
ZERO FIRM

**TEAM ZERO
PRESENTATION**

목차

1 프로젝트 개요

2 프로젝트 설계

3 UI/기능 소개

4 프로젝트 구조



Part 1

프로젝트 개요

1-1. 팀원 소개

1-2. 프로젝트 이름 및 주제

1-3. 프로젝트 주제 선정 동기

1-4. 프로젝트 기간/일정

Part 1.

프로젝트 개요

1-1. 팀원 소개



오시우

Back End

기획 / DB 구축 / 발표
공지사항 관리 시스템
메인, 컬렉션 콘텐츠
관리 및 출력
/ 디테일 화면 출력
/ 매핑 주소 기획



오성근

Back End

기획 / DB 구축 /
문의사항 시스템 구축
/ 이벤트 게임 페이지
/ 이미지 등록 시스템
메인 포스터 관리



우하영

Front End

기획 / 페이지 구성
화면 구성 및 UI,
레이아웃 디자인
자료 수집 및 시각화
PPT / 발표



이남호

Back End

기획 / DB 구축
로그인, 회원가입 등
회원관리 시스템
/ 마이페이지
/ 댓글 및 답글 기능

Part 1.

프로젝트 개요

1-2. 프로젝트 이름 및 주제

ZERO FIRM

주제

당신의 0번째 필름 (ZERO 필름)

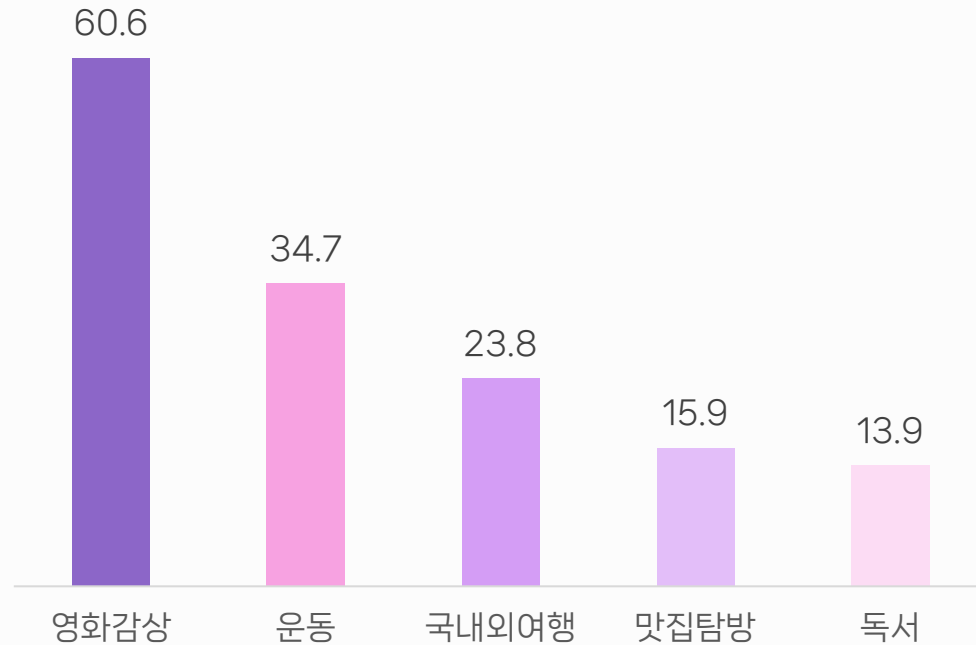
- ✓ 유저 중심 영화 평가 사이트
- ✓ 영화 소개 및 장르별 컬렉션
- ✓ 코멘트 기능을 활용한 유저 사이의 소통

프로젝트 개요

1-3. 프로젝트 주제 선정 동기

직장인들이 즐기는 취미생활

■ 영화감상 ■ 운동 ■ 국내외여행 ■ 맛집탐방 ■ 독서



[자료 / 잡코리아 / 복수응답]

직장인 10명 중 7명이
즐거하는 취미생활이 있다!

구체적으로 살펴봤을 때,
'영화/공연/스포츠 관람'의 경우 성별, 연령, 결혼
여부와 관계없이 **모든 사람들이 가장 많이 하는
취미 활동**으로 꼽기도 했다.

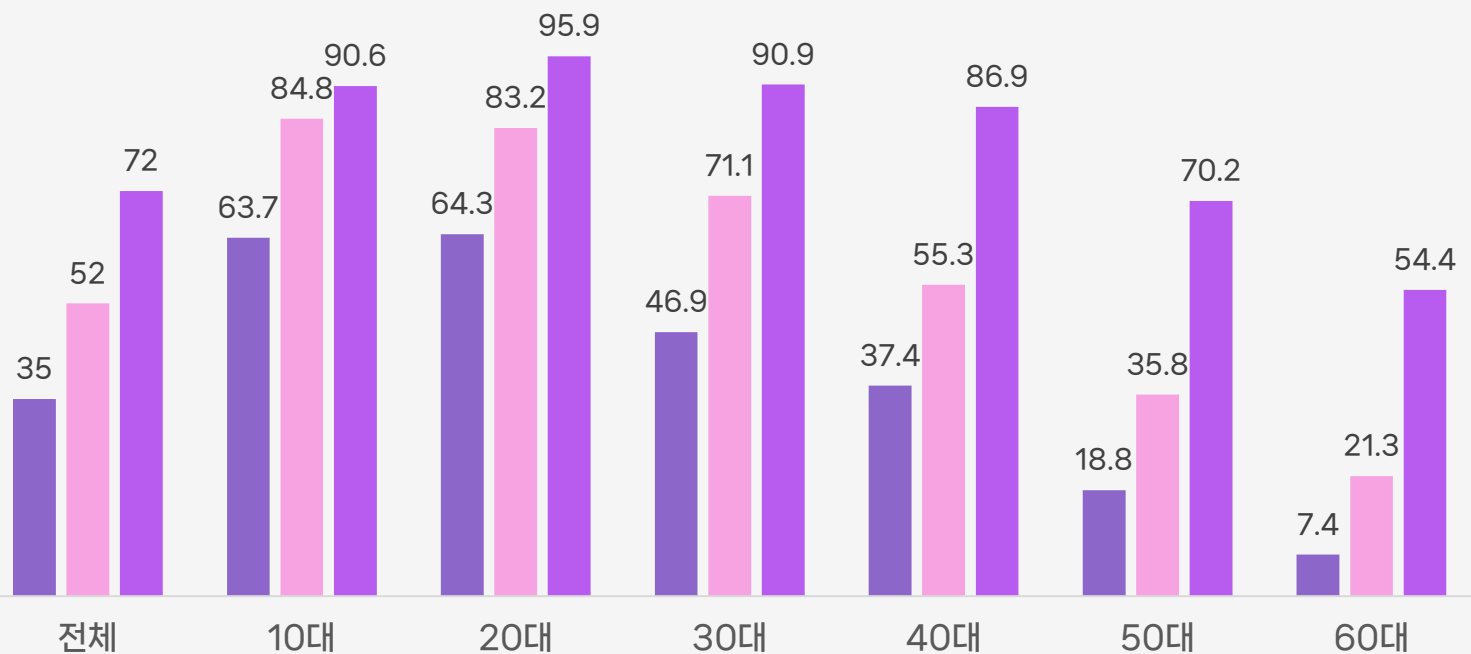
출처 : 데일리문화(<http://www.dailymunwha.com>)

프로젝트 개요

1-3. 프로젝트 주제 선정 동기

연령대별 OTT 이용 현황 변화

■ 2016년 ■ 2019년 ■ 2022년



[OTT 이용률 / 방송통신위원회]

정보통신정책연구원이
「세대별 OTT 서비스 이용 현황」
을 발표하였다.

2019년 이후 코로나19 발생과 함께 급격한
이용 증가 추세를 보임.

특히 Z세대의 경우 2명 중 1명은 하루에도 여
러 번 OTT 서비스를 이용하는 등 전형적인
헤비 유저의 특성을 보임.

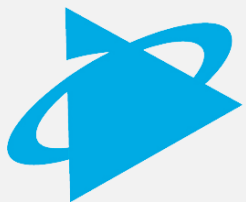
출처 : KDI 경제정보센터 (<https://eiec.kdi.re.kr>)

Part 1.

프로젝트 개요

1-3. 프로젝트 주제 선정 동기

WATCHA LAFTEL



coupang play

NETFLIX



MOLLE

누구나 평론가가 될 수 있다

많은 사람들의 아이디어가 모여 새로운 문화가 생겨나는데, 이러한 아이디어들을 생산적으로 가동시키기 위해서는 **토론과 정보의 교류** 또한 중요하다.

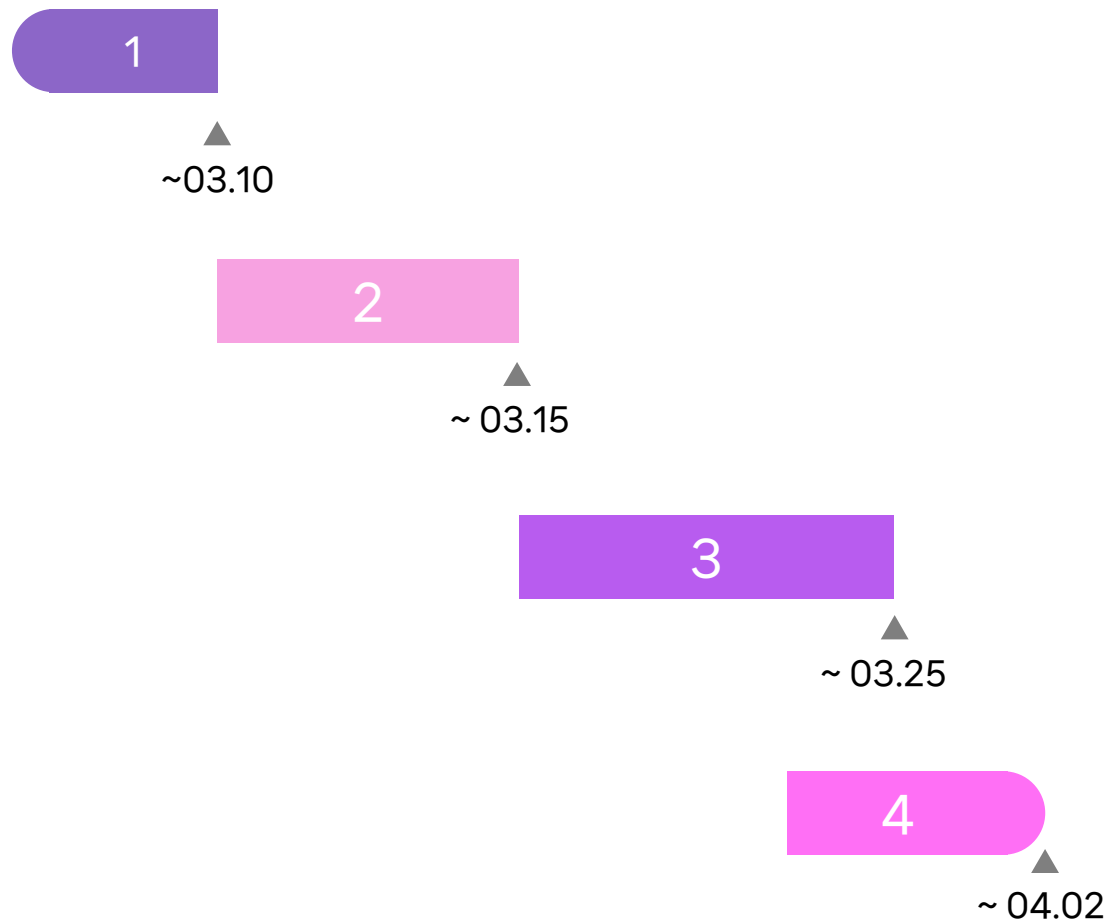
코로나19 발생과 함께 **OTT 서비스** 이용이 증가한 현재 활발한 커뮤니케이션의 매개체가 되기를 바라며,

우리의 목표는 소수 평론가의 의견 뿐만이 아닌 많은 유저가 만들어 가는 **소통의 장**이 되는 것이다.

프로젝트 개요

1-4. 프로젝트 일정/기간

전체 일정 : 2023.03.06~04.02



1.

기획, 회의 단계

- 기획/ 컨셉 구체화
- 역할 분담 및 자료수집

2.

DB 설계 단계

- 데이터 베이스 구상
- 테이블 설계

3.

초안 제작 단계

- 백, 프론트 동시 작업
- UI, 기능 구성 확정

4.

테스트, 수정 단계

- 연동, 매핑 완료 후 수정 작업

Part 2

프로젝트 설계

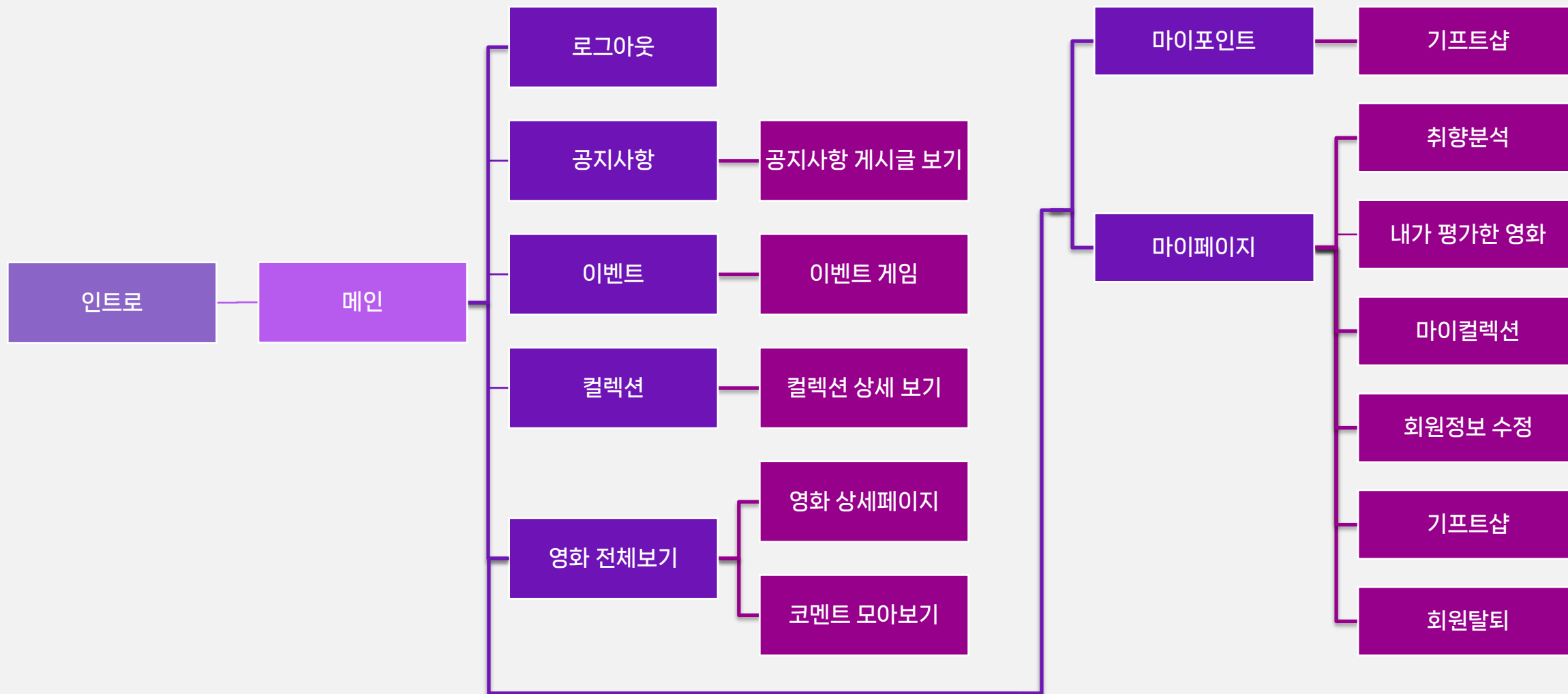
2-1. 사이트맵

2-2. 데이터베이스 테이블 구조

Part 2.

프로젝트 설계

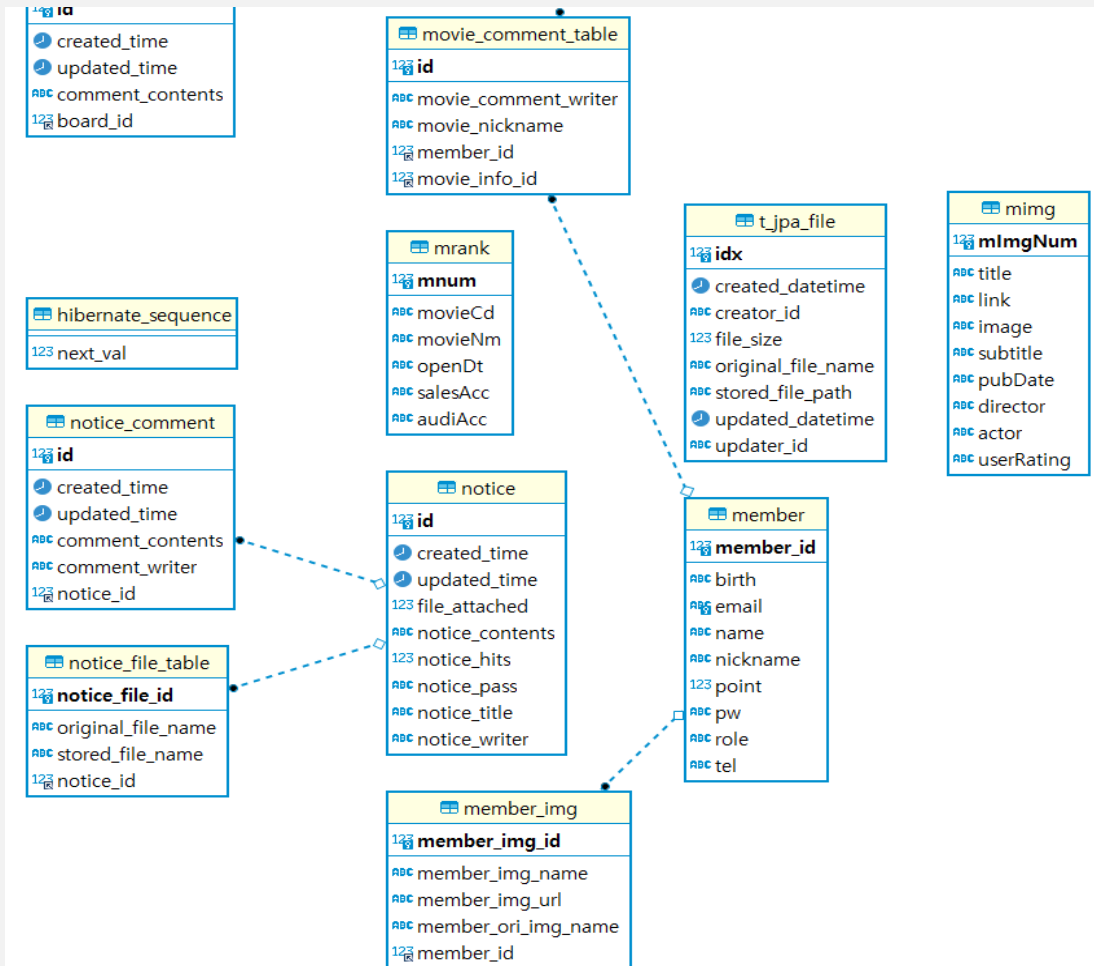
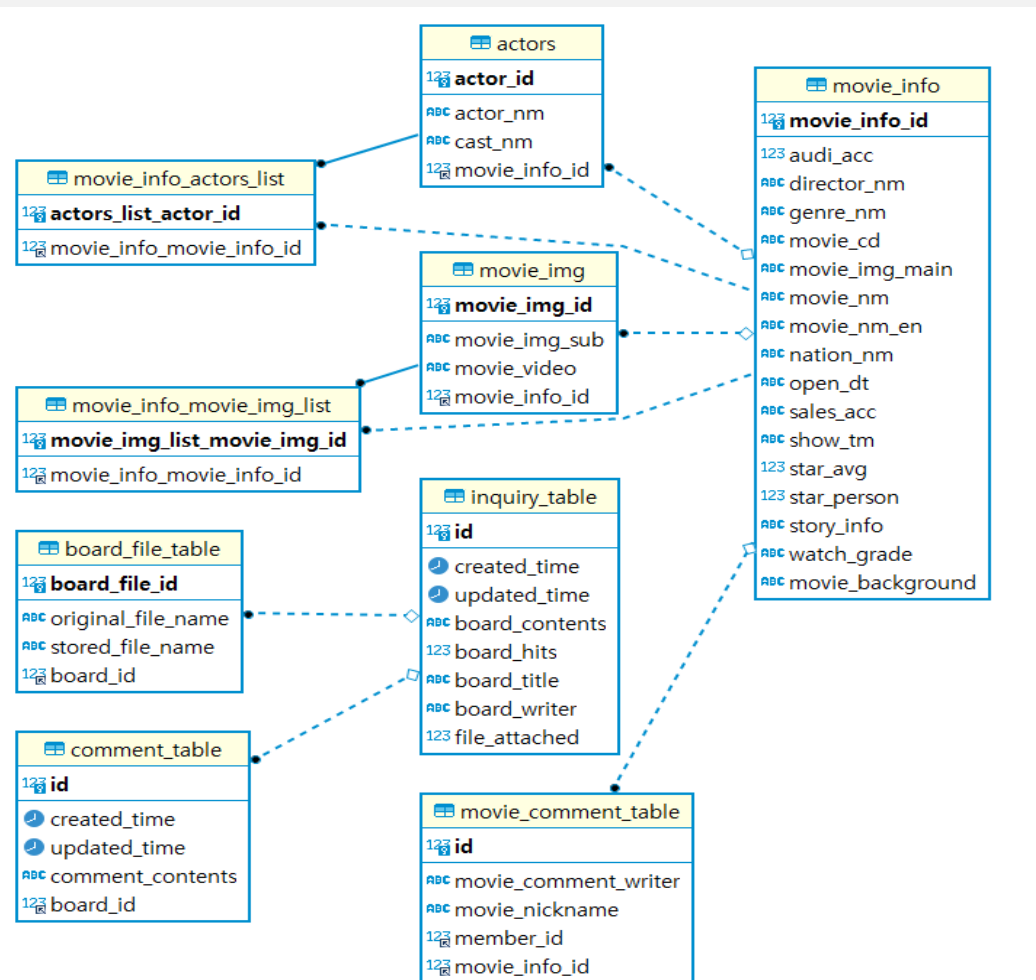
2-1. 사이트맵(로그인 했을 경우 기준)



Part 2. 프로젝트 설계

2-2. 데이터베이스 테이블 구조

- ✓ 테이블과 컬럼간의 연관성을 고려하여 @JoinColumn과 연관관계를 맺어 주었다.



Part 3

UI/기능 소개

3-1. 로그인/회원가입

3-2. 메인화면

3-3. 디테일 화면

3-4. 공지사항/문의사항(게시판)

3-5. 이벤트 게임

3-6. 마이페이지

3-7. 컬렉션 페이지

Part 3. UI/기능 소개

3- 1. 로그인/회원가입 - 로그인

localhost/member/logins

공지 컬렉션 로그인 회원가입

검색어를 입력하세요

로그인

아이디

비밀번호

로그인

회원가입

지금까지 ★22,832개의 리뷰와 함께하고 있어요.

서비스 이용약관 | 개인정보 처리방침

(34838)대전광역시 중구 중앙로 121번길 20(선화동 방산빌딩) 2층
대표자명 : 오시우, 오성근 | 개인정보보호책임자 : 우하영, 이남호
사업자 등록 번호 201-04-00707

COPYRIGHT © ZeroFilmT04, Inc. All rights reserved

로그인 페이지이다.

로그인 성공 시 메인 화면으로 이동한다.

아이디 또는 비밀번호가 다를 경우 alert창이 뜬다.

모든 계정은 role이 **admin, user**로 분리 되어있다.

Admin 계정으로 로그인 할 경우 바로 관리자 페이지로 이동한다.

localhost/member/logins

Part 3. UI/기능 소개

3- 1. 로그인/회원가입 - 회원가입

회원가입 페이지이다.

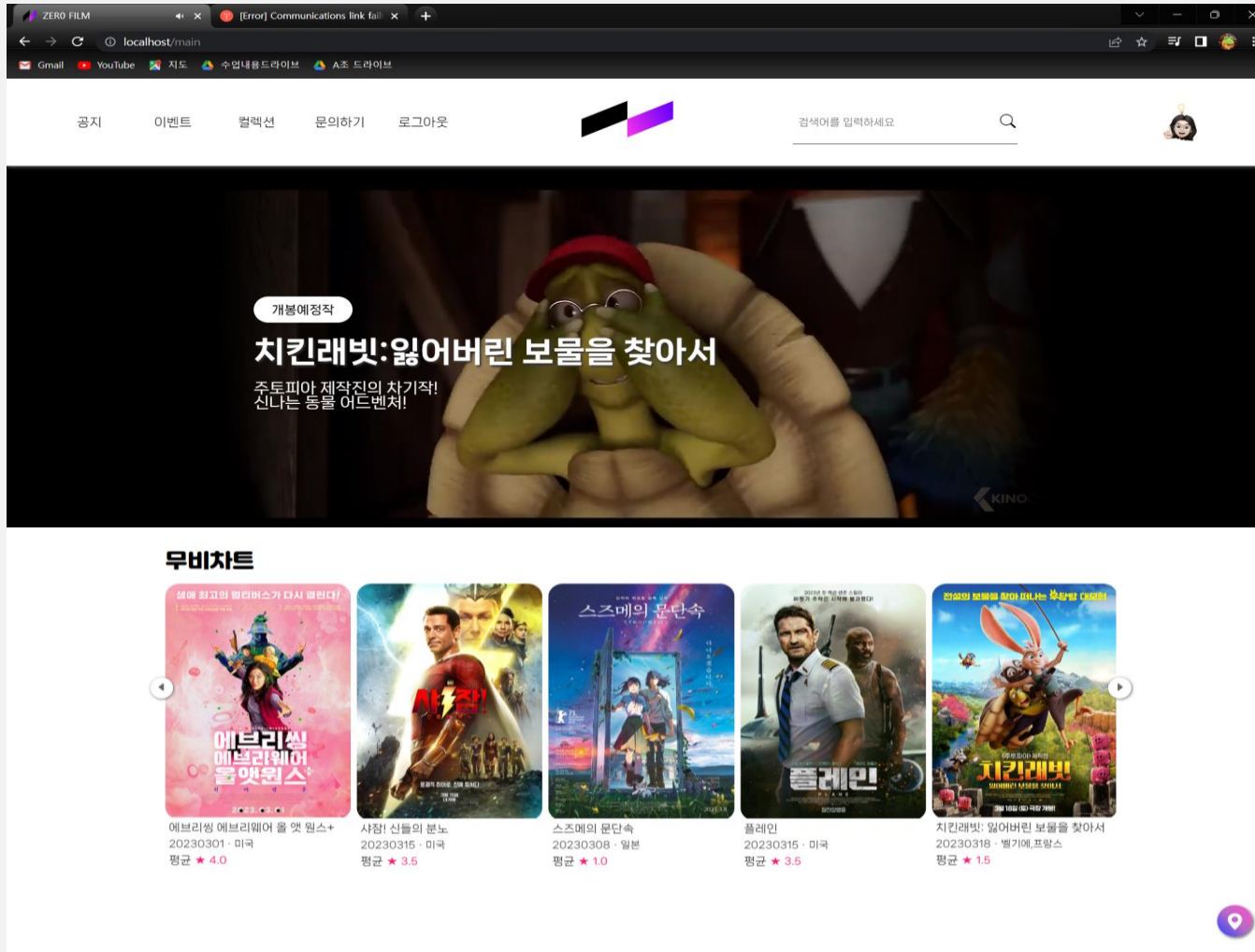
회원가입 성공 시 로그인 화면으로 이동한다.

아이디(이메일)중복, 비밀번호 길이 등의 제어가
포함 되어있다.

localhost/member/new

Part 3. UI/기능 소개

3- 2. 메인화면



로그인 한 후 **메인 페이지**이다.

메뉴 구성에서 차이가 있다.

로그인 전>공지, 컬렉션, 로그인, 회원가입

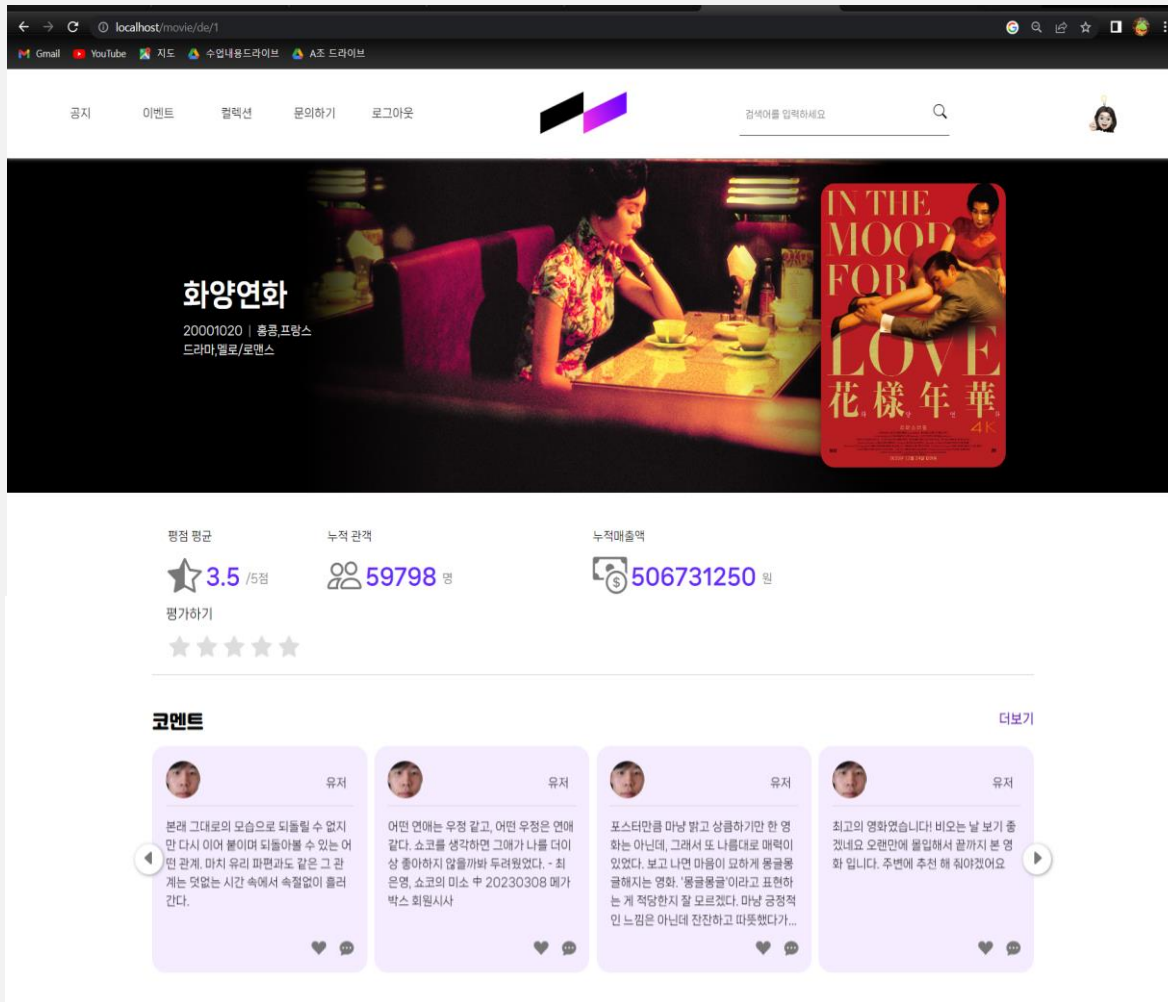
로그인 후>공지,이벤트,컬렉션,문의하기,로그아웃

무비차트는 **지난 달, 이번 달 기준 누적 관객수가 많은 순 10개**로 구성 되어있다.

localhost/main

Part 3. UI/기능 소개

3- 3. 디테일화면



영화 디테일 화면이다.

각자의 영화 아이디(movie_info_id)로 접속되도록 매핑이 되어있으며,

별점 평균, 누적관객, 누적매출액 같은 정보가 표시된다.

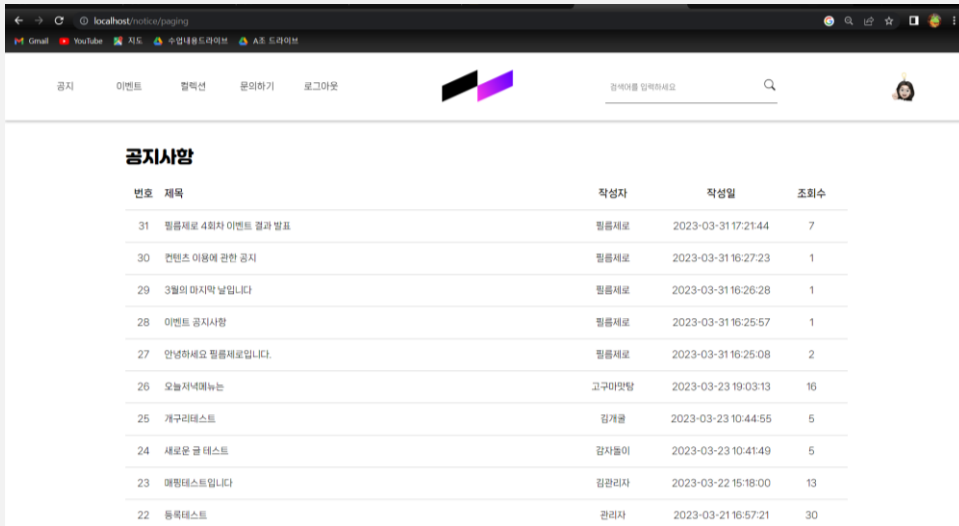
로그인 한 사용자에게 한해 **별점과 코멘트**를 남길 수 있다.

별점을 남기면 코멘트를 입력할 수 있는 모달 창이 나온다.

Localhost/movie/{id}

Part 3. UI/기능 소개

3- 4. 공지사항/문의사항(게시판) – 공지사항



The screenshot shows a web browser at localhost/notice/paging. The page has a navigation bar with links: 공지, 이벤트, 컬렉션, 문의하기, 로그아웃. Below the navigation bar is a search bar and a user profile icon. The main content area is titled '공지사항' and contains a table with 5 columns: 번호, 제목, 작성자, 작성일, and 조회수. The table lists 10 notices, with the first one being '필름제로 4회차 이벤트 결과 발표'.

| 번호 | 제목 | 작성자 | 작성일 | 조회수 |
|----|--------------------|-------|---------------------|-----|
| 31 | 필름제로 4회차 이벤트 결과 발표 | 필름제로 | 2023-03-31 17:21:44 | 7 |
| 30 | 컨텐츠 이용에 관한 공지 | 필름제로 | 2023-03-31 16:27:23 | 1 |
| 29 | 3월의 마지막 날입니다 | 필름제로 | 2023-03-31 16:26:28 | 1 |
| 28 | 이벤트 공지사항 | 필름제로 | 2023-03-31 16:25:57 | 1 |
| 27 | 안녕하세요 필름제로입니다. | 필름제로 | 2023-03-31 16:25:08 | 2 |
| 26 | 오늘저녁에보는 | 고구마맛탕 | 2023-03-23 19:03:13 | 16 |
| 25 | 개구리테스트 | 김개굴 | 2023-03-23 10:44:55 | 5 |
| 24 | 새로운 글 테스트 | 김자둘이 | 2023-03-23 10:41:49 | 5 |
| 23 | 매칭테스트입니다 | 김관리자 | 2023-03-22 15:18:00 | 13 |
| 22 | 등록테스트 | 관리자 | 2023-03-21 16:57:21 | 30 |

공지사항 리스트

공지사항 내용



The screenshot shows a web browser at localhost/notice/{id}. The page has a navigation bar with links: 공지, 이벤트, 컬렉션, 문의하기, 로그아웃. Below the navigation bar is a search bar and a user profile icon. The main content area is titled '공지사항' and contains a notice titled '필름제로 4회차 이벤트 결과 발표'. The notice text is: <p>영화제목을 찾아라 미니게임 당첨자 목록입니다.</p><p>"오시우"</p><p>"이남호"</p><p>"오성근"</p><p>"우하영"</p><p>축하드립니다.</p>

| 공지 | 이벤트 | 컬렉션 | 문의하기 | 로그아웃 |
|--|-----|-----|------|------|
| <p>공지사항</p> <p>필름제로 4회차 이벤트 결과 발표</p> <p>8 필름제로 2023-03-31T17:21:44</p> <p><p>영화제목을 찾아라 미니게임 당첨자 목록입니다.</p><p>"오시우"</p><p>"이남호"</p><p>"오성근"</p><p>"우하영"</p><p>축하드립니다.</p></p> <p>도움말</p> | | | | |

공지사항 페이지이다.

공지사항 리스트는 15개씩 페이지징 되어 있다.

공지번호, 제목, 작성자, 작성일, 조회수를 미리 볼 수 있다.

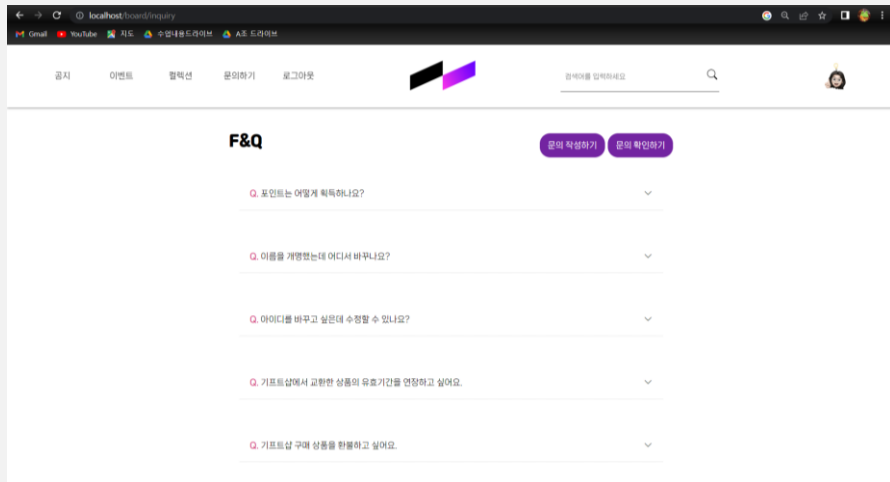
공지사항은 관리자만 작성할 수 있으므로 유저가 보는 페이지는 읽기만 가능하다.

Localhost/notice/paging?page=n

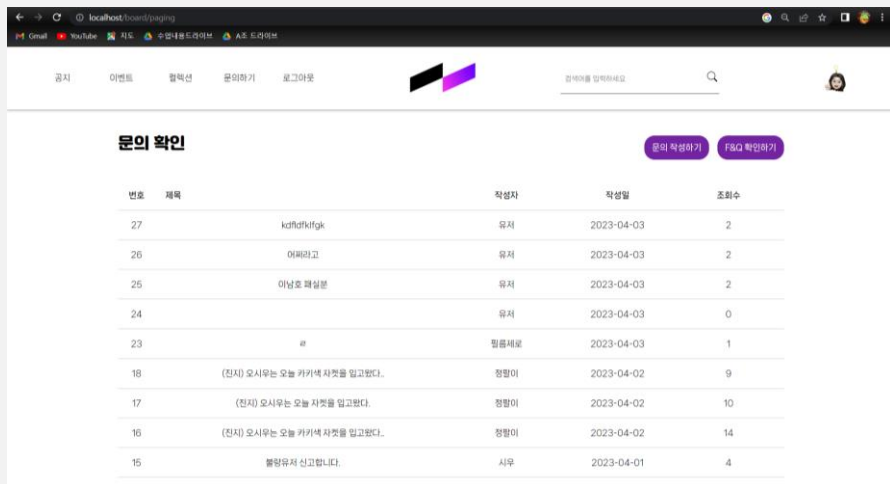
Localhost/notice/{id}?page=n

Part 3. UI/기능 소개

3- 4. 공지사항/문의사항(게시판) – 문의사항



문의사항 F&Q



문의사항 목록

문의사항 페이지이다.

F&Q에는 유저들이 자주 문의하는 내용을 볼 수 있고, 문의 확인 버튼을 통해 문의사항 리스트로 이동한다.

문의사항 목록은 10개씩 페이징 되어 있다.

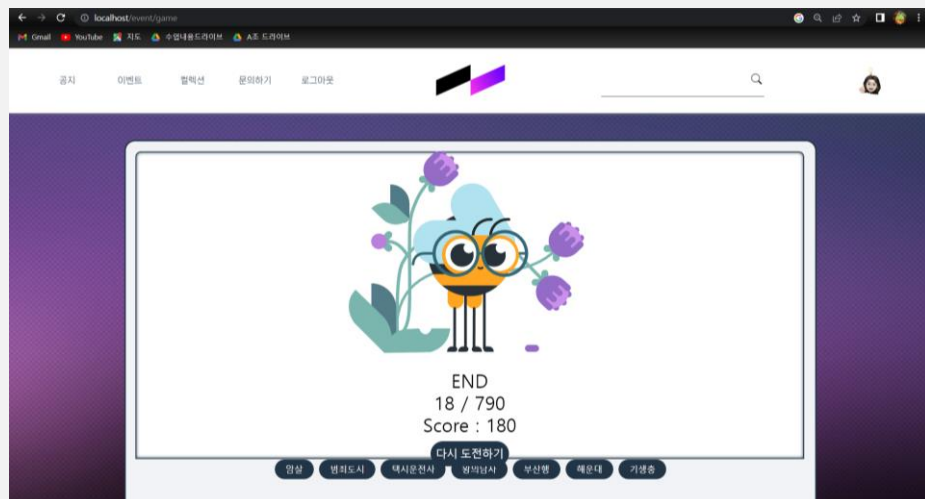
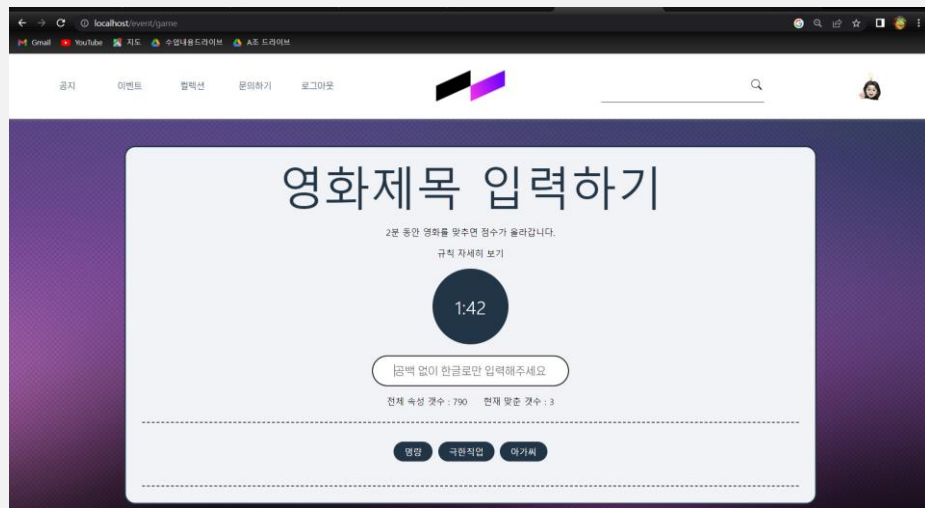
문의사항은 유저가 작성,수정,삭제할 수 있다.

Localhost/board/inquiry

Localhost/board/paging/n

Part 3. UI/기능 소개

3-5. 이벤트 게임



이벤트 게임 화면이다.

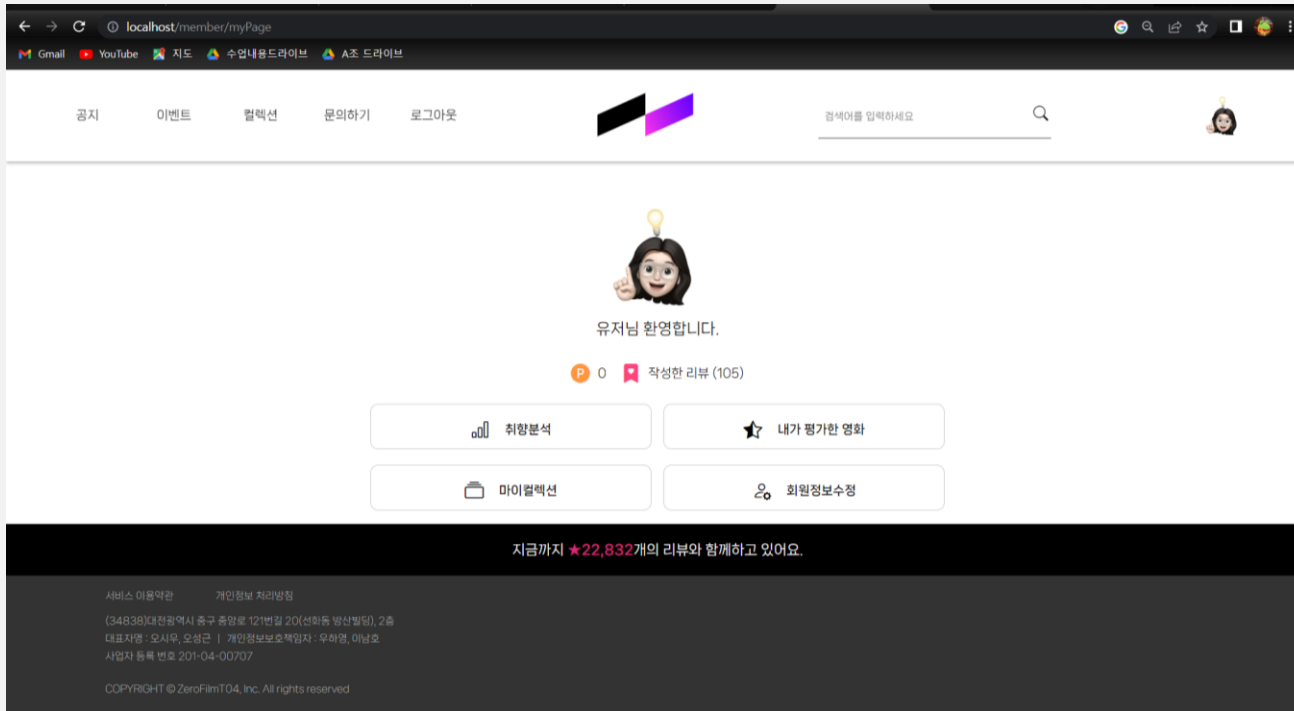
이벤트 배너클릭 또는 헤더의 이벤트 메뉴를 통해 들어갈 수 있다.

제한시간 2분 안에 영화 제목을 입력하는 게임이며, 2분이 지나면 맞춘 영화의 개수와 점수가 표시된다.

Localhost/event/game

Part 3. UI/기능 소개

3-6. 마이 페이지



마이페이지 화면이다.

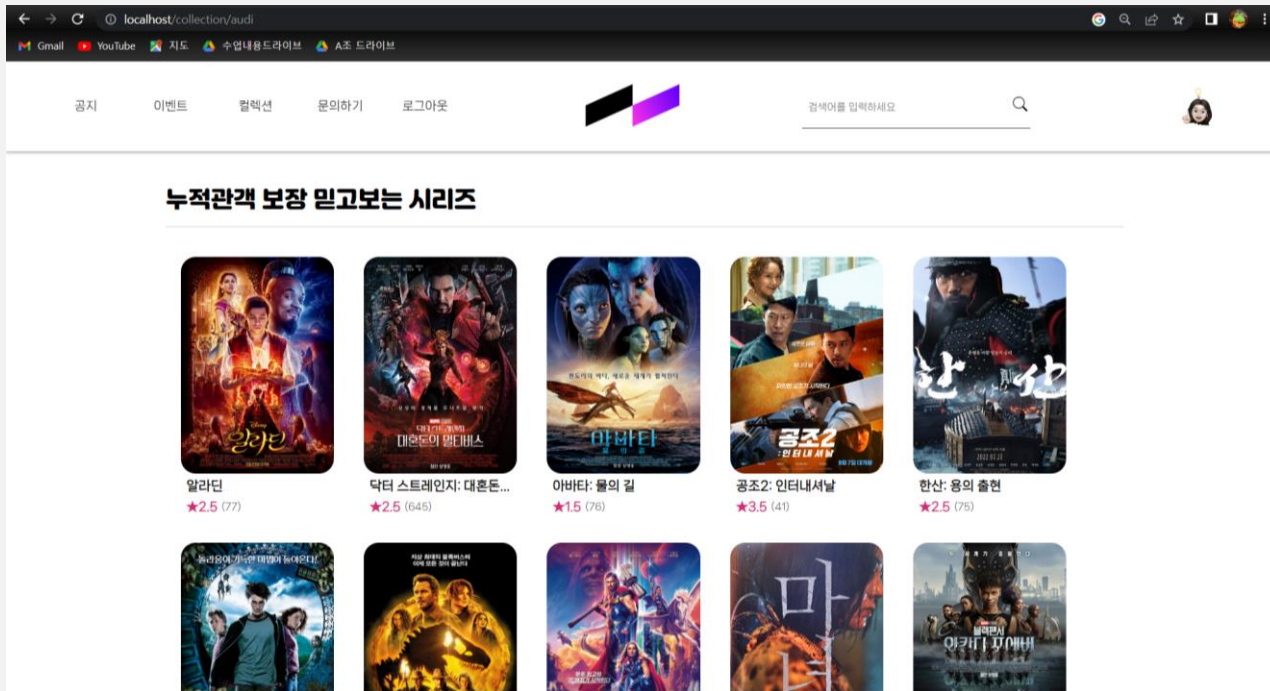
취향 분석, 내가 평가한 영화, 마이 컬렉션, 회원 정보 수정 메뉴로 구성되어 있다.

회원 정보 수정 페이지에서 회원 정보를 수정할 수 있으며, 프로필 사진을 등록할 수 있다.

Localhost/member/myPage

Part 3. UI/기능 소개

3-7. 컬렉션 페이지



컬렉션 화면이다.

컬렉션을 클릭하면 나오는 컬렉션 상세 페이지이다.

장르, 별점, 누적관객 등을 기준으로 만들어졌다.

영화 포스터 또는 제목을 클릭하면 해당 영화의 상세 페이지로 이동한다.

Localhost/collection/컬렉션이름

Part 4

프로젝트 구조

4-1. Project Structure

4-2. Spring Boot(API Server)

4-3. Spring Security

4-4. Thymeleaf

4-5. Lombok

4-6. JPA & Query Dsl (ORM)

4-7. JSON

프로젝트 구조

4-1. Project Structure

Spring Boot(API Server) 구조로 개발했으며, MVC패턴을 사용

| 사용한 기술 스택 |

- ✓ Spring Boot (API Server)
- ✓ Spring Security (Security)
- ✓ Thymeleaf
- ✓ Query Dsl
- ✓ Lombok
- ✓ JSON

- ✓ ModelMapper
- ✓ JPA
- ✓ MySQL
- ✓ Validation
- ✓ Git Book :

프로젝트 구조

4-2. Spring Boot (API Server)

| | |
|------------|--|
| Config | Project Configuration을 관리한다. |
| Security | Security, Oauth, jwt 관련 기능들을 관리한다. |
| Controller | API를 관리한다. |
| Dto | Request, Response Dto를 관리한다. |
| Repository | Domain + JPA / Query Dsl 을 관리한다. |
| Service | domain에 정의한 business logic 호출 순서를 관리한다. |
| Constant | 회원 계정의 role을 admin/user (enum)상수로 두고 관리한다. |
| Entity | 필요한 데이터를 저장할 테이블을 생성하고 관리한다. |

프로젝트 구조

4-3. Spring Security

```
@Override
protected void configure(HttpSecurity http) throws Exception {

    http.formLogin() FormLoginConfigurer<HttpSecurity>

        .loginPage("/member/login")           // 사용자 정의 로그인 페이지
        .defaultSuccessUrl("/main")           // 로그인 성공시
        .usernameParameter("email")          // 아이디 파라미터명 설정
        // .successHandler(new MyLoginSuccessHandler())
        .failureUrl(authenticationFailureUrl: "/member/login/error") // 로그인 실패 후 이동 페이지
        .and().logout() LogoutConfigurer<HttpSecurity>
        .logoutRequestMatcher(new AntPathRequestMatcher(pattern: "/member/logout"))
        .logoutSuccessUrl("/main"); // 로그아웃 성공 후 핸들러
}
```

```
@Override
public void configure(WebSecurity web) throws Exception{
    web.ignoring().antMatchers(...antPatterns: "/css/**", "/js/**", "/image/**", "/image/main/collection/**");
    // static 디렉토리 하위 파일에 대해 인증 없이 접근 가능하게 설정
}

1 usage
@Bean
public PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }

@Override //비밀번호 인코딩
protected void configure(AuthenticationManagerBuilder auth) throws Exception{
    auth.userDetailsService(memberService).passwordEncoder(passwordEncoder());
}
```

```
http.authorizeRequests()
    .mvcMatchers(...patterns: "/", "/member/**", "/item/**", "/image/**", "/images/**", "/login/**", "/main/**", //인가 api
        "/movie/**", "/event/**", "/notice/**") //주소에 이런식으로 표현되는 경로에 대해서는 모든 유저에 대하여 권한을 주겠다.
    .permitAll().mvcMatchers(...patterns: "/admin/**").hasRole("ADMIN") //주소에 이런식으로 표현되는 경로에 대해서는 관리자에 대하여만 권한을 주겠다.
    .anyRequest().authenticated();

http.exceptionHandling().authenticationEntryPoint(new CustomAuthenticationEntryPoint()); // 인증되지 않은 사용자가 접근할 경우 수행되는 작업 인증ap
}
```

프로젝트 구조

4-4. Thymeleaf

| th:each |

```
List<MovieInfo> movieList = movieService.movieRank();

System.out.println(movieList.get(0).getMovieNm());

model.addAttribute( attributeName: "movieList", movieList);
```

Controller부분에서 model 객체에 movieList를 담아줌.

```
<div th:each="movie:${movieList}" class="swiper-slide chartSlider">

  <div class="imgBox">
    <a th:href="x"> </a>
  </div>
  <div class="txtBox">
    <a th:href="@{/movie/de/${movie.id}}"> <h2 class="title" th:text="${movie.movieNm}"></h2></a>
    <div class="releaseInfo">
      <p class="date" th:text="${movie.openDt}"></p>
      <p class="country" th:text="${movie.nationNm}"></p>
    </div>

    <div class="rateData">
      <p>평균</p>
      <p class="rating" th:text="${movie.star_avg}"></p>
    </div>
  </div>
```

Html에서 해당 데이터의 개수 만큼 반복문을 돌려 데이터를 출력함.

| th:block / layout:fragment |

```
<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org" xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout" layout:decorate="~{body}">
  <th:block layout:fragment="css">
    <link rel="stylesheet" th:href="@{/css/common.css}" />
    <link rel="stylesheet" th:href="@{/css/main.css}" />
  </th:block>
  <th:block layout:fragment="script">
    <script th:src="@{/js/main.js}"></script>
    <script th:inline="javascript">
      /**/
      $(document).ready(function(){
        var msg = [[${errorMessage}]];
        if(msg !=null){
          alert(msg);
        }
      });
      /*]]*/
    &lt;/script&gt;
  &lt;/th:block&gt;
&lt;/html&gt;
&lt;body&gt;
  &lt;div layout:fragment="content"&gt;</pre>
</div>
<div data-bbox="449 653 937 842" data-label="Text">
<pre>&lt;th:block layout:fragment="script"&gt;&lt;/th:block&gt;
&lt;th:block layout:fragment="css"&gt;&lt;/th:block&gt; &lt;!-- th:block는 특정 html에 사용하겠다. --&gt;
&lt;/head&gt;
&lt;body&gt;
  &lt;div th:replace="header::header"&gt;&lt;/div&gt;
  &lt;div layout:fragment="content" class="content"&gt;&lt;/div&gt;
  &lt;div th:replace="footer::footer"&gt;&lt;/div&gt;
&lt;/body&gt;</pre>
<p>Body.html</p>
</div>
<div data-bbox="453 850 960 955" data-label="Text">
<p>Body html에 기본적으로 중복되는 css, script와 header, footer를 th:replace 를 통해 body에 포함시킨 후 th:fragment를 통해 설정한 이름을 찾아 해당 코드로 치환한다. 페이지 마다 따로 적용되어야 하는 css나 script는 th:block으로 layout:fragment 속성에 이름 지정 후 따로 적용 시킨다.</p>
</div>
<div data-bbox="877 967 997 997" data-label="Page-Footer">
<p>©Saebyeol Yu. Saebyeol's<br/>PowerPoint</p>
</div>
```

프로젝트 구조

4-5. Lombok

```
1 package com.zeromovie.filmzero.dto;
2
3 import com.zeromovie.filmzero.entity.actors;
4 import com.zeromovie.filmzero.entity.MovieInfo;
5 import lombok.*;
6
7 @Getter
8 @Setter
9 @ToString
10 @NoArgsConstructor
11 @AllArgsConstructor
12 public class MovieDto {
13
```

```
package com.zeromovie.filmzero.entity;
import com.zeromovie.filmzero.dto.MovieDto;
import com.zeromovie.filmzero.dto.NoticeFormDto;
import lombok.Getter;
import lombok.Setter;
import lombok.ToString;

import javax.persistence.*;
import javax.validation.constraints.NotNull;
import java.util.List;

@Entity
@Getter @Setter
@ToString
@Table(name = "movieInfo")
public class MovieInfo {
```

코드와 클래스의 단순화를 위해 Lombok 라이브러리를 사용하였다.

대부분의 entity와 dto에 lombok이 존재하며,
멤버변수의 값을 호출하기 위해 @Getter를, 멤버변수의 값을 변경하기 위해 @Setter를 사용하였다.

클래스의 변수들을 기반으로 ToString 메소드를 자동으로 완성시키기 위해 @ToString을 사용하였다.

프로젝트 구조

4-6. JPA & Query Dsl (ORM)

- ✓ JPA : 반복적인 CRUD 작업을 대체해 간단히 DB에서 데이터를 조회한다.
- ✓ Query Dsl : Join & Projections 등 JPA로 해결할 수 없는 SQL은 Query Dsl로 작성했다.

구조 설명 ① - movieRank

```

1 usage
62  @Override
63  public List<MovieInfo> getMovieRank() {
64      QueryResults<MovieInfo> results = query
65          .selectFrom((QMovieInfo.movieInfo))
66          .where(searchByLastMonth(),
67                searchByThisMonth())
68          .orderBy(QMovieInfo.movieInfo.audiAcc.desc())
69          .limit(10).fetchResults();
70      List<MovieInfo> movieInfoList = results.getResults();
71
72      System.out.println("movieRankRepositoryImpl");
73      System.out.println(movieInfoList.get(0).getMovieNm());
74      return movieInfoList;
75  }
76
77
78  }

```

movieInfo테이블의 데이터를 지난 달을 구하는 메소드 searchByLastMonth, 이번 달을 구하는 메소드 searchByThisMonth 를 사용하여 **누적관객 내림차순**으로 10개의 콘텐츠를 조회해 **movieInfoList** 라는 리스트에 저장한다.

구조 설명 ② - movieCollection - 장르별

```

13  public class MovieCollectionRepositoryImpl implements MovieCollectionRepository {
14      private JPAQueryFactory query;
15      public MovieCollectionRepositoryImpl(EntityManager em) {this.query = new JPAQueryFactory(em);}
16
17      1 usage
18      @Override
19      public List<MovieInfo> getRomanceCollection() { //멜로
20          QueryResults<MovieInfo> results = query
21              .selectFrom((movieInfo))
22              .where(movieInfo.genreNm.like( str: "%멜로%"))
23              .orderBy(QMovieInfo.movieInfo.audiAcc.desc())
24              .limit(16)
25              .fetchResults();
26          List<MovieInfo> movieInfoList = results.getResults();
27          return movieInfoList;
28      }

```

movieInfo 테이블에서 **like**를 사용하여

해당 컬렉션의 장르명이 포함된 콘텐츠를

누적 관객 내림차순으로 최대 16개를 조회하여 **movieInfoList**에 저장함.

조회 장르 - 액션, 애니, 범죄, 다큐, 호러, 로맨스, 스릴러

Part 4.

프로젝트 구조

4-6. JPA & Query Dsl (ORM)

구조 설명 ② - movieCollection - 별점 평균

```
1 usage
@Override
public List<MovieInfo> getStarCollection() { // 별점 높은 순
    QueryResults<MovieInfo> results = query
        .selectFrom((movieInfo))
        .orderBy(QMovieInfo.movieInfo.star_avg.desc())
        .limit(16)
        .fetchResults();
    List<MovieInfo> movieInfoList = results.getResults();
    return movieInfoList;
}
```

movieInfo 테이블의 **star_avg(별점평균)** 컬럼을

내림차순으로 조회하여 movieInfoList에 저장함

믿고보는 영화 러버들의 별점왕 시리즈 컬렉션

구조 설명 ② - movieCollection - 누적 관객

```
1 usage
@Override
public List<MovieInfo> getAudiCollection() { // 누적 관객
    QueryResults<MovieInfo> results = query
        .selectFrom((movieInfo))
        .orderBy(QMovieInfo.movieInfo.audiAcc.desc())
        .limit(16)
        .fetchResults();
    List<MovieInfo> movieInfoList = results.getResults();
    return movieInfoList;
}
```

movieInfo 테이블의 **audiAcc(누적관객)** 컬럼을

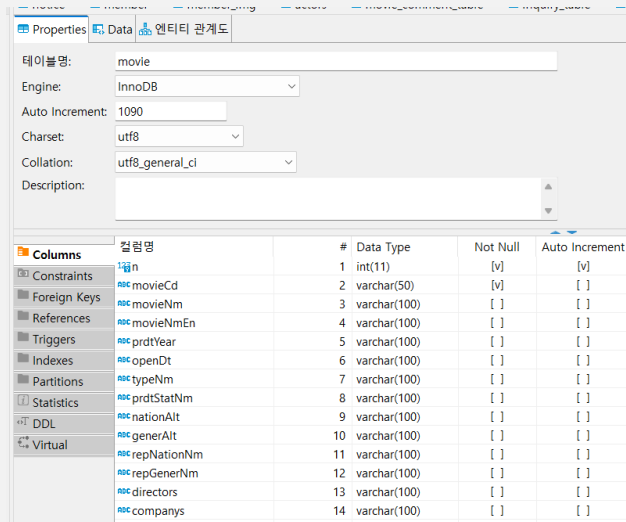
내림차순으로 조회하여 movieInfoList에 저장함

누적관객 보장 믿고보는 시리즈 컬렉션

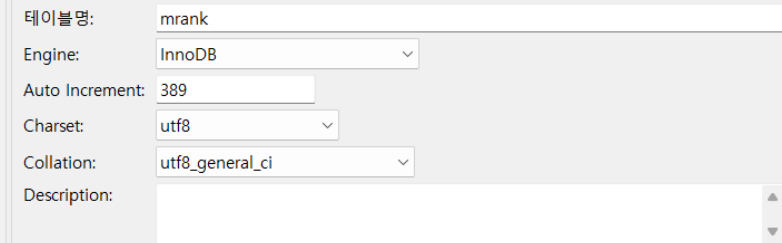
Part 4. 프로젝트 구조

4-7. JSON(영화 데이터)

- ✓ movieInfo,actors 테이블에 있는 영화, 배우 정보는 영화진흥위원회, 네이버 영화의 open api를 json으로 가져왔다.



| 컬럼명 | # | Data Type | Not Null | Auto Increment |
|-------------|----|--------------|----------|----------------|
| movie_id | 1 | int(11) | [v] | [v] |
| movieCd | 2 | varchar(50) | [] | [] |
| movieNm | 3 | varchar(100) | [] | [] |
| movieNmEn | 4 | varchar(100) | [] | [] |
| prdtYear | 5 | varchar(100) | [] | [] |
| openDt | 6 | varchar(100) | [] | [] |
| typeNm | 7 | varchar(100) | [] | [] |
| prdtStatNm | 8 | varchar(100) | [] | [] |
| nationAlt | 9 | varchar(100) | [] | [] |
| generAlt | 10 | varchar(100) | [] | [] |
| repNationNm | 11 | varchar(100) | [] | [] |
| repGenerNm | 12 | varchar(100) | [] | [] |
| directors | 13 | varchar(100) | [] | [] |
| company | 14 | varchar(100) | [] | [] |



| 컬럼명 | # | Data Type | Not Null | Auto Increment |
|----------|---|--------------|----------|----------------|
| mnum | 1 | int(11) | [v] | [v] |
| movieCd | 2 | varchar(20) | [] | [] |
| movieNm | 3 | varchar(100) | [] | [] |
| openDt | 4 | varchar(100) | [] | [] |
| salesAcc | 5 | varchar(100) | [] | [] |
| audiAcc | 6 | varchar(100) | [] | [] |



| 컬럼명 | # | Data Type | Not Null | Auto Increment | Key |
|---------------|---|--------------|----------|----------------|-----|
| actor_id | 1 | bigint(20) | [v] | [v] | PRI |
| actor_nm | 2 | varchar(255) | [] | [] | |
| cast_nm | 3 | varchar(255) | [] | [] | |
| movie_info_id | 4 | bigint(20) | [] | [] | MUL |

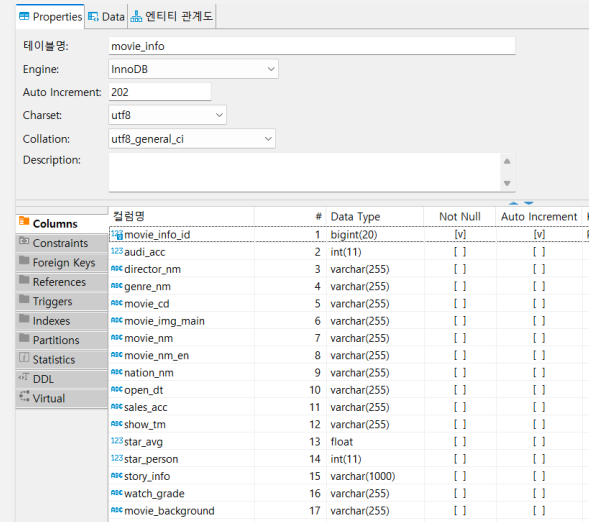
영화진흥위원회의 영화 상세정보,

일별 박스오피스 open api와

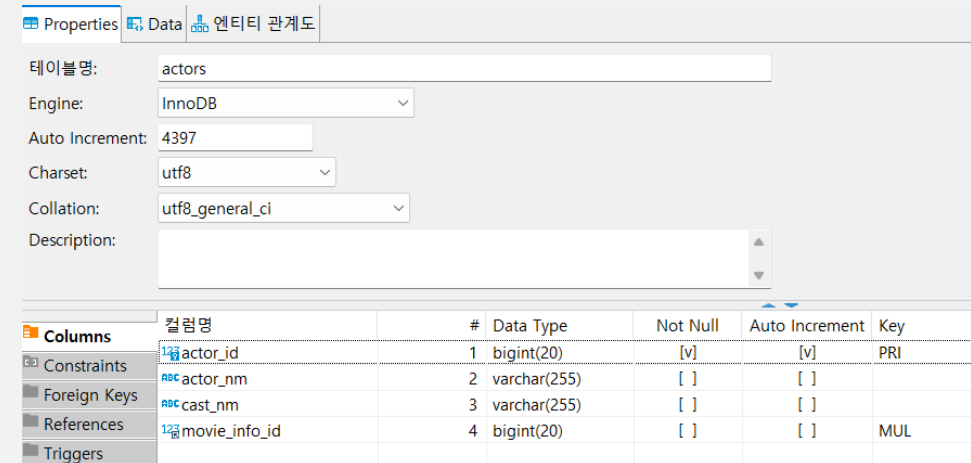
네이버 영화의 open api를 json으로

가져온 후 각각의 임시 테이블에 데이터

를 저장해 줌



| 컬럼명 | # | Data Type | Not Null | Auto Increment | Key |
|------------------|----|---------------|----------|----------------|-----|
| movie_info_id | 1 | bigint(20) | [v] | [v] | PRI |
| audi_acc | 2 | int(11) | [] | [] | |
| director_nm | 3 | varchar(255) | [] | [] | |
| genre_nm | 4 | varchar(255) | [] | [] | |
| movie_cd | 5 | varchar(255) | [] | [] | |
| movie_img_main | 6 | varchar(255) | [] | [] | |
| movie_nm_en | 7 | varchar(255) | [] | [] | |
| movie_nm | 8 | varchar(255) | [] | [] | |
| nation_nm | 9 | varchar(255) | [] | [] | |
| open_dt | 10 | varchar(255) | [] | [] | |
| sales_acc | 11 | varchar(255) | [] | [] | |
| show_tm | 12 | varchar(255) | [] | [] | |
| star_avg | 13 | float | [] | [] | |
| star_person | 14 | int(11) | [] | [] | |
| story_info | 15 | varchar(1000) | [] | [] | |
| watch_grade | 16 | varchar(255) | [] | [] | |
| movie_background | 17 | varchar(255) | [] | [] | |



| 컬럼명 | # | Data Type | Not Null | Auto Increment | Key |
|---------------|---|--------------|----------|----------------|-----|
| actor_id | 1 | bigint(20) | [v] | [v] | PRI |
| actor_nm | 2 | varchar(255) | [] | [] | |
| cast_nm | 3 | varchar(255) | [] | [] | |
| movie_info_id | 4 | bigint(20) | [] | [] | MUL |

각각 3개의 임시 테이블에서 실제 사용할 column만 각각 movie_info, actors 테이블에 저장해줌.

오시우

주소 : 대전광역시 서구

전화번호 : 010-4644-1453

이메일 : oyb010702@gmail.com

깃허브 : 주소