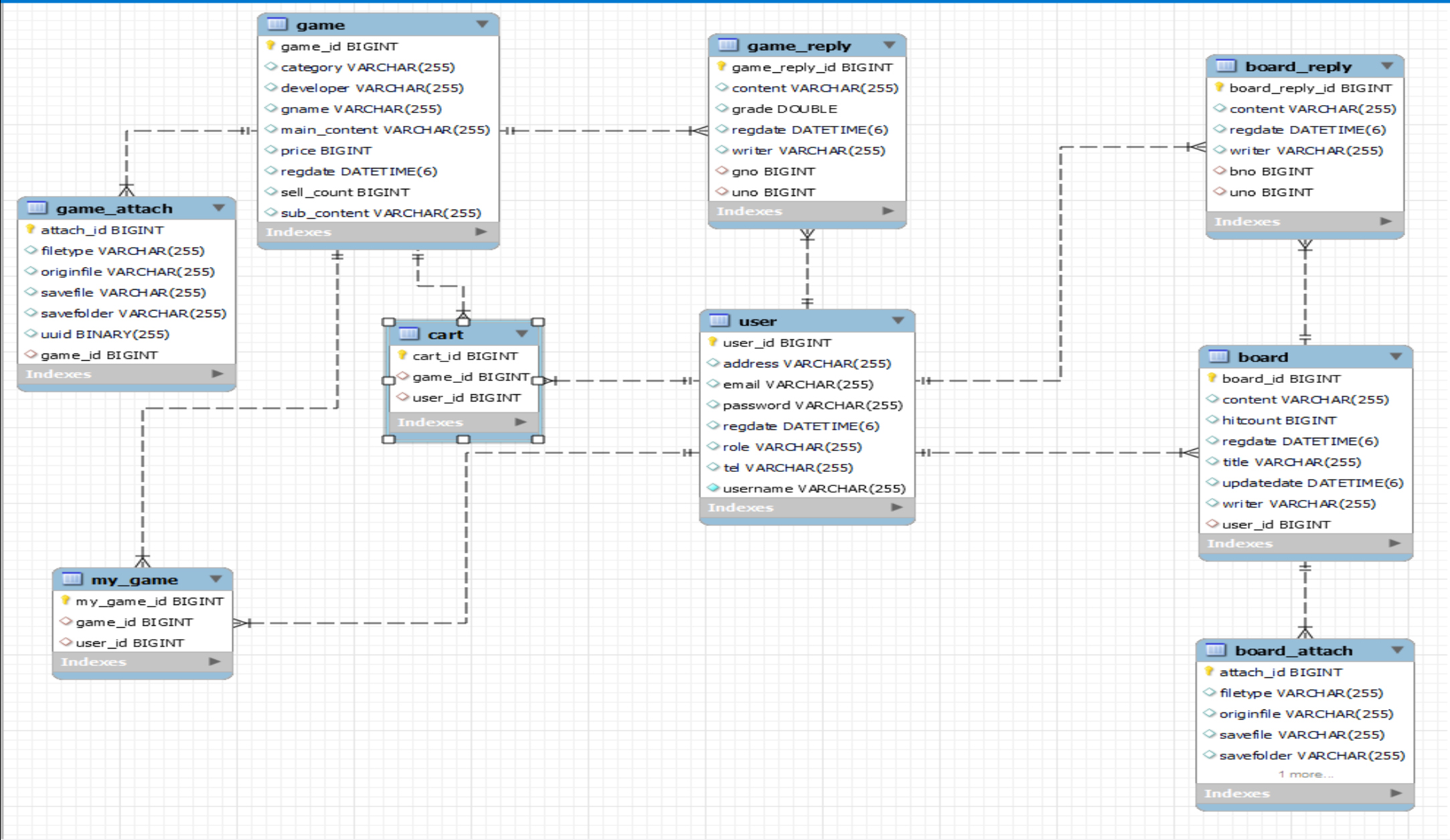
Entity - (board, cart, game, user.....)

```java
package com.team4.game.entity;

import java.util.Date;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Board {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="board_id")
    private Long id;

    private String title;
    private String writer;
    private String content;

    private Long hitcount;
    @CreationTimestamp
    @Temporal(TemporalType.TIMESTAMP)
    private Date regdate;
    @CreationTimestamp
    @Temporal(TemporalType.TIMESTAMP)
    private Date updatedate;

    @OneToMany(mappedBy = "board",fetch=FetchType.LAZY ,cascade = CascadeType.ALL)
    @JsonIgnoreProperties("board") // 무한참조 방지
    private List<BoardReply> boardReply;

    @ManyToOne(fetch=FetchType.LAZY)
    @JoinColumn(name="user_id")
    private User user;

    @OrderBy("id desc")
    @OneToMany(mappedBy = "board",fetch=FetchType.LAZY ,cascade = CascadeType.ALL)
    @JsonIgnoreProperties("board") // 무한참조 방지
    private List<BoardAttach> boardAttachs;

    @PrePersist
    public void prePersist() {
        this.hitcount= this.hitcount==null? 0 : this.hitcount;
    }

}
```

1. Lombok을 사용한 @어노테이션을 사용하여 DB에 테이블생성 및 객체 생성.

2. @OneToMany, @ManyToOne 사용하여 다른 테이블과의 참조 관계 형성.

MySQL - ERD


game
game_id BIGINT
category VARCHAR(255)
developer VARCHAR(255)
gname VARCHAR(255)
main_content VARCHAR(255)
price BIGINT
regdate DATETIME(6)
sell_count BIGINT
sub_content VARCHAR(255)
Indexes
game_reply
game_reply_id BIGINT
content VARCHAR(255)
grade DOUBLE
regdate DATETIME(6)
writer VARCHAR(255)
gno BIGINT
uno BIGINT
Indexes
board_reply
board_reply_id BIGINT
content VARCHAR(255)
regdate DATETIME(6)
writer VARCHAR(255)
bno BIGINT
uno BIGINT
Indexes
game_attach
attach_id BIGINT
filetype VARCHAR(255)
originfile VARCHAR(255)
savefile VARCHAR(255)
savefolder VARCHAR(255)
uuid BINARY(255)
game_id BIGINT
Indexes
cart
cart_id BIGINT
game_id BIGINT
user_id BIGINT
Indexes
user
user_id BIGINT
address VARCHAR(255)
email VARCHAR(255)
password VARCHAR(255)
regdate DATETIME(6)
role VARCHAR(255)
tel VARCHAR(255)
username VARCHAR(255)
Indexes
board
board_id BIGINT
content VARCHAR(255)
hitcount BIGINT
regdate DATETIME(6)
title VARCHAR(255)
updatedate DATETIME(6)
writer VARCHAR(255)
user_id BIGINT
Indexes
my_game
my_game_id BIGINT
game_id BIGINT
user_id BIGINT
Indexes
board_attach
attach_id BIGINT
filetype VARCHAR(255)
originfile VARCHAR(255)
savefile VARCHAR(255)
savefolder VARCHAR(255)
1 more...
Indexes

Repository - (board, cart, game, user.....)

```java
package com.team4.game.repository;

import org.springframework.data.domain.Page;


public interface BoardRepository extends JpaRepository<Board, Long> {


    Page<Board> findByTitleContaining(String keyword, Pageable pageable);

    Page<Board> findByWriterContaining(String keyword, Pageable pageable);

    Page<Board> findByContentContaining(String keyword, Pageable pageable);

    Page<Board> findByTitleContainingOrWriterContainingOrContentContaining(String keyword, String keyword1, String keyword2 ,Pageable pageable);

    void deleteByUserId(Long id);

}
```

```java
package com.team4.game.repository;

import java.util.List;

public interface GameReplyRepository extends JpaRepository<GameReply, Long>{

    @Query("select r from GameReply r join fetch r.game where game_id=?1")
    public List<GameReply> gameReplyList(Long gameId);

    @Query("select AVG(grade) from GameReply r where gno = ?1")
    Double findOneAvg(Long gameId);

    public void deleteByUserId(Long id);
}
```

Repository 를 사용
-Entity에 의해 생성된 DB에 접근하는 메서드들을 사
 용하기 위한 인터페이스이다.
-@Query를 사용하여 sql 문을 작성

Service, ServiceImpl - (board, cart, game, user.....)

```java
package com.team4.game.service;

import java.util.List;


public interface BoardService {

    public void insert(Board board,User user);

    void insertBoardAndFile(Board board, User user, List<BoardAttach> attachList);

    public List<Board> boardList();

    public Board findById(Long id);

    public void update(Board board);
    public void delete(Long id);

    public Long count();

    public void insetReply(BoardReply reply);

    public List<BoardReply> replyList(Long bno);

    public void replyDelete(Long id);

    public Page<Board> findAll(Pageable pageable);

    public Page<Board> findList(String field, String keyword, Pageable pageable);


}

```

```java
package com.team4.game.service;

import java.util.List;


@Service
public class BoardServiceImpl implements BoardService{
    @Autowired
    private BoardRepository boardRepository;

    @Autowired
    private BoardReplyRepository replyRepository;

    @Autowired
    private BoardAttachRepository boardAttachRepository;

    @PersistenceContext
    EntityManager em;


    @Transactional
    @Override
    public void insert(Board board, User user) {
        // TODO Auto-generated method stub
        board.setUser(user);
        boardRepository.save(board);
    }

    //게임, 파일 동시에 등록
        @Transactional
        @Override
        public void insertBoardAndFile(Board board, User user, List<BoardAttach> attachList) {
            // TODO Auto-generated method stub
            board.setUser(user);
            Board board2=boardRepository.save(board);
            System.out.println(board2);
            if(attachList.size()>0) {
                for(BoardAttach boardAttach:attachList) {
                    boardAttach.setBoard(board2);
                    boardAttachRepository.save(boardAttach);
                }
            }
        }

    @Override
    public Page<Board> findList(String field, String keyword, Pageable pageable) {
        System.out.println("field....."+field +"keyword....."+keyword);
        // TODO Auto-generated method stub
        Page<Board> boardList=null;
        if(field.equals("") || field==null) {
            boardList= boardRepository.findAll(pageable);
        }
        else if(field.equals("title")) {
            boardList= boardRepository.findByTitleContaining(keyword, pageable);
        }else if(field.equals("writer")) {
            boardList= boardRepository.findByWriterContaining(keyword, pageable);
        }else if(field.equals("content")) {
            boardList= boardRepository.findByContentContaining(keyword, pageable);
        }else if(field.equals("cwt")) {
```

Service Interface 만들기 , Service Intertface 구현한 class 만들기

Controller - (board, cart, game, user.....)

```java
package com.team4.game.controller;

import java.io.IOException;

@Log
@Controller
@RequestMapping("/board/*")
public class BoardController {

    @Autowired
    private BoardService boardService;
    @Autowired
    private UserService userService;
    @Autowired
    private MultipartModule multipartModule;

    @GetMapping("register")
    public void insert() {

    }

    //@PostMapping("insert1")
    public String insert1(Board board) {
        User user=userService.findByUsername(board.getWriter());
        boardService.insert(board,user);
        return "redirect:/board/list";
    }
    @PostMapping("insert")
    public String insert(Board board, HttpServletRequest request,
            @RequestParam("files") List<MultipartFile> files, @AuthenticationPrincipal PrincipalDetails principal) throws IllegalStateException, IOException {
        List<BoardAttach> attachList = multipartModule.fileUploadBoard(request, files);
        boardService.insertBoardAndFile(board, principal.getUser(), attachList);
        return "redirect:/board/list";
    }

    //전체보기(페이징)
    @GetMapping("list")
    public String listPage(@RequestParam(name="field",defaultValue="") String field,
            @RequestParam(name="keyword",defaultValue="")String keyword,
            @PageableDefault(size = 10, sort="id", direction = Sort.Direction.DESC) Pageable pageable, Model model) {
        Page<Board> searchList = boardService.findList(field,keyword,pageable);
        long pageSize=searchList.getSize();
        long blockSize=3l;
        long rowNm=searchList.getTotalElements();
        long totPage=(long)Math.ceil((double)rowNm/pageSize);
        long currPage=searchList.getPageable().getPageNumber();
        long startPage=((currPage)/blockSize)*blockSize;
        long endPage=startPage+blockSize;
        if(endPage>totPage)
            endPage=totPage;
        if(totPage==0)
            endPage=1;
        boolean prev=startPage>0?true:false;
        boolean next=endPage<totPage?true:false;

        model.addAttribute("startPage", startPage);
        model.addAttribute("blockSize", blockSize);
        model.addAttribute("endPage", endPage-1);
        model.addAttribute("prev", prev);
        model.addAttribute("next", next);
```

만들어 둔 Service 를 Controller 에서 호출하기.

@Service 라는 어노테이션으로 선언한 Class 를 사용하기위해서 @Autowired 라는 어노테이션을 사용해서 해당 서비스를 스프링에 등록시켜줌.

Controller - (board, cart, game, user.....)

```css
@charset "UTF-8";
/*Basic design of webpage*/
*{
  margin:0;
  padding: 0;
  box-sizing: border-box;
  font-family: 'Poppins', sans-serif;
}

 html{
  background: url("../img/slider-bg-2.jpg");
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
  height: 969px;
 }

body{
  display: grid;
  place-items: center;
  text-align: center;
  background-size: cover;
}
/*Basic design of login form*/
.content{
  width: 330px;
  border-radius: 10px;
  padding: 40px 30px;
  margin-top: 100px;
  box-shadow: -3px -3px 9px #aaa9a9a2,
              3px 3px 7px rgba(147, 149, 151, 0.671);
}
/*Design the headings*/
.content .text{
  font-size: 25px;
  font-weight: 600;
  margin-bottom: 35px;
  color: rgb(247, 233, 233);
}
/*Design the input spaces*/
.content .field{
  height: 50px;
  width: 100%;
  display: flex;
  position: relative;
}

.field input{
  height: 100%;
  width: 100%;
  padding-left: 45px;
  font-size: 18px;
  outline: none;
```

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>



<%@ include file="../includes/headerb.jsp" %>

    <div class="container">

  <h2>${board.writer} 글보기</h2>

    <div class="form-group">
      <label for="id">글번호</label>
      <input type="text" class="form-control" id="id" name="id" value="${board.id}" readonly="readonly">
    </div>
    <div class="form-group">
      <label for="title">제목</label>
      <input type="text" class="form-control" id="title" name="title" value="${board.title}" readonly="readonly">
    </div>
    <div class="form-group">
      <label for="writer">작성자</label>
      <input type="text" class="form-control" id="writer"name="writer" value="${board.writer}" readonly="readonly">
    </div>
        <div class="form-group">
      <label for="file">file</label>
      <div>
        <ul>
            <c:forEach items="${board.boardAttachs}" var="fileInfo">
                <li style="list-style:nome">
                            <%-- 실제경로(RealPath)가 아니어도 찾아줌 --%>
                            <img src="/resource/upload/board/${fileInfo.savefolder}/${fileInfo.savefile}" height="200">
                    ${fileInfo.originfile}
                </li>
            </c:forEach>
        </ul>
      </div>
    </div>

    <div class="form-group">
      <label for="writer">조회수</label>
      <input type="text" class="form-control" id="writer"name="writer" value="${board.hitcount}" readonly="readonly">
    </div>
     <div class="form-group">
      <label for="content">내용:</label>
      <textarea class="form-control" rows="5" id="content" name="content" readonly="readonly">${board.content}</textarea>
    </div>

    <div class="form-group text-right">
    <c:if test="${principal.user.username==board.writer}">
      <button type="button" class="btn btn-secondary btn-sm" id="btnUpdate">수정하기</button>
      <button type="button" class="btn btn-secondary btn-sm" id="btnDelete">삭제하기</button>
    </c:if>
      <button type="button" class="btn btn-secondary btn-sm" id="btnList">목록보기</button>
    </div>

    <section class="mb-5" style="color: black;">
        <div class="card bg-light">
            <div class="card-body">
                <!-- Comment form-->
                <form class="mb-4">
```

게임플렛폼 사이트인 만큼 프론트 부분도 중요하게 생각하였습니다.

css를 사용하여 jsp에 <link>태그를 사용하여 사용, 메인화면에서 다른 페이지와 중복되는 부분은 header , footer 로 나누어서 <%@ ...@> 스크립트 태그를 사용하여 중복화면 호출.

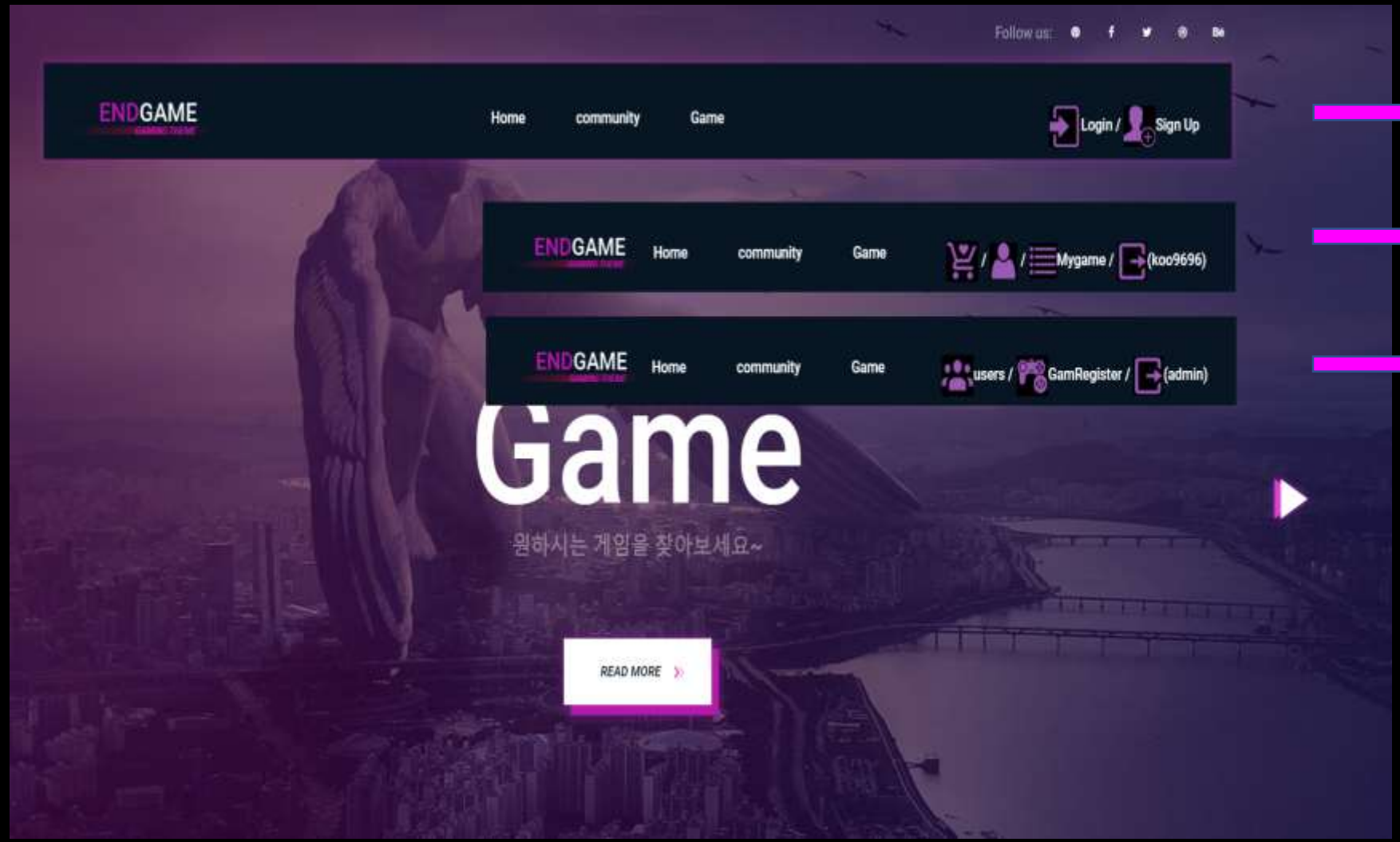
stage.3

화면 및 기능 구현





메인 화면 및 메인 화면 상단



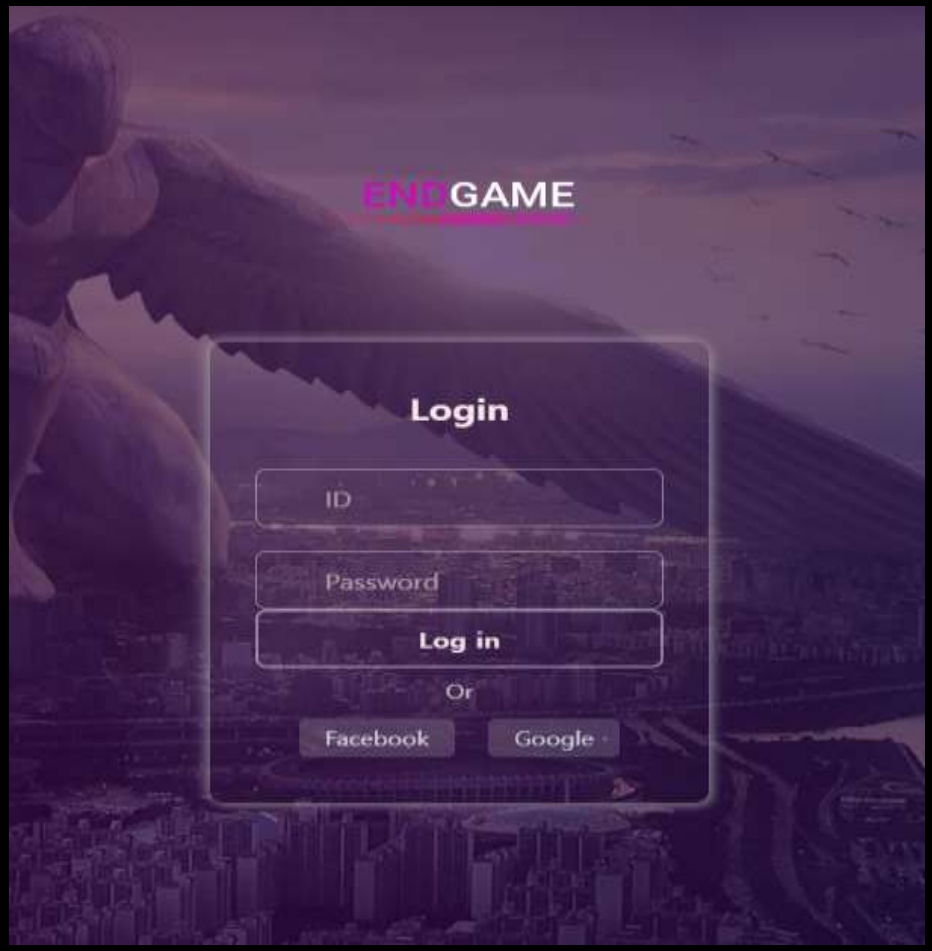
비회원

회원

관리자

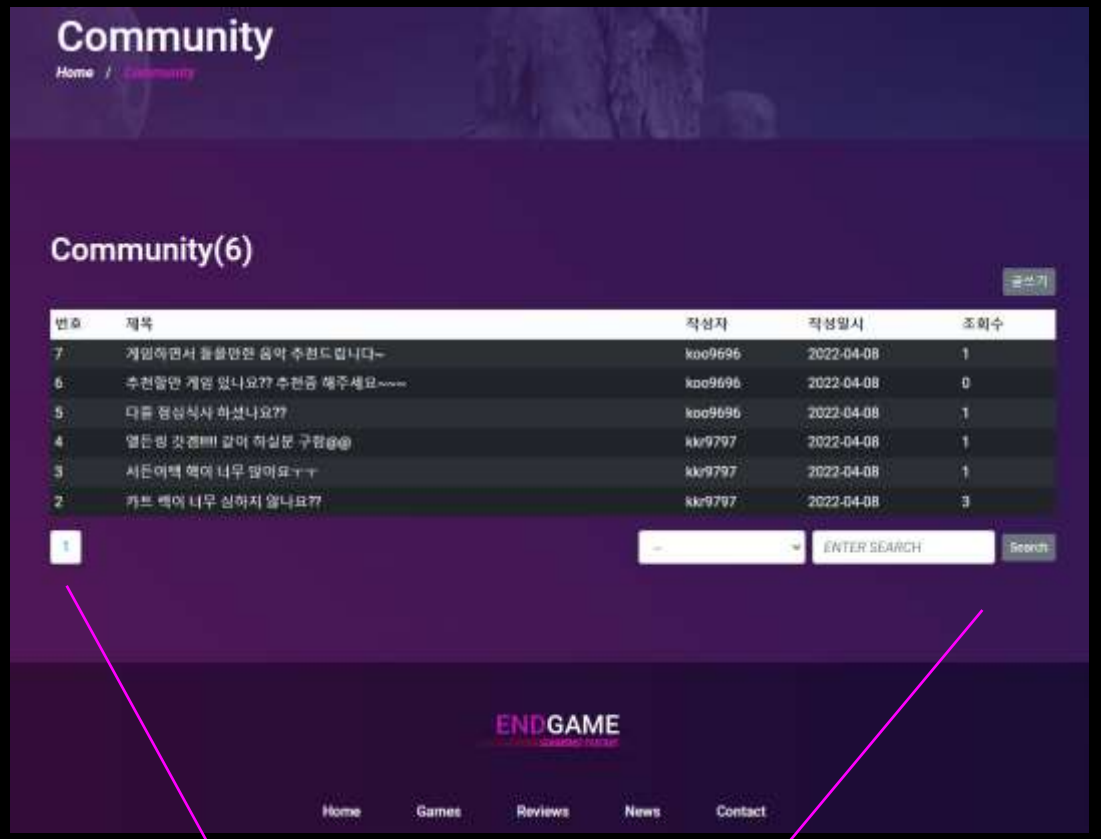
Spring Security 를 사용해 권한을 부여, 권한에 따라 접근할 수 있는 영역을 달리함.

로그인 및 회원가입 화면


ENDGAME
Login
ID
Password
Log in
Or
Facebook
Google

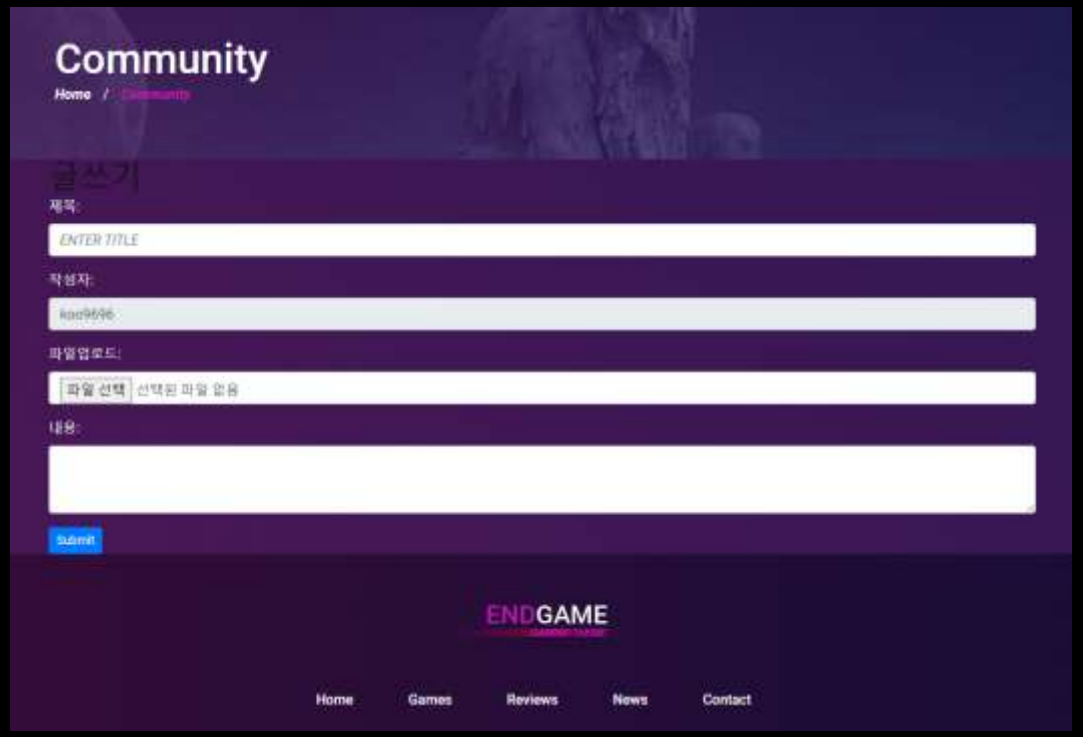

ENDGAME
회원가입
아이디:
중복확인
비밀번호
비밀번호 확인
email
휴대폰번호를 입력하세요
주소를 입력하세요
회원가입

자유게시판



검색과 페이징 기능

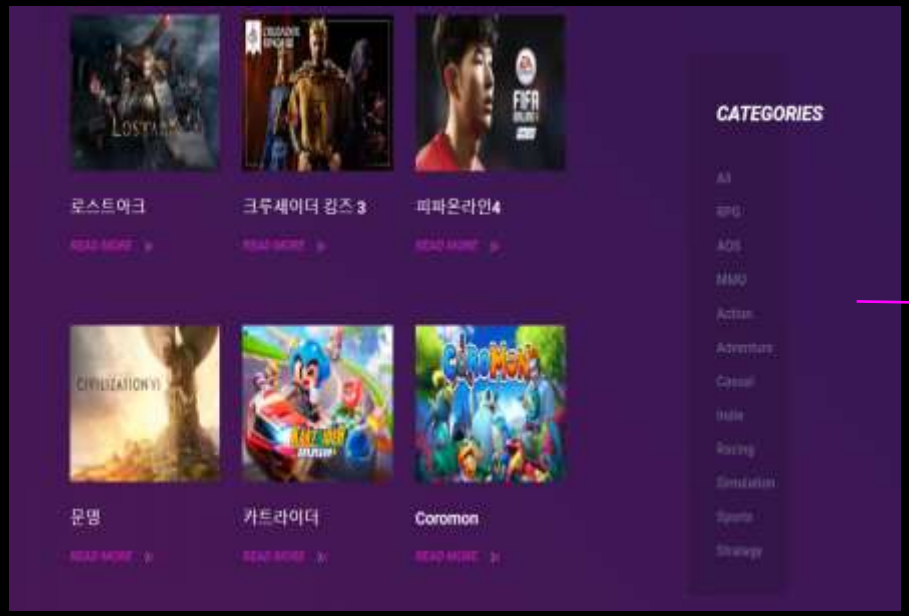
비회원 : 글보기
회원 : 글쓰기, 수정, 댓글
관리자:글삭제



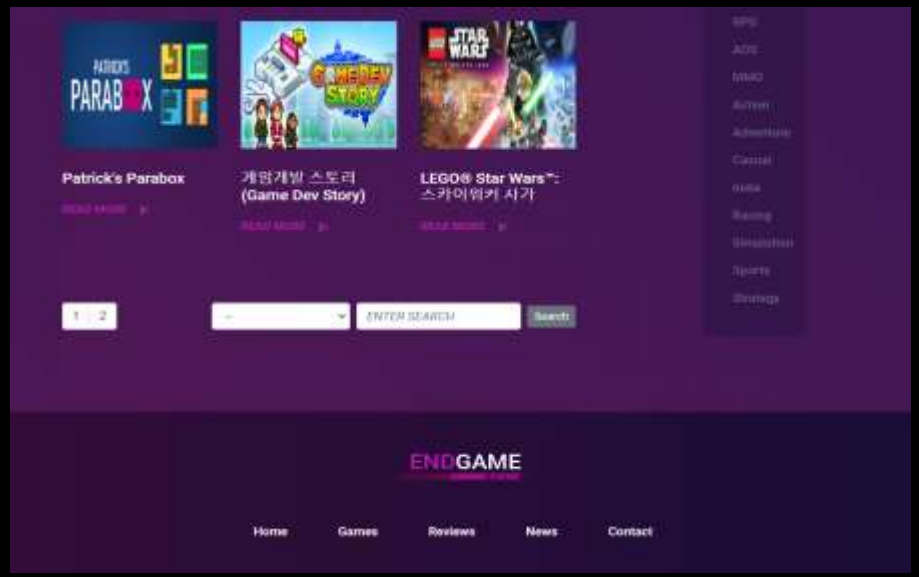
게임목록 1

카테고리&검색 결과로 나오는 게임들 5개의 이미지를 띄움

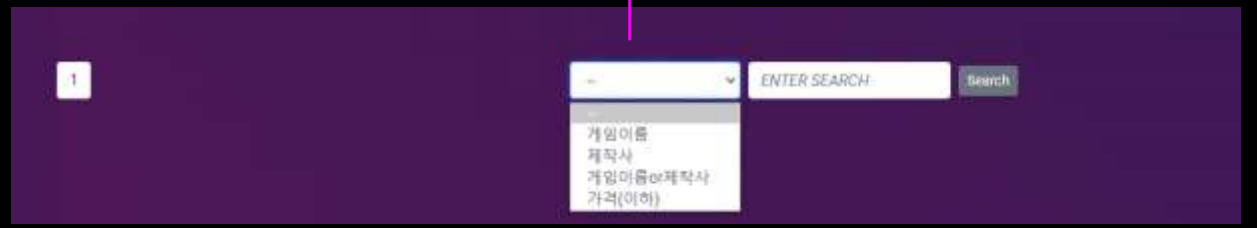
게임목록 2



카테고리를 선택 시
해당 카테고리의 게임들만 보여줌

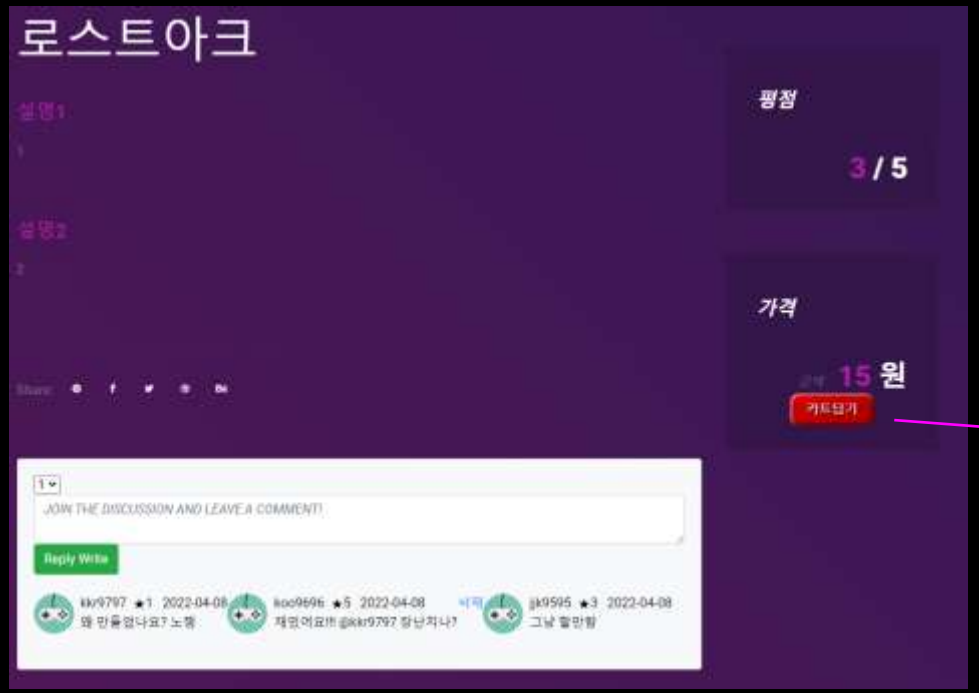


Community 와 마찬가지로 검색 과 페이징 기능이 가능하며 게임의 가격으로 검색했을 시 가격의 내림 차순으로 정렬



게임 상세보기

게임의 상세정보
(사진, 발매일, 개발자, 가격, 주설명, 보조설명, 판매량)

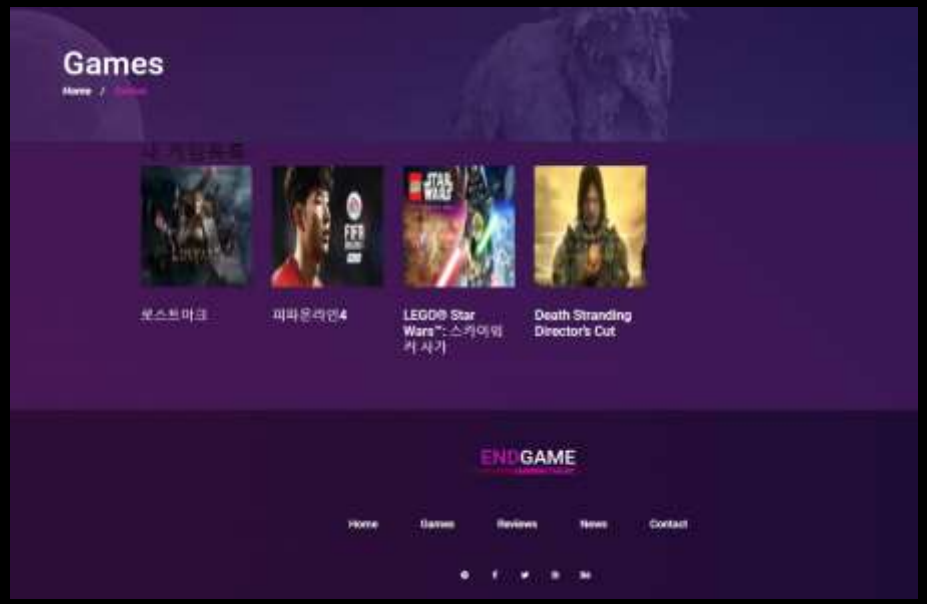


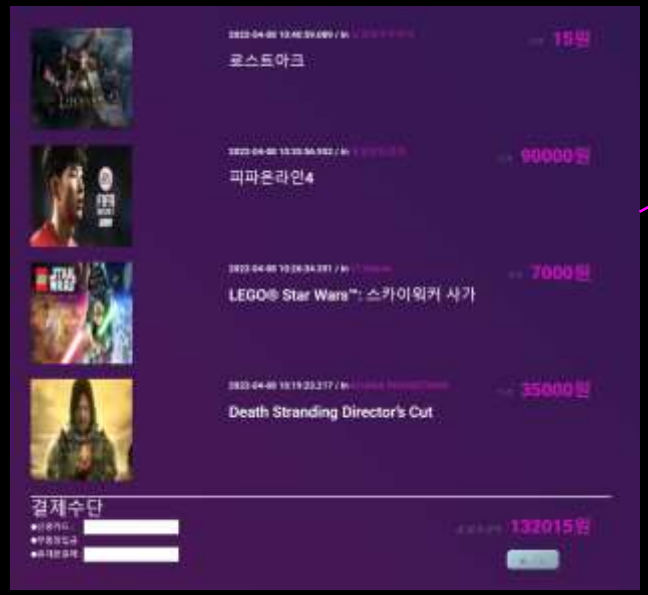
평점과 리뷰 작성 가능
평점의 평균을 바로 계산

카트에 바로 담기

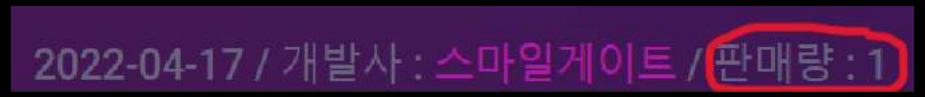
장바구니 기능 및 결제

장바구니에서 게임 개별 삭제
가격 총합을 계산
장바구니나 내 게임 목록에 있을 경우 카트에 추가 불가능
(중복 불가능)

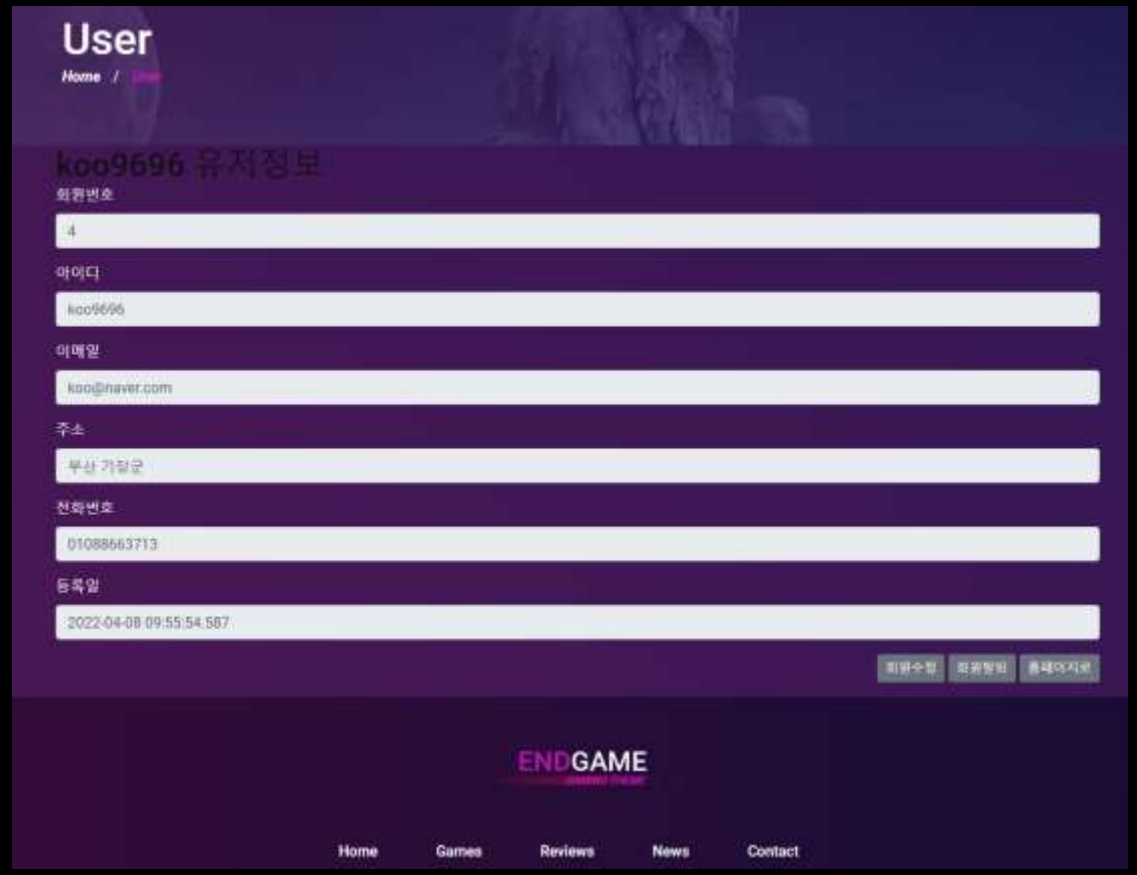




결제 시 내 게임목록에 추가
제품 판매량 증가



내 정보 수정 및 회원탈퇴



내 정보 수정 가능
DB와 Session을 동시에 수정해서 회원정보를 수정하여도 다른 곳에서 principle을 써도 수정한 정보가 반영되어 있음.

회원 탈퇴 시 유저의 모든 활동 내역 삭제
(댓글, 작성글, 카트, 내 게임목록, 평점)

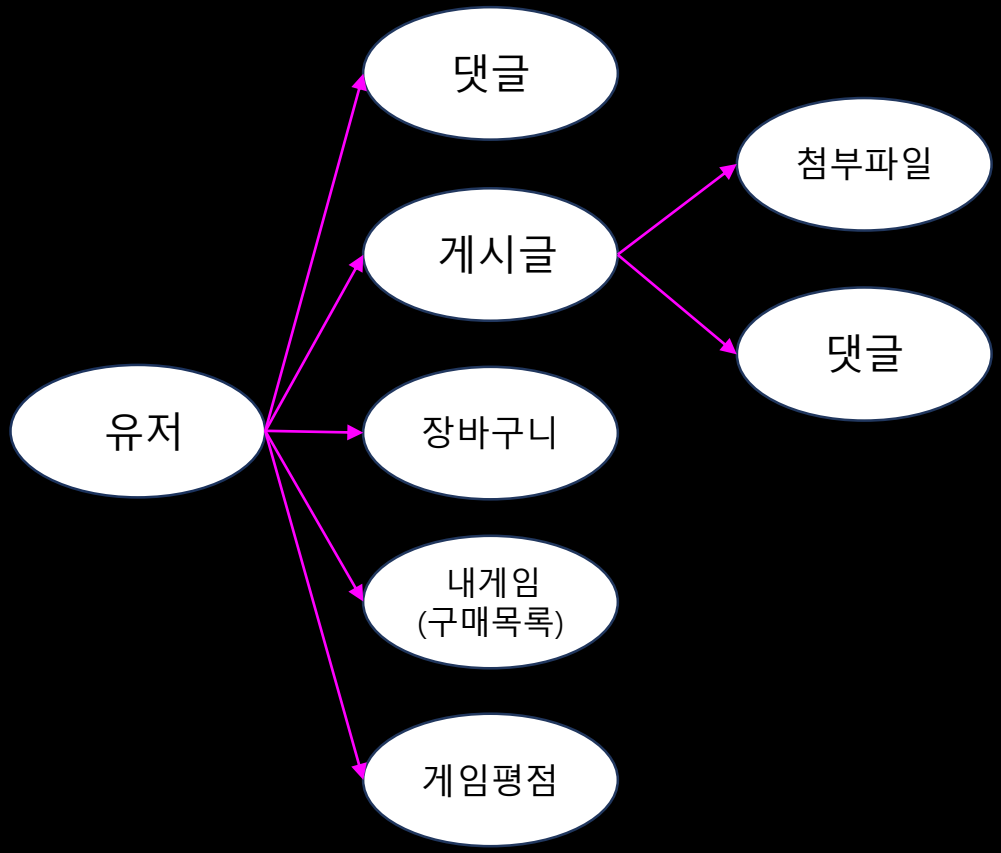
관리자 게임 관리 및 유저 관리

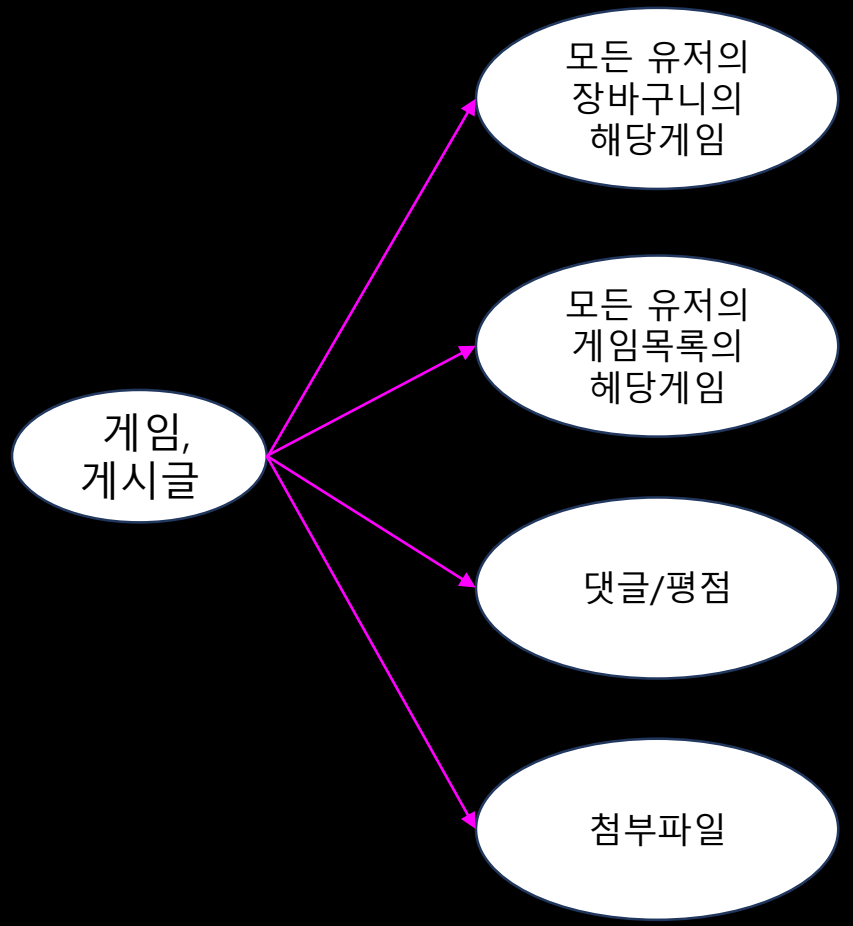
관리자: 게임등록, 수정 삭제
삭제 시 카트, 내 게임목록, 평점도 함께 삭제

회원삭제 : 탈퇴와 마찬가지로 유저가 활동한 모든 활동내역(글, 댓글, 카트, 구매, 평점)도 같이 삭제

삭제 절차

상위 요소를 삭제할 경우 하위요소 전부 삭제하도록 설정을 하여 오류제거
유저는 유저가 활동한 모든 내역 / 해당 게임, 게시글이 있는 모든 곳에서 삭제





stage.4

후기

후기

정재규 : 처음에 나름 의견교환이 되었다고 생각했으나 막상 분업했던 것들을 합칠 때 여러 문제가 생기는 걸보니 부족했다는 것을 알았습니다. 다음 팀프로젝트를 할 때는 서로 잘 이해했는지 확인을 하면서 진행하려고 합니다.

김경래 : Spring 과 JPA에 대한 개념이 충분치 않은 상태에서 개발을 진행하고 하다보니 어려움이 많이 있었습니다 . 개발만 중요한 것이 아니라 DB설계와 주소 등 기획과 구성의 중요성을 알았고 팀원들이 도와줌으로써 팀원 간의 소통, 협동의 중요성을 많이 느꼈습니다.

구주용 : 수업시간에 배웠던것을 활용하는 것도 중요하지만 무엇보다도 개인이 아닌 여러명에서 같이 진행되어야 하는 프로젝트이기 때문에 서로의 의사소통이 무엇보다도 중요하다고 생각 되었습니다. 저 또한 개발만 중요한 것이 아니라 DB설계와 주소 등 기획과 구성의 중요성을 알았고 팀원들이 도와줌으로써 팀원 간의 소통, 협동의 중요성을 많이 느꼈습니다.