Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Головко Данила Валерійовича*
(ПІБ)

академічної групи *122-21ск-1*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка веб-додатку для підбору кінофільмів з використанням фреймворку Laravel та мови програмування JavaScript*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Спірінцев В.В.*			
розділів:				
спеціальний	*доц. Спірінцев В.В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2024

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2024 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-21ск-1* *Головко Данила Валерійовича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка веб-додатку для підбору кінофільмів з використанням фреймворку Laravel та мови програмування JavaScript*

затверджена наказом ректора НТУ «ДП» від *23.04.2024* № *375-С*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***25.05.2024 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***31.05.2024 р.***

Завдання видав _____ *доц. Спірінцев В.В*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Головко Д.В*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2024 р.*

Термін подання кваліфікаційної роботи до ЕК: *10.06.2024 р.*

РЕФЕРАТ

Пояснювальна записка: 93 с., 44 рис., 11 таблиць, 4 дод., 22 джерел.

Об’єкт розробки: веб-додаток для підбору кінофільмів.

Мета кваліфікаційної роботи: розробка веб-додатку для підбору кінофільмів з використанням фреймоворку Laravel та мови програмування JavaScript, що сприятиме поглибленню знань і досвіду в області веб-розробки, зокрема в проєктуванні та реалізації універсальних веб-додатків, а також застосування сучасних підходів і технологій на практиці.

У вступі викладено актуальність теми, сформульовано мету та об’єкти розробки, а також обґрунтовано вибір теми кваліфікаційної роботи.

У першому розділі проведено детальний аналіз предметної області, визначено основні вимоги до програмного продукту, обрано технології та інструменти для розробки.

У другому розділі описані етапи розробки веб-додатку, використання архітектурних патернів та принципів проєктування, докладно викладено алгоритми роботи програми, методи взаємодії з користувачем, а також здійснено опис програмної реалізації та виклику програми.

Економічний розділ містить розрахунок вартості розробки та впровадження програмного продукту.

Актуальність розробки системи для підбору кінофільмів полягає у все зростаючому інтересі до перегляду кінофільмів у цифровому форматі. Користувачі стикаються зі складністю у пошуку та підборі фільмів за власними вподобаннями. Веб додаток надасть можливість не тільки здійснювати пошук, але й отримати деталізовану інформацію про фільми, акторів, рейтинги та відгуки.

Практичне значення полягає у створенні веб-додатку для підбору кінофільмів, який забезпечує: інтуїтивно зрозумілий інтерфейс та інструменти фільтрації, що дозволяють швидко знайти потрібний фільм, а також функціонал для обміну думками через коментарі та онлайн-чати. Цей додаток надасть зручні інструменти як для користувачів, так і для адміністраторів, забезпечуючи простоту у використанні та ефективне управління контентом.

У майбутньому розвиток веб-додатку може включати впровадження машинного навчання для рекомендацій фільмів на основі попередніх переглядів та вподобань користувачів. Додаткові функції можуть також включати інтеграцію з різними стрімінговими сервісами для перегляду фільмів

Використання технологій JavaScript та PHP Laravel дозволить забезпечити зручність, швидкодію та масштабованість системи.

Список ключових слів: ВЕБ-ДОДАТОК, БАЗА ДАНИХ, PHP, HTML, CSS, JAVASCRIPT.

ABSTRACT

Explanatory note: 93 pages, 44 figures, 11 tables, 4 appendices, 22 references.

Object of development: an informational web application for selecting movies.

Purpose of the work: development of a web application for movie selection using the Laravel framework and JavaScript programming language, which will contribute to deepening knowledge and experience in the field of web development, particularly in the design and implementation of versatile web applications, as well as the practical application of modern approaches and technologies.

The introduction outlines the relevance of the topic, formulates the purpose and objectives of the development, and justifies the choice of the topic for the qualification work.

In the first chapter, a detailed analysis of the subject area was conducted, the main requirements for the software product were determined, and the technologies and tools for development were selected.

The second chapter describes the stages of web application development, the use of architectural patterns and design principles, provides detailed algorithms for the program's operation, user interaction methods, and a description of the program's implementation and invocation.

The economic chapter includes the calculation of the cost of development and implementation of the software product.

The relevance of developing an information system for selecting movies lies in the growing interest in watching movies in digital format. Users face difficulties in finding and choosing movies according to their preferences. The web application will provide not only search capabilities but also detailed information about movies, actors, ratings, and reviews.

The practical significance lies in the creation of a web application for movie selection, which provides: an intuitive interface and filtering tools that allow users to quickly find the desired movie, as well as features for sharing opinions through comments and online chats. This application will offer convenient tools for both users and administrators, ensuring ease of use and efficient content management.

Future development of the web application may include the implementation of machine learning for movie recommendations based on users' previous views and preferences. Additional features may also include integration with various streaming services for movie viewing.

The use of JavaScript and PHP Laravel technologies will ensure convenience, speed, and scalability.

List of keywords: WEB APPLICATION, DATABASE, PHP, HTML, CSS, JAVASCRIPT.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ - Програмне забезпечення

ОС - Операційна система

БД - База даних

СУБД - Система управління базами даних

CMS - Система управління контентом

CSS - Cascading Style Sheets

SQL - Structured Query Language

HTML - Hyper Text Markup Language

ВСТУП

В епоху веб технологій люди віддають перевагу онлайн-платформам для підбору, перегляду та обговорення кінофільмів. На сьогоднішній день вже існують такі сервіси як IMDB та Justwatch — інтернет-ресурс, які надають інформацію про кінофільми. Серед переваг сервісів подібного типу можна виділити вузьку направленість, зручність пошуку. Проте сучасні інтернет-ресурси не завжди забезпечують достатньо функцій для зручного пошуку фільмів і спілкування користувачів.

Актуальність розробки веб-додатку для підбору кінофільмів підтверджується зростаючим інтересом до перегляду кінофільмів у онлайн форматі. Користувачі стикаються зі складнощами в пошуку фільмів за власними вподобаннями через обмежені можливості наявних платформ. Дана веб-система надасть комплексні інструменти для пошуку фільмів, включаючи детальну інформацію про акторів, жанри, рейтинги та відгуки. Використання сучасних технологій, таких як JavaScript та PHP Laravel, забезпечить масштабованість та зручність системи.

Об'єкт дослідження: веб-додаток для підбору кінофільмів.

Мета роботи: розробка веб-додатку для підбору кінофільмів, що сприятиме отриманню досвіду в проєктуванні та реалізації веб-сайтів, а також поглибленню важливих знань у написанні універсальних програм мовами PHP, HTML, CSS та JavaScript. Цей проєкт сприяє подальшому вивченню методів, принципів та підходів до написання ефективного програмного забезпечення.

Для досягнення мети необхідно вирішити наступні завдання:

− проаналізувати особливості функціонування вже існуючих веб-додатків у кіноіндустрії;

− визначити основні вимоги до програмного продукту (функціональні характеристики, безпека, програмна сумісність);

− обрати технології та мови програмування для створення веб-додатку;

− описати структуру додатку та алгоритми її функціонування;

− впровадити адміністрування контенту (додавання та оновлення даних про фільми, жанри, акторів);

− інтегрувати функції оцінювання, коментування та спілкування між користувачами;

− забезпечити захист інформації та безпеку даних користувачів;

Практичне значення полягає у створенні веб-додатку для підбору кінофільмів, який забезпечує: зрозумілий інтерфейс та інструменти фільтрації які дозволяють швидко знайти потрібний фільм і дозволяє користувачам обмінюватися думками через коментарі та онлайн-чати, а також створити комфортний інструмент для користувачів та адміністраторів.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Рекомендаційні системи є ключовою складовою сучасних веб-додатків, що надають користувачам персоналізовані пропозиції. Вони аналізують поведінку користувачів і допомагають знайти контент, який найкраще відповідає їхнім уподобанням.

Content-based Filtering (Контентна фільтрація):

– базується на атрибутах об'єктів (наприклад, жанр, актори);

– використовує схожість між контентом, який користувач раніше оцінив, і новими елементами;

– обмежена різноманітність рекомендацій.

Collaborative Filtering (Колаборативна фільтрація):

– враховує взаємодію користувача з іншими користувачами;

– user-based Collaborative Filtering (Фільтрація на основі користувачів): рекомендує елементи, які подобаються подібним користувачам;

– item-based Collaborative Filtering (Фільтрація на основі елементів): рекомендує схожі елементи, які оцінювали інші користувачі;

– краще враховує різноманітність смаків.

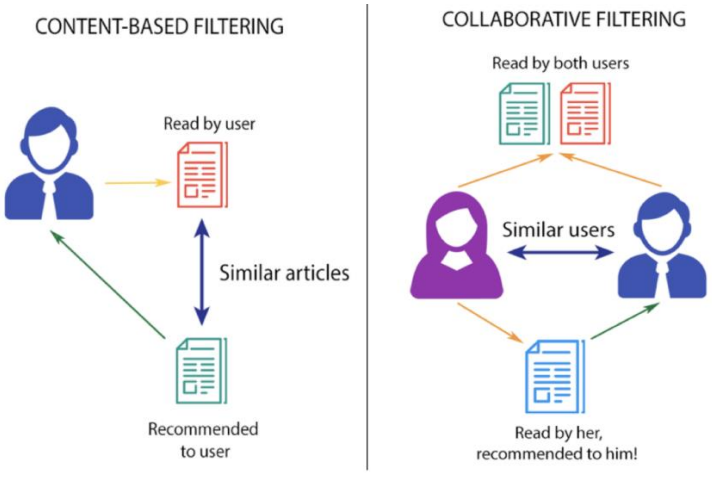

Рис. 1.1. Фільтрація на основі контенту

Машинне навчання дозволяє побудувати більш складні та точні моделі рекомендацій. Для створення системи рекомендацій можна використовувати такі алгоритми, як KNN (K-nearest neighbors), який обчислює схожість між фільмами за допомогою метрик, таких як косинусна відстань.

В майбутньому для розвитку веб-додатку можна буде використовувати штучний інтелект та машинне навчання для створення більш точних та ефективних рекомендацій. Наразі, ці функції не реалізовані, але планується їх впровадження в майбутніх версіях додатку.

При розробці додатку були розглянуті сайти, які орієнтовані на пошук кінофільмів та їх перегляд. Дослідження функціоналів сайту допомогло з'ясувати, які дані потрібно зберігати та які параметри пошуку можна використовувати при організації фільтрування списку фільмів.

Під час проведення дослідження, було розглянуто функціонал сайту IMDb («IMDb» — найбільший інтернет-сервіс про кіно) [1]. Найбільша база даних та вебсайт про кінематограф. База даних значною мірою заповнюється добровольцями. У базі зараз зібрана інформація про 10,1 млн фільмів і телесеріалів, є інформація майже про 11,5 млн акторів, режисерів та інших професіоналів кіно зі всього світу [2].

Дуже цікавим явищем на IMDb є онлайн-голосування. Кожний відвідувач сайту може проголосувати за свої улюблені фільми, і навпаки, проголосувати проти не улюблених. Таким чином на IMDb виявлені 250 найкращих фільмів всіх часів, а також 100 найгірших фільмів [2].

Розрахунок зваженого рейтингу (WR, weighted rating) фільмів виробляється на основі справжньої оцінки Баєса за спеціальною формулою:

$$WR = \frac{v}{(v+m)} \cdot R + \frac{m}{(v+m)} \cdot C \tag{1.1}$$

де:

WR — підсумковий зважений рейтинг;

R — середня оцінка фільму (за десятибальною шкалою);

v — кількість голосів, що було віддано за фільм;

m — мінімальна кількість голосів для включення в рейтинг (на цей час 25 000);

C — середня оцінка серед усіх фільмів (на грудень 2012 року C = 7,1).

Було встановлено, що цей сайт має додаткові параметри пошуку, систему рейтингу та відгуків, а також корисний функціонал кабінетів користувачів. На підставі цього до даної кваліфікаційної роботи буде реалізовано додаткові параметри пошуку, а також функціонал рейтингу та відгуків.

JustWatch - це веб-сайт, який надає інформацію про доступність фільмів і телешоу на різних легальних потокових платформах, таких як Netflix, HBO Max, Amazon Prime Video, Hulu, Apple TV та Disney+ та інших [3].

JustWatch функціонує як пошукова система, збираючи інформацію про доступність в Інтернеті фільмів та серіалів із сервісів потокового відео на запит.

Він об'єднує інформацію з більш ніж 100 бібліотек відеоконтенту, а також надає інформацію про якість дозволу відео, ціни та варіанти купівлі або оренди [4]. Користувачі також можуть створювати списки шоу та фільмів та ділитися цими списками з іншими користувачами. Що кажучи про Justwatch, то сайт має зручний та інтуїтивно зрозумілий інтерфейс.

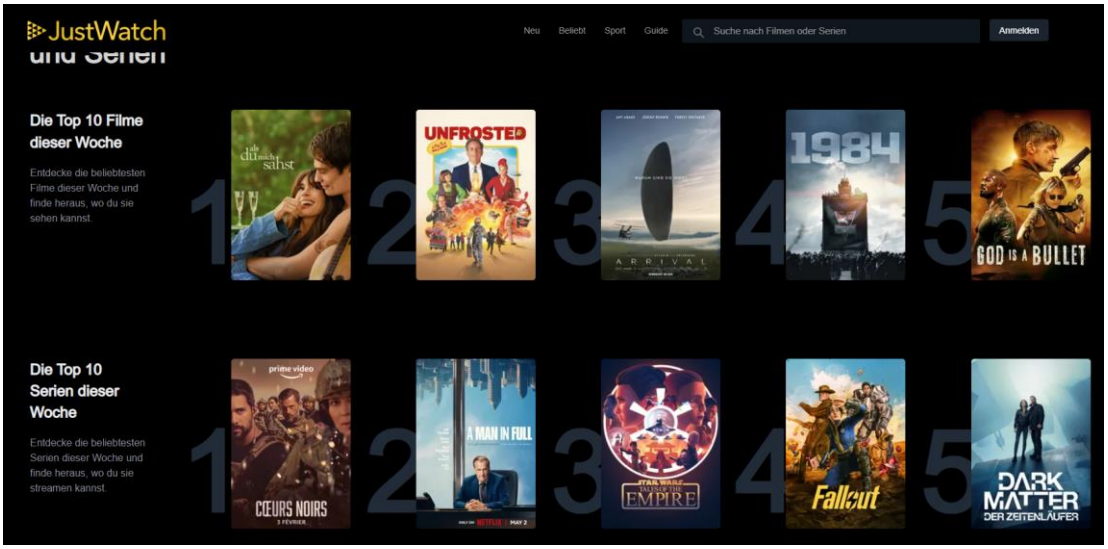

Рис. 1.2. Інтерфейс сайту Justwatch

Подібний інтерфейс також буде реалізовано у даній кваліфікаційній роботі. Сайт має зручну пошукову систему і, взагалі, роботу з інформацією про кінофільми, є підробки за тегами з фільтрами, а також інформація про акторів.

З цього приводу буде реалізовано інформація про акторів для кожного кінофільму, клікабельне посилання на інформацію про нього.

Також були розглянуті і інші сайти з подібною тематикою та, проаналізовавши їх, було помічено відсутність онлайн обговорення кінофільмів. Виходячи з цього, буде додано онлайн-чат для спілкування користувачів.

1.2. Призначення розробки та галузь застосування

Як об'єкт розробки розглядається веб-додаток для підбору кінофільмів, що забезпечує ефективний пошук та обговорення фільмів серед користувачів. Система дозволяє здійснювати швидкий пошук фільмів за різними параметрами та отримувати детальну інформацію про кінофільми, включаючи акторів, жанри, рейтинги та відгуки.

Розроблений додаток призначений для:

− надання інструментів для вибору та перегляду кінофільмів за допомогою інтуїтивного інтерфейсу;

− створення інструментів адміністрування контенту та захисту даних;

− забезпечує управління вмістом бази даних (додавання, редагування інформації про фільми, жанри та акторів) та перегляд статистики;

− можлівість користувачам залишати оцінки, коментарі та вести онлайн-чати про кінофільми;

− інтуїтивно зрозумілий інтерфейс із деталізованим контентом та підтримкою відображення для будь-яких пристроїв.

Веб-додаток повинен пропонувати можливості швидко знаходити фільм на будь-який смак, щоб допомогти та полегшити глядачу підбору кінофільм для перегляду. Основна мета цього додатку — забезпечити користувачам інтуїтивно зрозумілий, зручний та ефективний інструмент, який дозволяє швидко знайти фільми.

1.3. Підстава для розробки

Відповідно до ОПП, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу (проект). Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки (виконання кваліфікаційної роботи) є:

– ОПП за спеціальністю 122 «Комп'ютерні науки»;

– навчальний план та графік навчального процесу;

– наказ ректора Національного технічного університету «Дніпровська політехніка» № 375-С від 23.04.2024р.;

– завдання на кваліфікаційну роботу на тему «Розробка веб-додатку для підбору кінофільмів з використанням фреймворку Laravel та мови програмування JavaScript».

1.4. Постановка завдання

Впродовж роботи над кваліфікаційної роботи передбачається розробка веб-додатку, призначеного для підбору кінофільмів, з використанням технологій JavaScript для клієнтської частини та PHP Laravel для серверної частини. Система надаватиме користувачам доступ до обширної бази даних фільмів, включаючи описи, рейтинги, відгуки та інформацію про акторів і жанри.

Основні цілі:

− створення бази даних: організація бази даних, яка буде включати інформацію про кінофільми, акторів, рейтинги, жанри, користувачів. База даних має бути оптимізована для швидкого пошуку та обробки запитів.

− розробка веб-інтерфейсу: створення користувацького інтерфейсу, який дозволяє ефективно взаємодіяти з базою даних, включаючи функції пошуку, перегляду інформації про фільми, а також ведення облікового запису користувача.

− функціональність адміністрування: система має включати інструменти для адміністрування контенту, такі як додавання нових фільмів, оновлення даних про акторів та редагування інформації про жанри.

− інтеграція функцій оцінювання та спілкування: впровадження можливостей для обговорення фільмів через чати та коментарі, щоб користувачі могли обмінюватися думками та рекомендаціями.

− забезпечення безпеки даних: реалізація механізмів захисту інформації в базі даних, включно з захистом від несанкціонованого доступу та забезпеченням цілісності і конфіденційності даних.

Етапи розробки:

− аналіз вимог: визначення технічних та функціональних вимог до системи на основі потреб користувачів та вимог ринку.

− проектування системи: розробка архітектури системи, включаючи структурне проектування бази даних, визначення логічної структури сторінок та інтерфейсу користувача.

− розробка: програмування серверної та клієнтської частин системи, реалізація функціональності згідно з вимогами.

− заповнення контентом: введення даних у систему, їх структурування та оптимізація.

− тестування: верифікація та валідація роботи системи, пошук та усунення помилок, перевірка відповідності вимогам безпеки.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Для успішного запуску та роботи з інтерфейсом необхідно встановити Docker, який дозволяє контейнеризацію та легку переносимість середовища розробки та запуску. Що дає змогу легко виконати налаштування середовища з допомогою попередньо налаштованих Docker-образів, які містять всі необхідні залежності та служби.

Користувацькі дані та налаштування програми мають зберігатися в базі

даних mySQL, що забезпечує централізоване та надійне зберігання інформації.

Система повинна відповідати таким фунціональним вимогам:

− сумісність із різними операційними системами та браузерами для забезпечення стабільної роботи;

− швидка відповідь на запити користувачів;

− можливість внесення змін до компонентів без негативного впливу на інші частини;

− забезпечення масштабованості додатку;

− мінімальна частота збоїв системи та стійкості бази даних;

− захист інформації від втрати, блокування та несанкціонованого редагування;

− детальна документація для спрощення підтримки та розширення функціоналу;

− забезпечення переносимості між різними програмно-апаратними платформами та можливість розгортання на будь-якому сервері;

− зберігання системних даних у реляційній базі даних із ефективним управлінням об'ємом та швидким доступом.

1.5.2. Вимоги до інформаційної безпеки

Вимоги до інформаційної безпеки повинні включаючи шифрування конфіденційних даних користувачів у базі даних, а також реалізацію механізмів автентифікації та авторизації для контролю доступу до ресурсів [5].

SQL-ін'єкція відбувається, коли зловмисник вводить шкідливий SQL код у вхідний параметр, який потім передається та виконується сервером бази даних [6]. Це може призвести до несанкціонованого доступу до даних, їх модифікації або навіть видалення.

Заходи щодо запобіганню SQL-ін'єкцій:

− використовування підготовлених запитів з параметризованими SQL-запитами;

− уникання SQL-інструкцій за допомогою рядків із введенням користувача;

− валідація та нормалізацію введених даних, щоб не допустити шкідливого коду;

Cross-Site Scripting (XSS) відбувається, коли зловмисник вміщує шкідливі скрипти у контент, який потім відображається іншим користувачам [7]. Ці скрипти можуть виконуватися в браузері, дозволяючи атакуючому красти cookie-файли, сесії, паролі або навіть переписувати вміст веб-сторінки на клієнтській стороні.

Заходи щодо запобіганню XSS:

− екранізація всіх вхідних даних, що відображаються на сторінці;

− використовування Content Security Policy (CSP) для запобігання виконанню довільного коду;

− забезпечення кодування символів для запобігання ін'єкціям у HTML та JavaScript;

Cross-Site Request Forgery (CSRF) відбувається, коли зловмисник змушує користувача відправити запит на веб-сервер, в якому вони вже аутентифіковані, з метою виконання дій без їхнього відома або згоди [8].

Заходи щодо запобіганню CSRF:

− використання унікальних токенів CSRF для кожного запиту на зміну даних;

− застосування додаткових рівнів автентифікації для важливих операцій.

В рамках кваліфікаційної роботи буде реалізовано низку заходів використовуючи функціональні можливості, що надаються фреймворком Laravel. Зокрема, буде забезпечено шифрування конфіденційних даних користувачів у базі даних за допомогою вбудованих механізмів Laravel для шифрування даних.

Для запобігання SQL-ін'єкціям буде використовуватись підготовлені запити з параметризованими SQL-запитами, що є стандартною практикою у Laravel завдяки використанню Eloquent ORM.

Щодо запобігання Cross-Site Scripting (XSS), вхідні дані, що відображаються на сторінці, будуть екранізовані автоматично завдяки Blade шаблону у Laravel, що забезпечує належне кодування символів.

Для захисту від Cross-Site Request Forgery (CSRF) у Laravel використовуються CSRF токени, які автоматично генеруються та перевіряються на кожному запиті, що змінює дані. Таким чином, використання вбудованих функцій Laravel дозволяє значно спростити та підвищити ефективність впровадження заходів інформаційної безпеки у веб-додатку.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для забезпечення функціонування веб-додатку необхідний персональний комп'ютер, планшет або мобільний телефон із доступом до інтернету та встановленим сучасним веб-браузером. Технічні характеристики пристроїв користувачів не мають строгих обмежень, оскільки обробка даних здійснюється на стороні сервера.

1.5.4. Вимоги до інформаційної та програмної сумісності

Ці вимоги забезпечують сумісність та стабільну роботу розроблюваного веб-додатку в різних середовищах та з різними програмними компонентами:

- операційна система: Unix, Linux, Microsoft Windows 10;
- інтернет-браузер: Microsoft Internet Explorer 10 або вище, Mozilla Firefox 11 або вище, Opera 12.1 або вище, Google Chrome 16 або вище.

Веб-орієнтована система має бути реалізована з використанням наступних технологій:

- мова програмування: PHP для серверної частини
- верстання веб-сторінок: HTML5 і CSS3;
- клієнтські сценарії: бібліотека jQuery;
- система управління базами даних (СУБД): MySQL.

РОЗДІЛ 2

ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Веб-додаток повинен мати адаптивний, зручний для користувача дизайн.

Програма повинна виконувати такі функції роботи з фільмами:

− додавання, редагування та видалення кінофільмів;

− перегляд детальної інформації про кінофільми;

− пошук кінофільмів за наступними параметрами: за назвою, за іменами актерів та режисів. До пошуку можуть бути застосовані додаткові параметри: жанр, країна, рік;

− сортування інформації з наступними параметрами: за назвою, за датою виходу, за порядком. Сортування може бути застосовано до результатів пошуку або фільтрів.

− можуть бути застосовані фільтри: за жанром, за країною, за ріком кінофільма.

Завдання кваліфікаційної роботи повинно бути реалізоване з використанням таких ролей користувачів: «Адміністратор», «Користувач».

Адміністратор додатку повинен мати змогу управляти інформацією про кінофільми та передивлятися інформацію щодо нових користувачів додатку.

Користувач повинен мати змогу шукати та переглядати інформацію про кінофільми, залишати оцінки та відгуки, а також вести бесіди в онлайн чаті сайту.

Веб-додаток повинний забезпечувати валідацію введених даних для запобігання помилок. Інформація повинна зберігатися в структурованому та безпечному форматі та забезпечувати швидку відповідь на запити користувачів.

Для захисту даних застосувати сучасні методи шифрування даних користувачів.

2.2. Опис застосованих математичних методів

Під час створення програмного продукту для кваліфікаційного проєкту були застосовані логічні оператори для визначення умов і вибору подальших дій. Оператор || (АБО) застосовувався для перевірки виконання принаймні одного з набору критеріїв, тоді як оператор && (І) використовувався для переконання в тому, що всі зазначені умови є істинними одночасно. Для обертання логічного значення умови часто вживався оператор ! (НЕ), який дозволяв змінювати її на протилежне.

2.3. Опис використаної архітектури та шаблонів проектування

У процесі розробки веб-додатку було обрано комплексний підхід до архітектури та використання шаблонів проектування, заснованих на найкращих практиках програмування. В основу архітектури закладено модель MVC що дозволяє розділити логіку додатку на три ключові компоненти, забезпечуючи високу модульність та спрощення підтримки коду [10].

Компоненти MVC:

− модель (Model). Відповідає за зберігання даних, логіку та правила бізнесу.

− вид (View). Відображення інформації, яка представлена у моделі, користувачу.

− контролер (Controller). Забезпечує зв'язок між користувачем та системою, керує вхідними даними користувача та використовує моделі для виконання запитів користувача.

Принципи об'єктно-орієнтованого програмування та SOLID були прийняті як основа для створення гнучкого та масштабованого коду. SOLID допоміг забезпечити низьку зв'язаність між компонентами, можливість легкої заміни та розширення частин програми без втручання в інші модулі.

Принципи SOLID [11]:

− single-responsibility principle: кожен клас має мати одну єдину відповідальність.

− open-closed principle: сутності мають бути відкритими для розширення, але закритими для модифікації.

− liskov substitution principle: об'єкти у програмі можна замінювати їх підтипами без впливу на правильність виконання програми.

− interface segregation principle: багато спеціалізованих інтерфейсів краще ніж один універсальний.

− dependency inversion principle: модулі високого рівня не повинні залежати від модулів низького рівня. Обидва типи модулів повинні залежати від абстракцій.

Принципи DRY і KISS були застосовані для мінімізації повторень та спрощення розуміння коду [12,13]. DRY принцип націлений на зменшення повторення інформації програмного коду. KISS принцип, який закликає до спрощення дизайну та імплементації. Мета полягає в тому, щоб розробка та подальше утримання системи були якнайпростішими та зрозумілими. що робить систему легшою для підтримки та розвитку. Це дозволило уникнути дублювання логіки та забезпечити високий рівень перевикористання коду. В основі Laravel лежить архітектура, що дозволяє відділяти бізнес-логіку від логіки обслуговування клієнтських запитів, що ідеально підходить для розробки веб-додатків з високим рівнем взаємодії між клієнтом і сервером [14]. Яка зазвичай включає три основні рівні: презентаційний рівень (або клієнтська частина), логіка додатку (або серверна частина), і рівень даних. Ця структура дозволяє ефективно розділити відповідальність за різні аспекти додатку, спрощуючи тим самим розробку, тестування, і підтримку.

Ось детальний опис кожного рівня:

1 Презентаційний рівень (Presentation Layer). Цей рівень відповідає за взаємодію з кінцевим користувачем. У моєму випадку, це комбінація HTML, CSS, і JavaScript, які виконуються у веб-браузері користувача.

2 Логіка додатку (Business Logic Layer). На цьому рівні виконуються всі обчислення та обробка даних, які є необхідними для виконання бізнес-вимог програми. Це зазвичай контролери та сервіси, які обробляють дані, отримані від клієнта, і формують відповіді. Контролери керують маршрутами вхідних запитів та відповідей HTTP, взаємодіючи з моделями для виконання бізнес-логіки.

3 Рівень даних (Data Layer). Це база даних і механізми управління даними, які використовуються для зберігання та відновлення даних. Використовуємо Eloquent ORM, один із найпопулярніших ORM-інструментів у PHP-світі, який дозволяє взаємодіяти з базою даних за допомогою об'єктно-орієнтованих моделей [15]. Це забезпечує чітке відділення логіки додатку від логіки управління даними, сприяючи чистоті коду і легкості підтримки.

2.4.Опис використаних технологій та мов програмування

Додаток кваліфікаційної роботи було написано з використанням середовища PhpMyStorm.

Програмне забезпечення JetBrains PhpStorm є спеціалізоване засіб веб-розробки, орієнтовані на веб-додатки та інші види програм, які можна створювати за допомогою мови PHP і з використанням HTML, JavaScript і CSS. Рішення PhpStorm здійснює розгортання і синхронізацію проєктів через протокол FTP. Середа PhpStorm пропонує функції автоматичного завершення мовних конструкцій PHP в коді, інспектування коду, різні алгоритми рефакторинга і швидку навігацію по коду [16].

Підтримуються передові технології веб-розробки, включаючи HTML5, CSS, Sass, SCSS, Less, Stylus, Compass, CoffeeScript, TypeScript, ECMAScript Harmony, шаблони Jade, Zen Coding, Emmet, і, звичайно ж, JavaScript.

PhpStorm включає в себе всю функціональність ВебStorm (HTML / CSS редактор, JavaScript редактор) і додає повнофункціональну підтримку PHP і баз даних / SQL [17].

Ключові можливості:

− інтелектуальний редактор PHP коду з підсвічуванням синтаксису, автодоповнення коду, розширеними настройками форматування коду, запобіганням помилок нальоту;

− підтримує PHP 7.0, 5.6, 5.5, 5.4 і 5.3, генератори, співпрограми і все синтаксичні поліпшення;

− PHP рефакторингом, code (re) arranger, детектор дубльованого коду;

− підтримка Vagrant, Composer, вбудований REST клієнт, Command Line Tools, SSH консоль;

− підтримка фреймворків (MVC view для Symfony2, Yii) і спеціалізовані плагіни для провідних PHP фреймворків (Symfony, Magento, Drupal, Yii, CakePHP і багато інших) ;

− візуальний відладчик для PHP додатків, валідація конфігурації відладчика, PHPUnit з покриттям коду (підтримка PHPUnit 5), а також інтеграція з профілювальником;

− повний набір інструментів для фронтенд-розробки;

− підтримка стилів коду, вбудовані стилі PSR1 / PSR2, Symfony2, Zend, Drupal та інші;

− інтеграція з системами управління версіями, включаючи уніфікований інтерфейс;

− віддалене розгортання додатків і автоматична синхронізація з використанням FTP, SFTP, FTPS і ін;

− Live Edit: зміни в коді можна миттєво переглянути в браузері без перезавантаження сторінки;

− PHP UML;

− інтеграція з баг-трекера;

− інструменти роботи з базами даних, SQL редактор;

− багатоплатформовий (Windows, Mac OS X, Linux) [17].

Програма кваліфікаційної роботи написана мовами програмування:

- PHP 7.2;
- HTML 5;
- CSS 3.

Та з використанням таких фреймворків як:

- Bootstrap 4;
- Laravel.

PHP (англ. PHP: Hypertext Preprocessor) — скриптова мова програмування, була створена для генерації HTML-сторінок на стороні веб-сервера. PHP є однією з найпоширеніших мов, що використовуються у сфері веб-розробок (разом із Java, .NET, Perl, Python, Ruby). PHP підтримується переважною більшістю хостинг-провайдерів. PHP інтерпретується веб-сервером у HTML-код, який передається на сторону клієнта. На відміну від скриптової мови JavaScript, користувач не бачить PHP-коду, тому що браузер отримує готовий html-код. Це є перевагою з точки зору безпеки, але погіршує інтерактивність сторінок. Але ніхто не забороняє використовувати PHP для генерування JavaScript-кодів, які виконуються вже на стороні клієнта [21].

В області побудови веб-сайтів визначається наявністю великого набору вбудованих засобів і додаткових модулів для розробки веб-додатків. [

Основні переваги через які було обрано PHP:

1. Розробка за допомогою PHP дає багато можливостей. При належному рівні володіння, за допомогою шаблонізатора можна створювати не тільки сценарії для веб-додатків, але й повноцінні програми.

2. Вивчення PHP не вимагає багато часу.

3. Багатоплатформений. PHP може бути запущений в будь-якій операційній системі, включаючи юніксоід.

4. Підтримка веб-серверів. Складно знайти, який би не працював з PHP.

5. Безкоштовне розповсюдження. Можливо, PHP не був такий популярний для створення веб-додатків, якби не був безкоштовним, як і більшість інструментів для роботи з ним [21].

Для реалізації відображення веб-сторінок було використано HTML. (HyperText Markup Language) - мова розмітки гіпертексту - призначений для створення Веб-сторінок. Під гіпертекстом в цьому випадку розуміється текст, пов'язаний з іншими текстами покажчиками-посиланнями. HTML являє собою досить простий набір кодів, які описують структуру документа [18].

При розробці додатку було обрано HTML5. HTML5 надає такі переваги:

− «чистий» код за рахунок видалення усього зайвого з коду і перерозподілу функціональних елементів;

− поліпшені можливості по роботі сайтів на різних операційних системах і з різними браузерами;

− використання веб-браузера для прослуховування музики в режимі онлайн, перегляду відео і відтворення flash-анімації;

− взаємодія з сервером піддалося оптимізації, що дозволило полегшити роботу з сайтом кінцевого користувача.

При створенні дизайну веб-сторінок було використано каскадні таблиці стилів CSS. CSS - це формальна мова, яка служить для опису оформлення зовнішнього вигляду документа, створеного з використанням мови розмітки (HTML, XHTML, XML). Отже, завдяки CSS можна встановлювати кольори, шрифти, розташування окремих блоків і інших аспектів представлення зовнішнього вигляду веб-сторінок.

Основною метою розробки CSS було відділення опису логічної структури веб-сторінки (яке проводиться за допомогою HTML або інших мов розмітки) від опису зовнішнього вигляду цієї веб-сторінки (яке тепер проводиться за допомогою формальної мови CSS) [19].

Bootstrap - це фреймворк на основі HTML і CSS. Він містить стилі для основних елементів, які застосовуються в верстці. Використання такого фреймворка значно прискорює процес створення сторінок. Стандартні стилі легко міняти, що забезпечує гнучкий і простий процес створення макетів сайтів.

Переваги фреймворка Bootstrap:

1 Висока швидкість створення якісної адаптивної верстки навіть початківцями веб-розробниками.

2 Багатобраузерность і багатоплатформеність (коректне відображення та робота сайту у всіх підтримуваних цим фреймворком браузерах)

3 Наявність великої кількості готових добре продуманих компонентів, протестованих величезним співтовариством веб-розробників на різних пристроях.

Bootstrap пропонує використання сітки, що допомогає в створенні адаптивного макету. Сітка дозволяє задавати класи для блоків, які вказують, яку ширину повинен займати елемент і як він відображатиметься на різних пристроях. Сітка функціонує як таблиця, в якій є свої ряди і стовпці, а максимальна кількість стовпців 12 [20].

Окрім сітки в Bootstrap існує величезна кількість всіляких компонентів: навігаційні меню, форми, таблиці, модальні вікна, вкладки, сповіщення, спливаючі підказки і [20] т.д.

Laravel — це фреймворк для веб-додатків з виразним і елегантним синтаксисом. Він дозволить спростити вирішення основних наболілих завдань, таких як аутентифікація, маршрутизація, сесії і кешування. Laravel - це спроба об'єднати все найкраще, що є в інших PHP фреймворк. [14]

Основні переваги Laravel:

1 Велика екосистема з миттєвим розгортанням своєї платформи. Офіційний сайт надає безліч інформації для ознайомлення.

2 У Laravel є свій движок для шаблонів Blade, «гарний» синтаксис мови, який сприяє вирішенню всіх необхідних завдань, таких як аутентифікація, сесії, кешування і маршрутизація RESTful.

Eloquent ORM - реалізація шаблону проєктування ActiveRecord на PHP. Дозволяє строго визначити відносини між об'єктами бази даних. Стандартний для Laravel будівник запитів Fluent підтримується ядром Eloquent [15].

2.5. Опис структури програми та алгоритмів її функціонування

При дослідженні предметної області для реалізації програми кваліфікаційної роботи була обрана СУБД MySQL. При розробці схеми бази даних було прийнято поділити її на дві основні сутності: «Фільми», «Користувачі». Повну ER-діаграму представлено в додатку А.

ER-діаграма, яка описує взаємодію сутностей групи «Фільми», представлена на рис. 2.1.

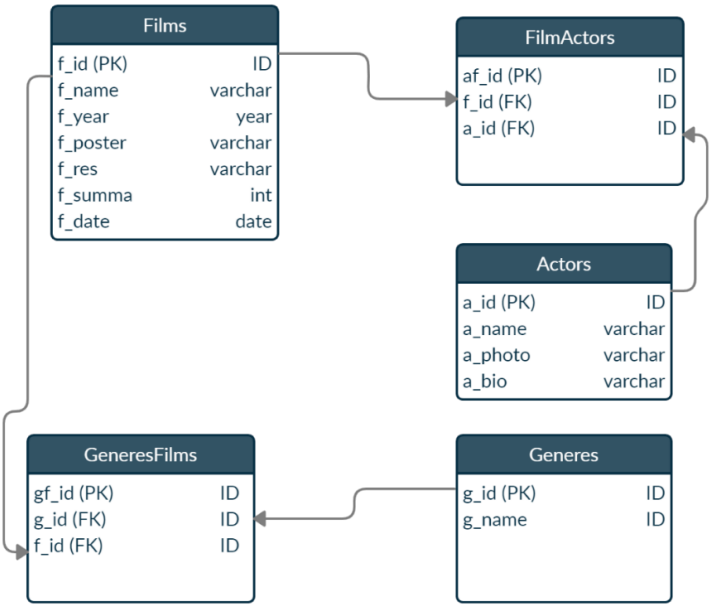

Рис. 2.1. ER-діаграма групи «Фільми»

Для взаємодії з таблицями бази даних фреймворк Laravel пропонує механізм об'єктно-орієнтованого відображення (ORM): таблиці бази даних представлені у вигляді класів, де властивості — це поля таблиці, а запис у таблиці — об’єкт класу. Всі класи повинні бути потомками класу Model, який містить методи для

взаємодії з безпосередньо базою даних: вибірки, фільтрації, сортування даних, додавання, редагування та видалення полів.

Для синхронізації класів та таблиць бази даних використовуються міграції: набір автоматично згенерованих файлів, які фіксують зміни у програмних класах та після спеціальної команди створюють сукупність SQL-запитів, які відправляються на сервер БД та змінюють структуру таблиць необхідним чином. Після розробки таблиць бази даних, вони були описані програмно у вигляді PHP-класу. Приклад опису FilmsModels представлено у лістингу 2.1.

Листинг 2.1 – Опис таблиці FilmsModels у вигляді класу

```
class FilmsModel extends Model
{
    use Sortable;
    use Rateable;
    public $sortable = ['f_id','f_name','f_date', 'f_summa'];
    public $timestamps = false;
    protected $primaryKey = 'f_id';
    use HasFactory;
    protected $fillable = [
        'f_name',
        'f_country',
        'f_res',
        'f_summa',
        'f_year',
        'f_date',
    ];
    public function images()
    {
        return $this->hasOne(Image::class);
    }
    public function actors()
    {
        return $this->belongsToMany(Actors::class);
    }
    public function generes()
    {
        return $this->belongsToMany(Generes::class);
    }
```

Таблиця FilmModels описує необхідну інформацію про кінофільми. Опис всіх полів FilmModels таблиці представлено у таблиці 2.1.

Таблиця 2.1

FilmModels — «Фільми»

Поле	Призначення
f_id	Ідентифікатор фільму, ключове поле
f_name	Назва фільму
f_year	Рік видавництва фільму
f_country	Країна-виробник фільму
f_res	Режисер фільму
f_summa	Бюджет фільму
f_date	Дата виходу фільму

Таблиця Actors, містить необхідні дані, що включають інформацію про акторів. Опис полів представлено у таблиці 2.2.

Таблиця 2.2

Actors — «Актори»

Поле	Призначення
a_id	Ідентифікатор актора, ключове поле
a_name	Ім’я та прізвище актора
a_photo	Фотографія актора
a_bio	Коротка біографія актора

Створено таблицю FilmActor, що містить інформацію про кінофільми та акторів, які знімались у цих фільмах.

Оскільки декілька фільмів може мати декілька акторів, між таблицями Films та ActorFilm відношення «багато до багатьох» відповідно. Опис полів представлено у таблиці 2.3.

Таблиця 2.3

FilmActor — «Фільм-Актор»

Поле	Призначення
af_id	Ідентифікатор таблиці, ключове поле
f_id	Ідентифікатор фільму
a_id	Ідентифікатор актору

Створено таблицю Generes, яка містить інформацію про жанри. Опис полів представлено у таблиці 2.4 Також була створена таблиця GeneresFilm що містить інформацію про фільм та його жанр. З огляду на те, що декілька фільмів може мати декілька жанрів відношення між таблицями Films та GeneresFilm встановлено «багато до багатьох» відповідно. Опис полів представлено у таблиці 2.5.

Таблиця 2.4

Generes — «Жанри»

Поле	Призначення
g_id	Ідентифікатор жанру, є ключовим полем
g_name	Назва жанру

Таблиця 2.5

GeneresFilm — «Фільм-Жанри»

Поле	Призначення
gf_id	Ідентифікатор таблиці, є ключовим полем
g_id	Ідентифікатор жанру
f_id	Ідентифікатор фільму

ER-діаграму групи «Користувачі» представлено на рис. 2.2.

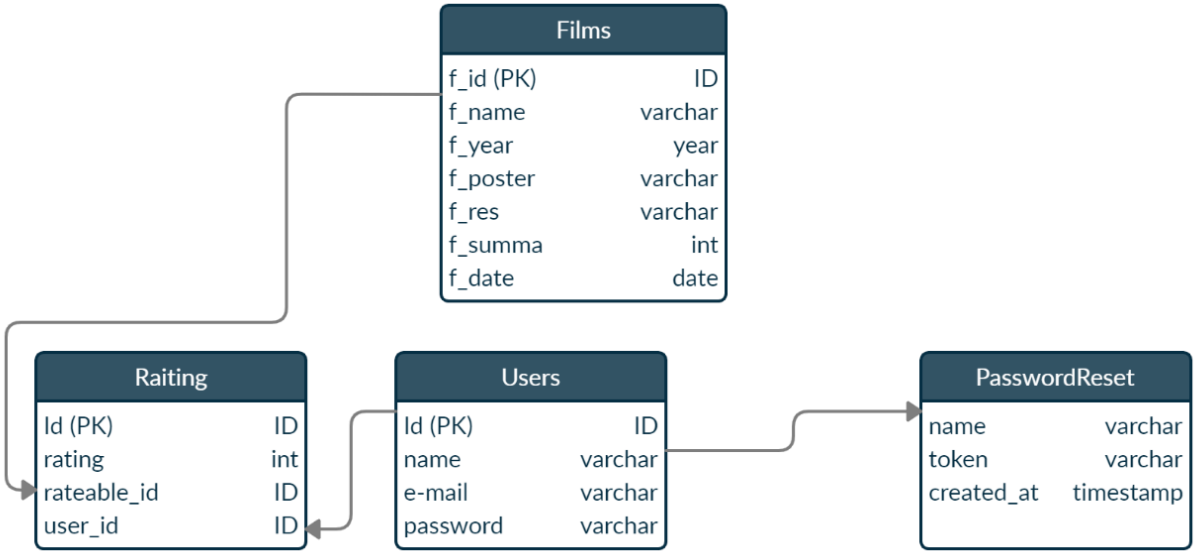

Рис. 2.2. ER-діаграма групи «Користувачі»

Створено таблицю Users, що містить інформацію про обліковий запис користувача. Опис полів представлено у таблиці 2.6.

Таблиця 2.6

Users — «Користувачі»

Поле	Призначення
id	Ідентифікатор користувача, є ключом полем
name	Нікнейм користувача
e-mail	Електрона адреса користувача
password	Пароль користувача
Role	Тип: користувач, адміністратор

Створено таблицю ResetPassword, що містить інформацію про скидання паролей користувача. Опис полів представлено у таблиці 2.7.

Таблиця 2.7

ResetPassword — «Скидання паролю»

Поле	Призначення
email	Електрона адреса користувача
token	Пароль користувача
create_at	Дата скидання паролю

Створено таблицю Raiting, що містить інформацію про оцінки користувачей щодо фільмів. Опис полів представлено у таблиці 2.8.

Таблиця 2.8

Raiting — «Рейтинг»

Поле	Призначення
id	Ідентифікатор, є ключом полем
raiting	Оцінка користувача
Relable_id	Ідентифікатор фільму
User_id	Ідентифікатор користувача

Створено таблицю Comments, що містить інформацію про коментарі користувачів щодо фільму. Опис полів представлено у таблиці 2.9.

Таблиця 2.9

Comments — «Коментарі»

Поле	Призначення
id	Ідентифікатор, є ключом полем
Commenter_id	Ідентифікатор користувача
comment	Комментар (текст)
Child_id	Ідентифікатор користувача для дочірнього коментарю
rating	Рейтинг коментарю
Created_at	Дата відправлення коментарю

Створено таблицю CommentsRating, що містить інформацію про оцінки коментарів користувачів щодо фільму. Опис полів представлено у таблиці 2.10.

Таблиця 2.10

CommentsRating — «Оцінювання коментарів»

Поле	Призначення
id	Ідентифікатор, є ключом полем
Commeter_id	Ідентифікатор користувача
Comment_id	Ідентифікатор коментарю
Comment_vote	Голос користувача

Створено таблицю Chat, що містить інформацію про листування користувачів. Опис полів представлено у таблиці 2.11.

Таблиця 2.11

Chat — «Чат»

Поле	Призначення
id	Ідентифікатор, є ключом полем
user_id	Ідентифікатор користувача
message	Текст повідомлення
created_at	Дата відправлення повідомлення

Для реалізації функціоналу чату в контролері Chat потрібно передбачити, щоб користувач був авторизований, що показано на діаграмі взаємодії, яка представлена на рис. 2.3.

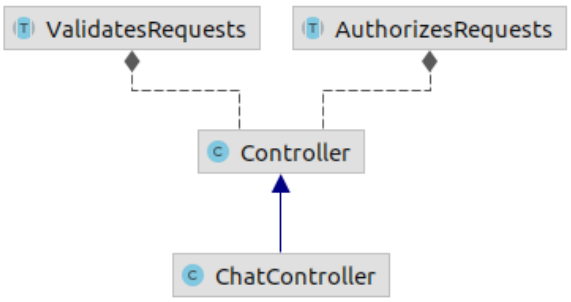

Рис. 2.3. Схема зв’язку класів контролера «Chat»

Для того, щоб авторизовані користувачі могли синхронно спілкуватися між собою, був використаний сокет підключення, що показано на діаграмі взаємодії, яка представлена на рис. 2.4.

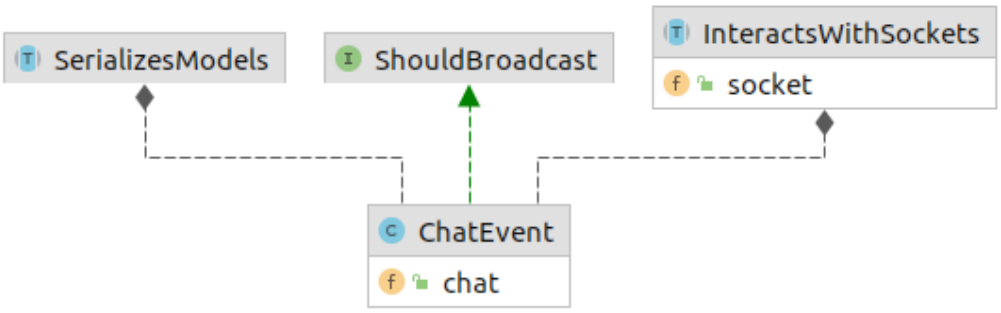

Рис 2.5. Схема зв’язку класів події «ChatEvent»

Поточний проєкт складається з трьох основних додатків: блоку переліку фільмів та перегляду інформації, системи авторизації та реєстрації, системи управління інформацією про кінофільми.

Додаток переліку фільмів та перегляду інформації містить такі сторінки: Layout_page, Review_page, Single_page, Actors_page, які можуть містити наступні блоки елементів:

Сторінка Layout_page складається з наступних компонентів:

− Main_navigation — меню, що містить основні компоненти переходу на інші сторінки;

− Main_content – загальна частина, що застосовується у сторінках виведення переліку кінофільмів.

Сторінка Review_page складається з наступних компонентів:

− Filters — компонент для вибору бажаного фільтру, включає в себе форми: generesInput – для застосування фільтру за жанрами, CountryInput – для застосування фільтру за країнами, YearInput – для застосування фільтру за ріком

− SearchName — компонент пошуку за іменем фільму;

− SearchwithFilters — компонент пошуку з додатковими параметрами, має такі форми як: SearchInput – введення інформації для пошуку фільмів за ім’ям або

фамілією актора чи режисера з можливо застосованих до них додаткових параметрів

− Movielist — виведення переліку фільмів.

Для сторінки було назначено наступні обробники подій:

− Filter — натискання на кнопку «Фільтрувати» для застосування фільтрів;

− Search — натискання на значок пошуку для застосування пошуку;

− Clear — натискання на кнопку «Очистити» для застосування пошуку.

Сторінка Single_page складається з наступних компонентів:

− MoviePoster — виведення зображення фільма;

− MovieMeta — виведення повної інформації про фільм;

− Raiting — виведення рейтингу фільму;

− CommentStore — виведення всіх коментарів;

− CommentReply — виведення форми відповіди на коментар;

− CommentEdit — виведення форми редагування коментар;

− CommentRaiting — виведення рейтингу комментарю.

Для сторінки було назначено наступні обробники подій:

− ActorOne — натискання на гіперсилку, що веде на сторінка інформації про акторів;

− RaitingSubmit — натискання на кнопку «Відправити оцінку»;

− CommentSubmit — натискання на кнопку «Відправити коментар»;

− CommentReplySubmit — натискання на кнопку «Відповісти»;

− CommentEditSumbit — натискання на кнопку «Оновити»;

CommentDelete — натискання на кнопку «Видалити» ;

− CommentRaitingAction — натискання на кнопку для оцінювання коментарю.

Сторінка Actors_page складається з наступних компонентів:

− ActorPoster — виведення зображення актора;

− ActorMeta — виведення повної інформації про актора.

Додаток системи авторизації та реєстрації містить такі сторінки: Auth_login_page, Auth_register_page, Auth_email_page, Auth_reset_page які можуть містити наступні блоки елементів.

Сторінка Auth_login_page складається з наступних компонентів:

− Email_adress — поле для введення e-mail користувача;

− Password — поле для введення паролю користувача.

Для сторінки було назначено наступні обробники подій:

− ForgotPassword — натискання на гіперсилку «ForgotPassword», що веде на сторінку Auth_Email_Page;

− Login — натискання на кнопку «Вхід» для входу до профілю.

Сторінка Auth_register_page складається з наступних компонентів

− Name — поле для введення нікнейму користувача;

− Email_adress — поле для введення e-mail користувача;

− Password — поле для введення паролю користувача;

− PasswordConfirm — поле для повторного введення паролю користувача

Сторінка Auth_email_page складається з наступних компонентів

− Email_adress — поле для введення e-mail користувача.

Для сторінки було назначено наступні обробники подій:

− Submit — натискання на кнопку «Відправлення письма» для відправлення електроного письма на вказану в полі електрону почту.

Сторінка Auth_reset_page складається з наступних компонентів:

− Password — поле для введення нового паролю користувача;

− PasswordAgain — поле для повторного введення нового паролю користувача.

Для сторінки було назначено наступні обробники подій:

− Submit — натискання на кнопку «Скинути пароль» для входу з новими даними користувача до профілю;

Додаток системи управління інформацією про кінофільми містить такі сторінки: Home_page, Info_index_page, Info_create_page, Info_edit_page, які можуть містити наступні блоки елементів:

Сторінка Home_page складається з наступних компонентів:

− SessionStatus — для встановлення статусу входу до профіля;

− CanIsAdmin — для доступу до адмін панелі.

Для сторінки було назначено наступні обробники подій:

− Info — натискання на гіперсилку, що веде на сторінку з переліками кінофільмів для подальшого застосування редагування, видалення або створення інформації,

Сторінка Info_index_page складається з наступних компонентів:

− Movielist – виведення переліку кінофільмів;

− InfoCreate – створення нової інформації про кінофільми;

− InfoEdit – редагування створеної інформації про кінофільми;

− InfoDelete – видалення інформації про кінофільми.

Для сторінки було назначено наступні обробники подій:

− InfoCreate — натискання на кнопку «New Info» для переходу на сторінку створення інформації;

− InfoEdit — натискання на кнопку «Edit» для переходу на сторінку редагування інформації;

− InfoDelete — натискання на кнопку «Delete» для видалення інформації.

Сторінки Info_create_page та Info_edit_page складається з наступних компонентів:

− Fimage — завантаження зображення фільму;

− Fname — поле для введення назви фільму

− Fcountry — поле для введення країни фільму;

− FRes — поле для введення режисера фільму;

− FSumma — поле для введення бюджету фільму;

− Fyear — поле для введення ріку фільму;

− Fdate — поле для введення дати виходу фільму;

− Factors — поле для введення актора, що містить варіанти вибору акторів

− Fgeneres — поле для введення жанру, що містить варіанти вибору жанрів.

Для сторінок було назначено наступні обробники подій:

− Create — натискання на кнопку «Create» для створення інформації;

− Edit — натискання на кнопку «Оновлення» для оновлення інформації.

Сторінка info_user_page складається з наступних компонентів:

− UserDiagram — виведення графіку нових користувачів.

Сторінка Chat_page складається з наступних компонентів

− ChatMassege — виведення повідомлень від користувачів;

− ActiveUsers — виведення інформації про онлайн користувачів;

− ChatTyping — показує інформації про користувача, який пише повідомлення в даний момент.

Для сторінки було назначено наступні обробники подій:

− ChatEnter — натискання на кнопку «Enter» на клавіатурі для відправлення повідомлення.

Основні модулі програмної системи та схему їх взаємозв’язку представлено на рис. 2.5.

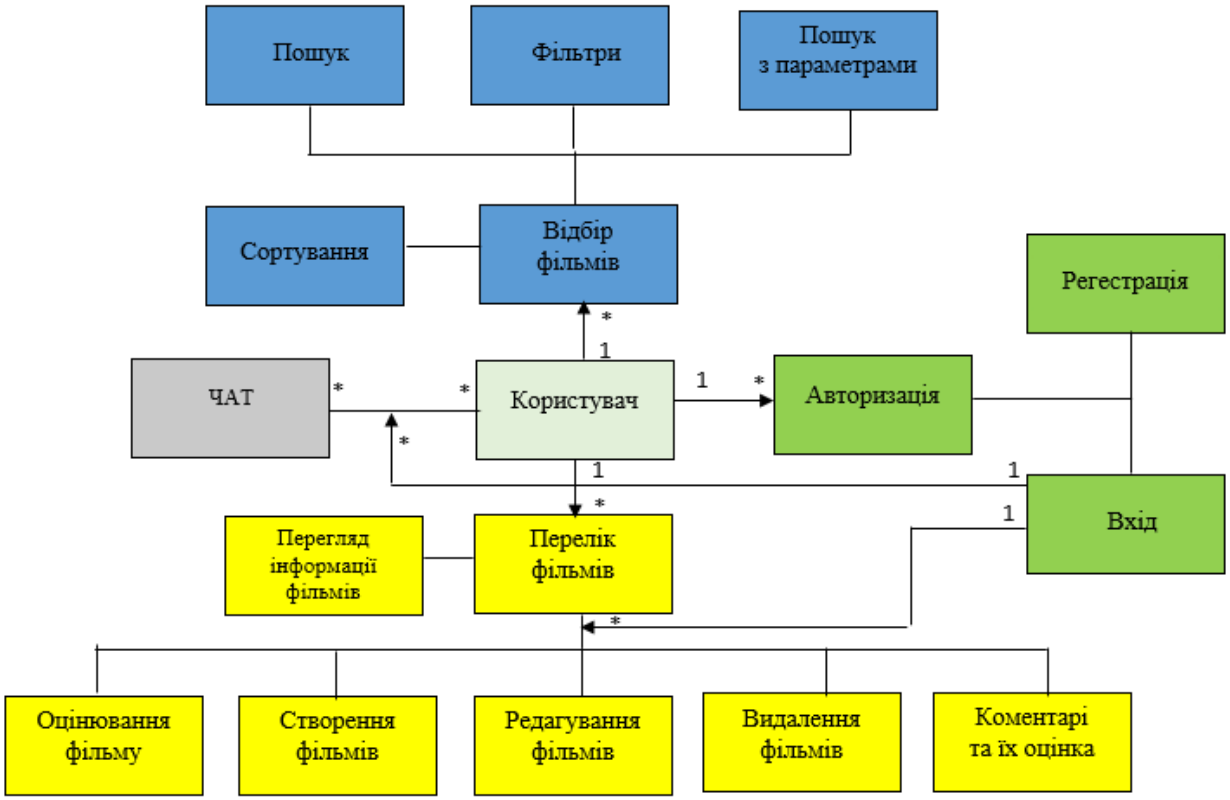

Рис. 2.5. Схема взаємозв’язку основних модулів

2.6. Обґрунтування та організація вхідних та вихідних даних програми

У розробленому веб-додатку для підбору кінофільмів, основу структури даних складає база даних (БД), що містить інформацію про кінофільми, акторів, жанри, а також користувачів, включаючи їх взаємодію через оцінки й коментарі.

Вхідні дані:

– Кінофільми: інформація про назву фільму, рік видання, країну-виробника, режисера, бюджет, дату випуску та список акторів.

– Актори: Включає ім'я та прізвище, фотографію, коротку біографію.

– Жанри: Класифікація фільмів за жанрами.

– Користувачі: інформація про користувачів, включно з нікнеймом, електронною адресою, паролем та за ролью (користувач, адміністратор).

– Рейтинг і коментарі: інформація про оцінки та відгуки, залишені користувачами, а також деталі голосування за коментарі.

Вихідні дані:

– Інформаційні сторінки фільмів: відображають зібрані дані про кінофільми.

– Профілі акторів: сторінки з детальною інформацією про акторів

Кодування даних: всі текстові дані кодуються у форматі UTF-8 для підтримки міжнародних символів та забезпечення коректного відображення інформації.

Введення даних: Користувачі вводять інформацію через веб-інтерфейс у діалоговому режимі.

2.7 Опис розробленого програмного продукту

2.7.1 Використані технічні засоби

Основні забезпечення серверу для робити ПЗ:

- об’єм оперативної пам’яті: 1ГБ;
- операційна система: Ubunty Linux 16;
- підтримка протоколів: HTTP;
- HTTP-сервер: Nginx, Gunicorn;
- підтримувані бази даних: MySQL.

Основні технічні засоби для роботи ПЗ:

- наявність пристрою з ОС 2010 року випуску та пізніше;
- наявність одного з перерахованих браузерів зі вказаною або вищою версією: Google Chrome 16, Opera 12.1, Mozilla Firefox 11, Internet Explorer
- Персональний комп’ютер повинен мати доступ до мережі Інтернет.

2.7.2 Використані програмні засоби

Для кожної частини веб-додатку було використано свої програмні засоби.

Клієнтська частина для створення інтерактивного та адаптивного інтерфейсу користувача:

- HTML5;
- CSS;
- JavaScript.

Серверна частина для створення комплексних веб-додатків, що забезпечує високий рівень абстракції та безпеки:

- PHP Laravel

Середовище розробки що дозволяє легко налаштовувати та масштабувати розробницьке середовище з урахуванням всіх необхідних залежностей:

- DevilBox на Docker.

2.7.3 Виклик та завантаження програми

Для запуску програми потрібно:

1. Завантажити Docker-контейнер Devilbox та налаштувати його згідно з рекомендаціями для розгортання середовища Laravel.

2. Помістити код програми в директорію, яка синхронізована з Docker-контейнером.

3. Запустити Devilbox за допомогою Docker, що автоматично ініціює необхідні сервіси (Nginx, MySQL, PHP).

4. Використовуючи веб-браузер, ввести адресу локального сервера 127.0.0.1

2.7.4 Опис інтерфейсу користувача

Переглядати головну сторінку сайту можуть як авторизовані, так і не авторизовані користувачі. Для користувача сторінка з списком фільмів буде мати вигляд, який представлено на рис. 2.5

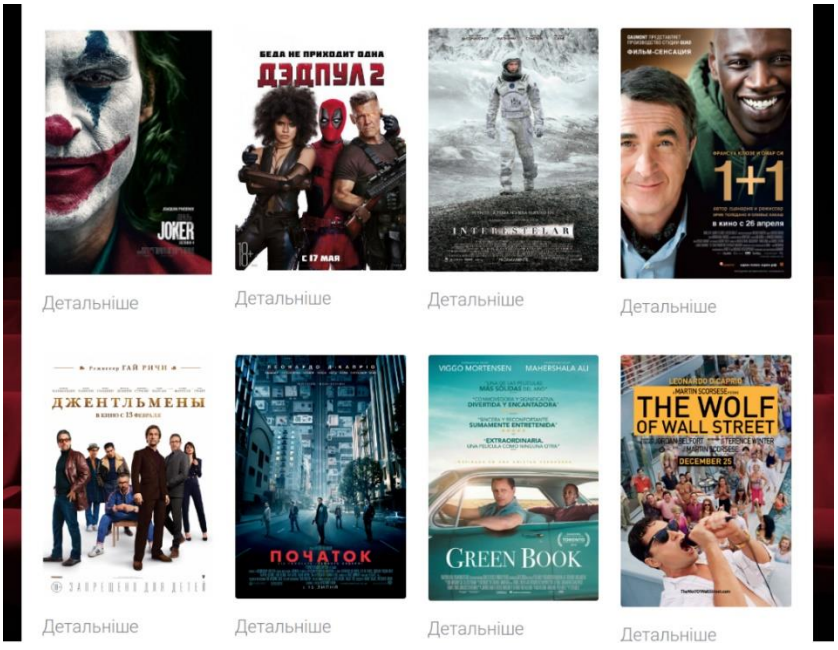

Рис. 2.6. Перегляд списку кінофільмів

Для пошуку кінофільму користувач повинен ввести параметри пошуку у поле форми, що зображена на рис. 2.6

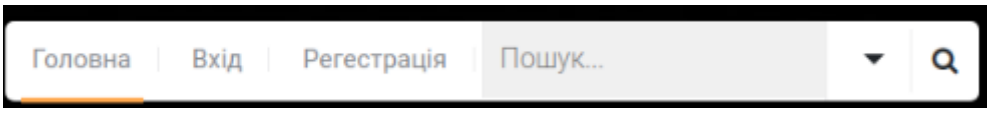

Рис. 2.6. Введення параметрів пошуку

Перелік кінофільмів можна фільтрувати за: жанром, країною, ріком. Для їх застосування треба обрати фільтр з випадаючого списку форми та натиснути на кнопку «Фільтрувати», рис. 2.7.

Рис. 2.7. Фільтрація

Для задання додаткових параметрів пошуку треба встановити бажані парамерти з випадаючого списку та натиснути на кнопку «Пошук з параметрами», рис. 2.8.

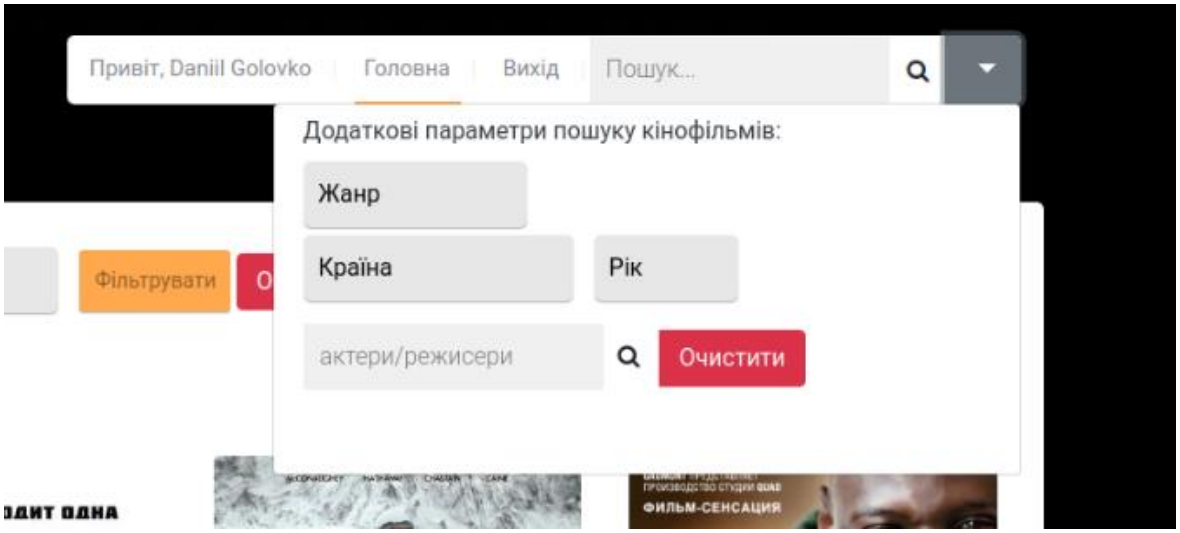

Рис. 2.8. Додаткові параметри пошуку

Якщо знайдені результати не поміщаються на сторінці пошуку, внизу з’являються кнопки для перехода на наступні сторінки, що представлено на рис. 2.9.

Рис. 2.9. Кнопки навігації по сторінках

Якщо в результаті пошуку не було знайдено жодного кінофільму, буде виведено відповідне повідомлення, представлене на рис. 2.10.

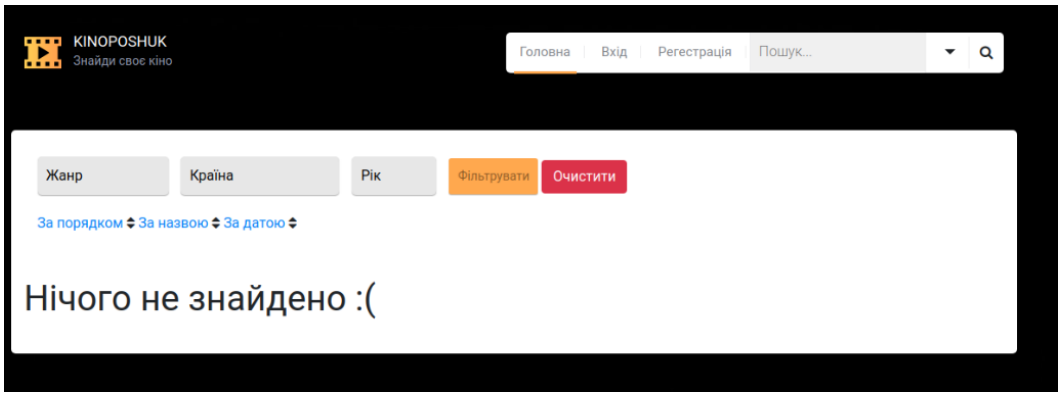

Рис. 2.10. Повідомлення про те, що кінофільми не знайдено

Для того, щоб очистити всі введені параметри фільтрації треба натиснути на кнопку «Очистити», рис. 2.11

Рис. 2.11. Кнопка для очищення результатів пошуку

Перелік кінофільмів можно сортувати за: порядком, іменем, датою. Для того щоб відсортувати за бажаною категорією, потрібно обрати відповідний варіант, рис. 2.12. Приклад сортування переліку кінофільмів представлено на рис. 2.13.

Рис. 2.12. Категорії для сортування

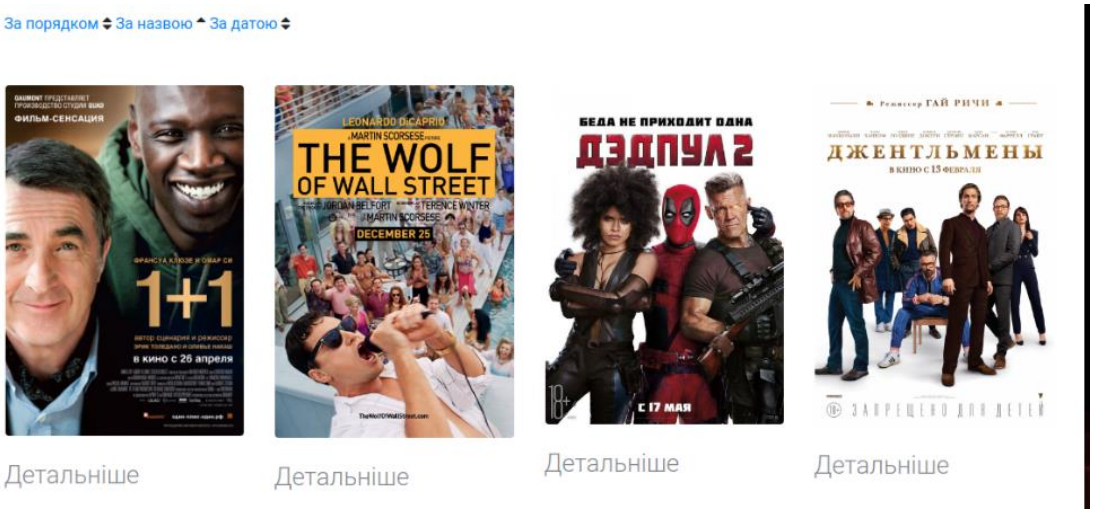

Рис. 2.13. Результати сортування

Для перегляду повної інформації про кінофільм треба натиснути на кнопку «Детальніше» на блоку кінофільма, рис. 2.14.

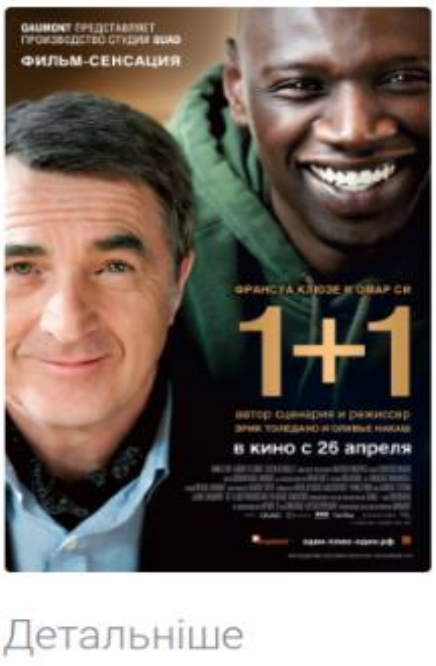

Рис 2.14. Кнопка для переходу на сторінку з повною інформацією

На сторінці кінофільма користувач може переглянути детальну інформацію, щодо характеристик кінофільма, рис. 2.15.

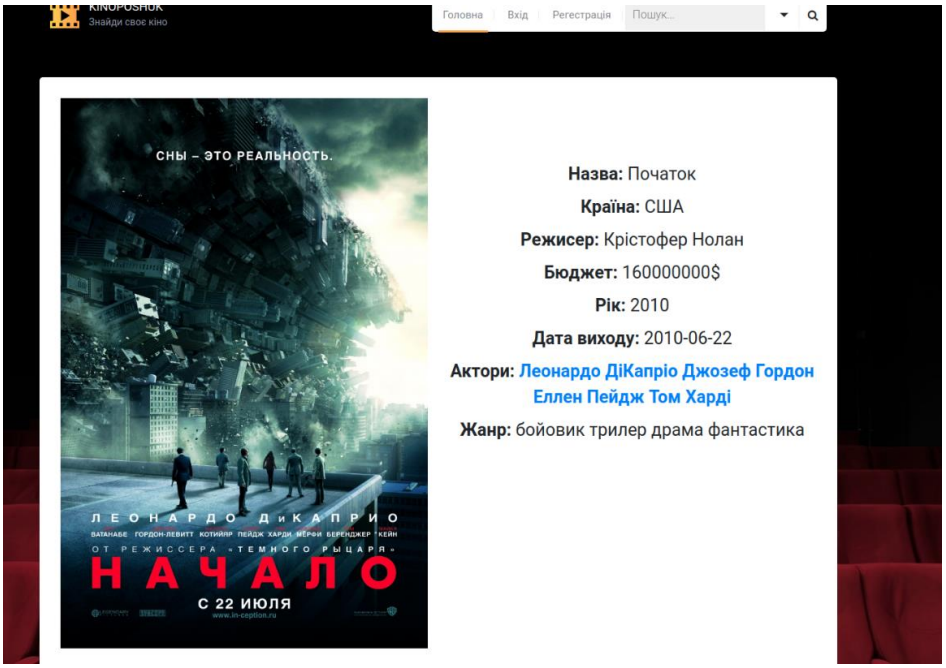

Рис 2.15. Перегляд сторінки з детальною інформацією

На сторінці кінофільма користувач може відправити свою оцінку щодо фільму, але тільки авторизовані користувачі мають на це право, представлено на рис. 2.16.

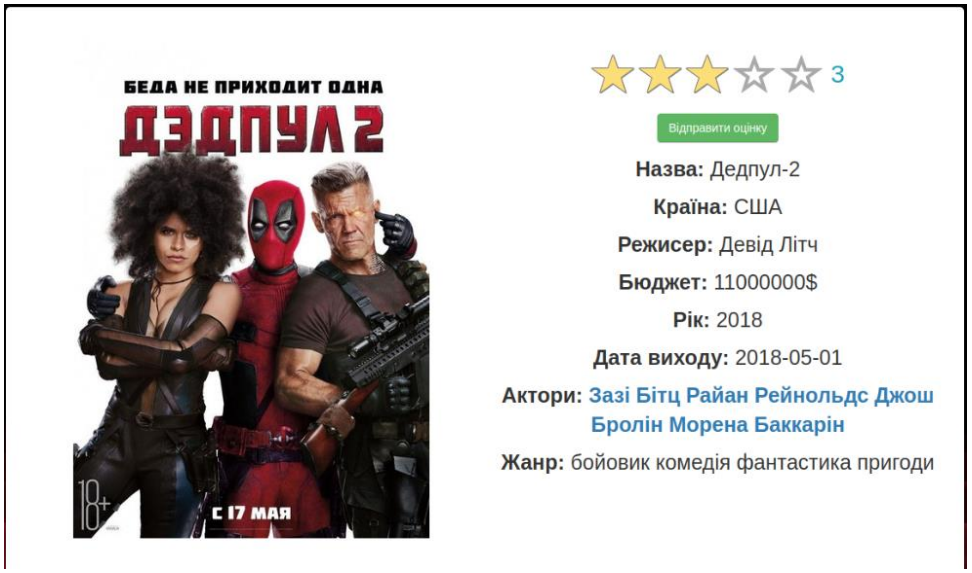

Рис. 2.16. Оцінювання кінофільма

Також, користувач може тільки один раз залишати відгук до одного фільму, представлено на рис. 2.17.

Рис. 2.18. Оцінювання кінофільма

Користувач може залишати коментар щодо фільму, але він повинен бути авторизований, рис. 2.19.

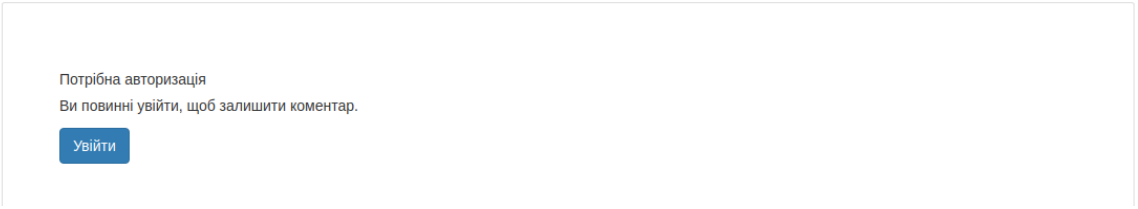

Рис. 2.19. Авторизація для коментарю

Після авторизації, користувач може ввести своє повідомлення у формі для введення, а також продивитися інші коментарі, якщо вони відсутні то виводиться відповідне повідомлення, рис. 2.20

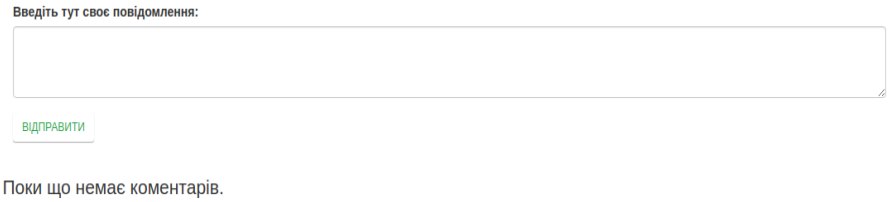

Рис 2.20. Форма для введення повідомлення

Після відправки свого коментаря, користувач може регадувати його або видалити, а інши користувачі відповісти, рис. 2.21.

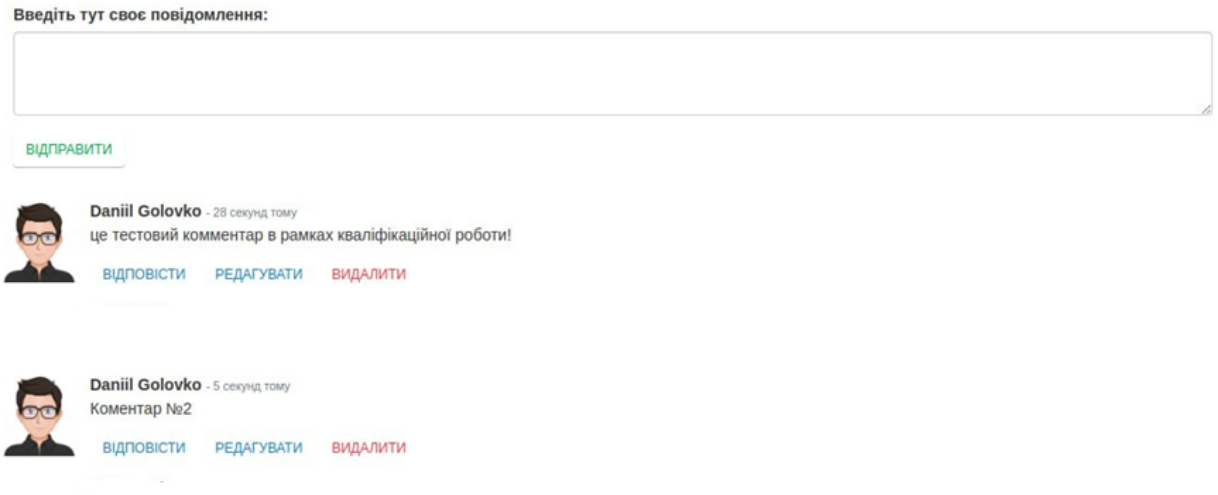

Рис. 2.21. Коментарі

Клацнувши на кнопку «Редагувати», буде відображена наступна форма, рис. 2.22.

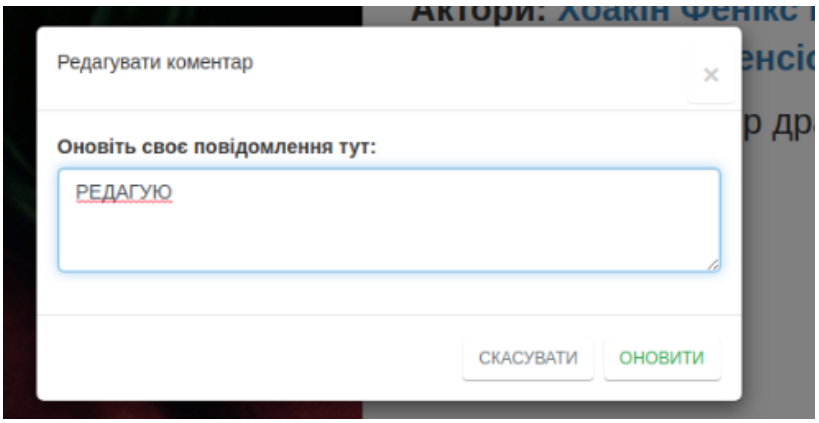

Рис. 2.22. Форма редагування

Клацнувши на кнопку «Відповісти», буде відображена наступна форма, рис. 2.23.

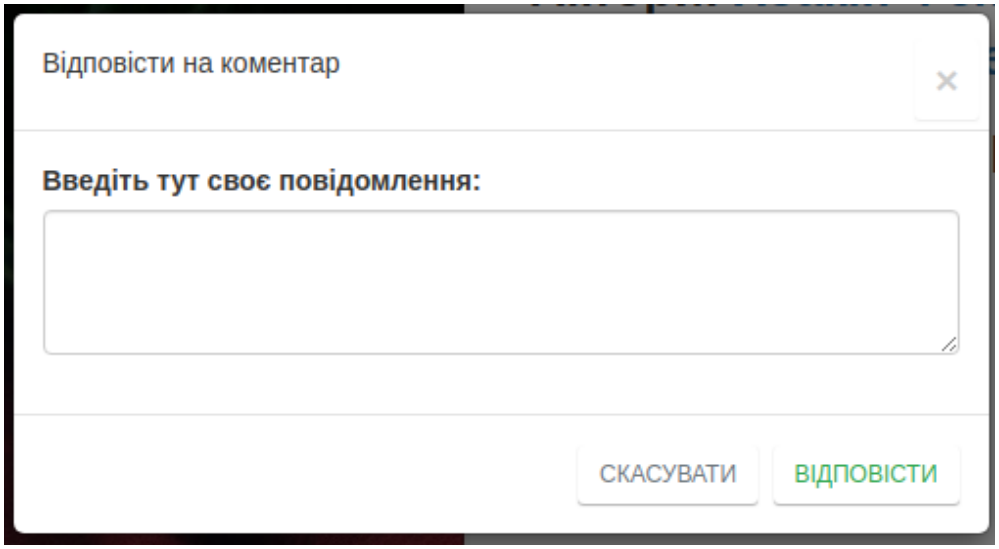

Рис. 2.23. Форма відповіді

Також користувачі можуть надати рейтинг коментарю, рис. 2.24.

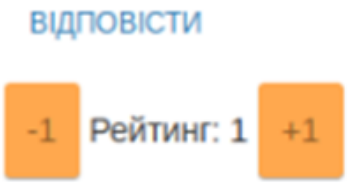

Рис. 2.24. Оцінювання коментаря

На сторінці кінофільма користувач може перейти до детальної інформації про актора, натиснувши на його ім’я, представлено на рис. 2.25.

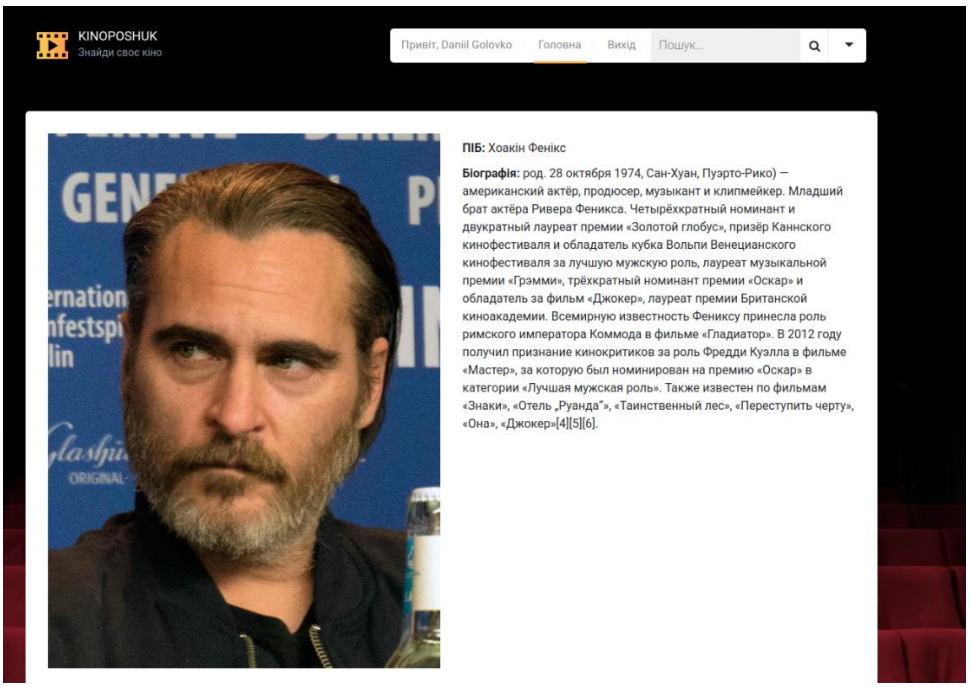

Рис. 2.25. Сторінка актора

Користувач може зареєструватися у системі. Для цього йому потрібно на головному меню клацнути на кнопку «Регестрації», після чого його буде переадресовано на сторінку регестрації. Щоб зарегеструвати свій обліковий запис, користувачу потрібно коректно ввести такі дані, як: нікнейм, e-mail, пароль. Це представлено на рис. 2.26.

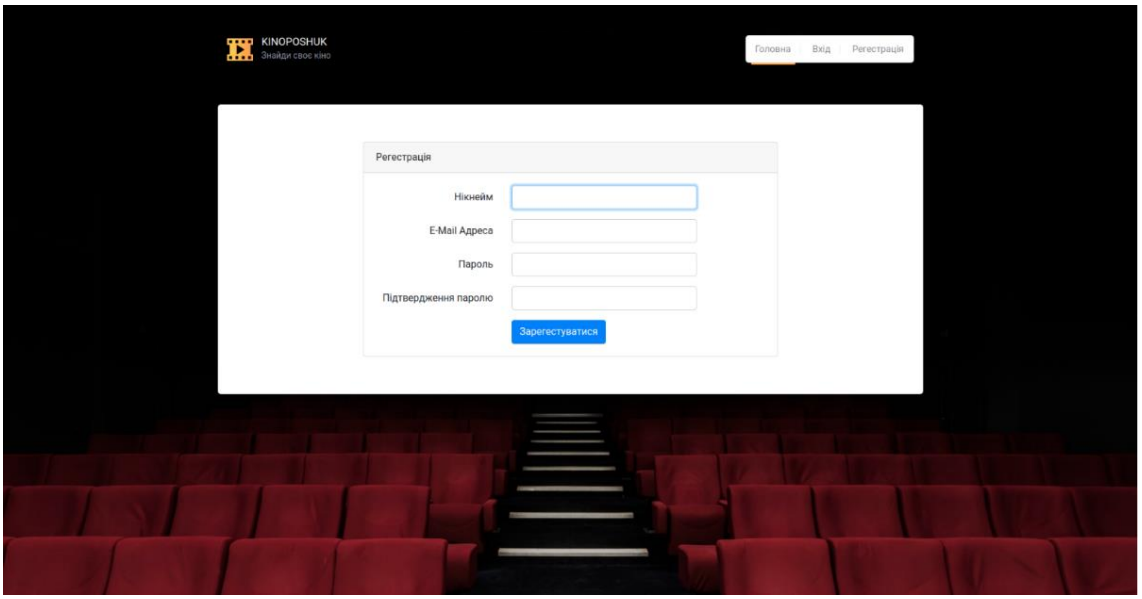

Рис. 2.26. Реєстрація облікового записку

Якщо користувач вже має обліковий запис, він може увійти до нього, клацнувши на кнопку «Вхід» на головному меню, після чого його буде переадресовано на сторінку входу.

Щоб користувач міг зайти в обліковий запис, йому потрібно ввести його e-mail і пароль, який використовувався при реєстрації. Це представлено на рис. 2.27.

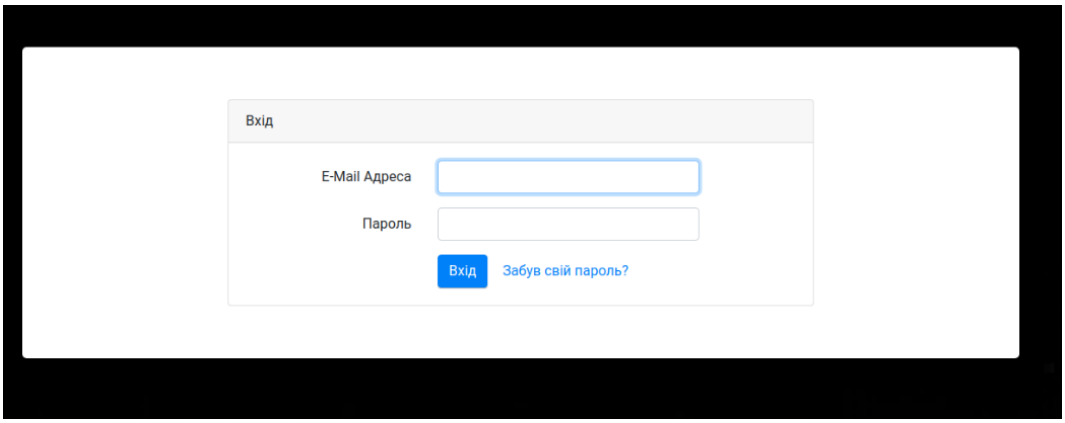

Рис. 2.27. Вхід в обліковий запис

Якщо користувач забув свій пароль, він може натиснути на кнопку «Забув пароль», та перейти на сторінку скидання пароля, рис. 2.28.

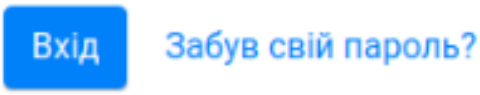

Рис 2.28. Забув пароль

На сторінці скидання паролю користувачу потрібно вказати свій e-mail, куда йому буде відправлено письмо з переходом на сторінку встановлення нового паролю, рисунки 2.29-2.30.

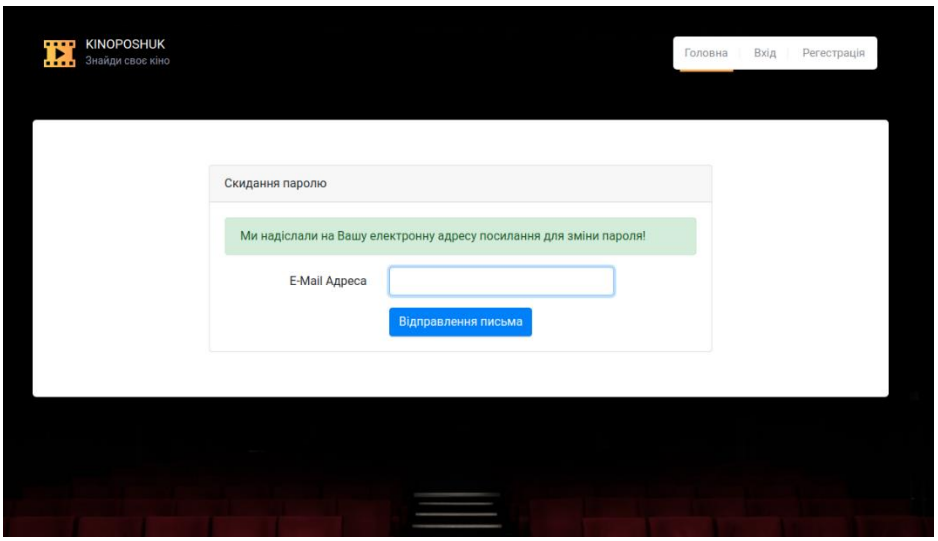

Рис 2.29. Відправлення на e-mail письма зі скиданням паролю

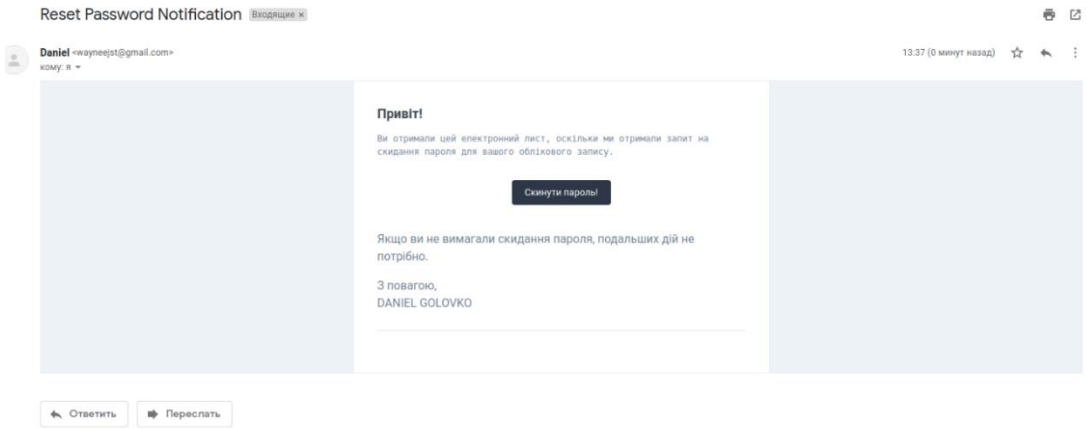

Рис. 2.30. Електроне письмо зі скиданням паролю

На сторінці встановлення нового паролю користувачу потрібно ввести дані про новий пароль, рис. 2.31.

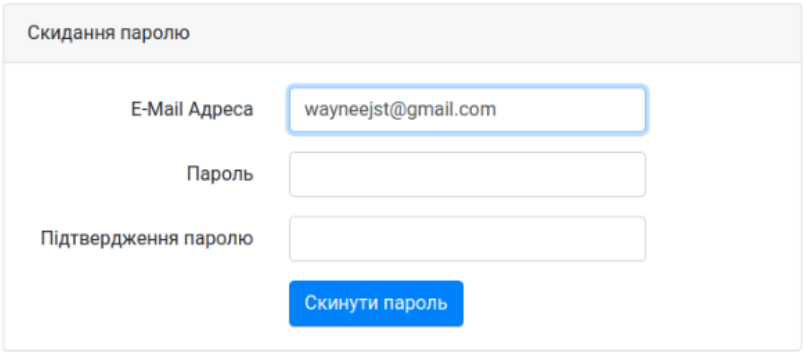

Рис. 2.31. Новий пароль

На сайті присутній чат для авторизованих користувачів. Щоб почати спілкування, треба в головному меню клацнути на вкладку «Чат», потім відкриється сторінка чату.

На сторінці виводиться інформація про онлайн користувачів, імена та тексти повідомлень користувачів. Також можна самому відправити повідомлення, написавши свій текст в певному полі для його введення, це представлено на рис. 2.31.

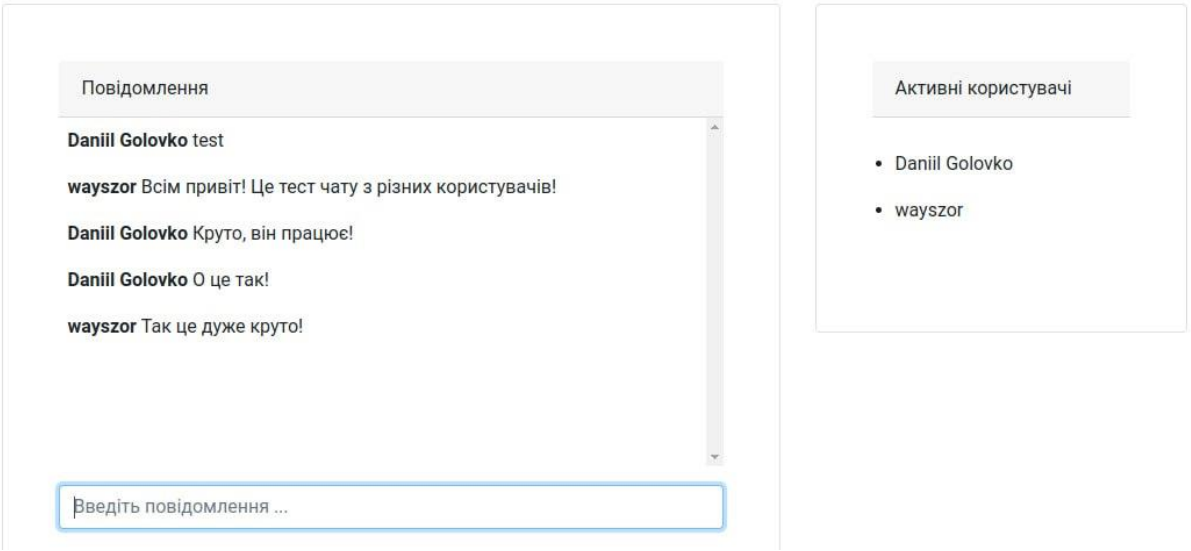

Рис. 2.31. Сторінка чату

Створювати інформацію про кінофильми можут лише авторизовані користувачі з правами адміністратора. Для переходу до адмін-панелі потрібно натиснути на кнопку «ВИ МАЄТЕ ПРАВА АДМІНІСТРАТОРА», рис. 2.32.

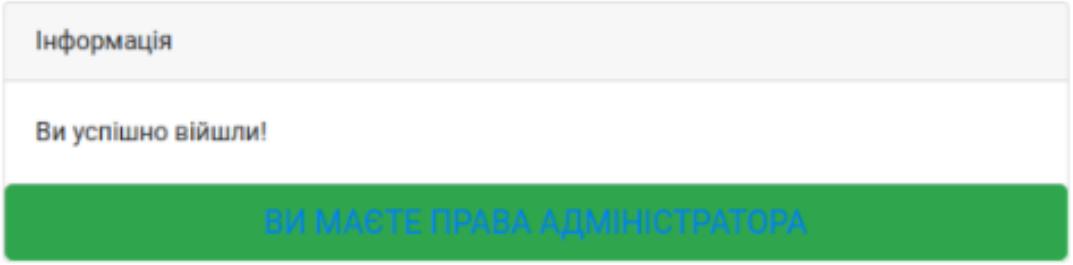

Рис. 2.32. Перехід до адмін-панелі

Буде виведено сторінку з переліком кінофільмів. В нижній частині кожного кінофільма, знаходяться кнопки управління, які виводяться лише для адміністратора. Ці кнопки дозволяють видаляти, редагувати інформацію про кінофільми, тощо, рис. 2.33.

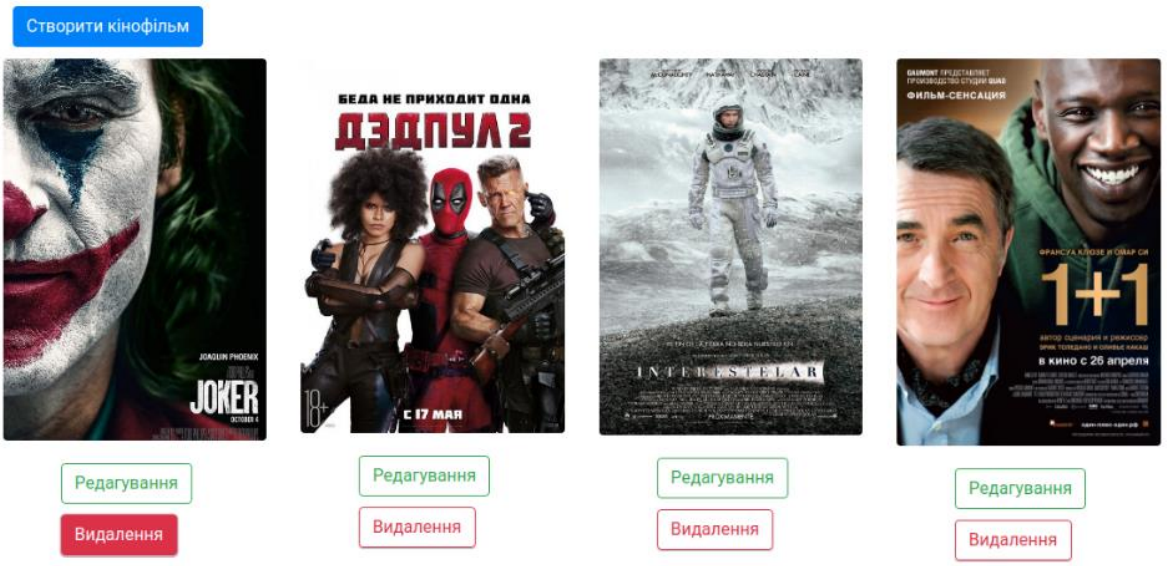

Рис. 2.33. Управління інформацією про кінофільми

Натиснувши на кнопку «Створити кінофільм», буде виведено форму для створення інформації про кінофільми, рис. 2.34. Адміністратору потрібно ввести коректну інформацію щодо назви, бюджету, режисерів, акторів, жанрів, а також додати зображення.

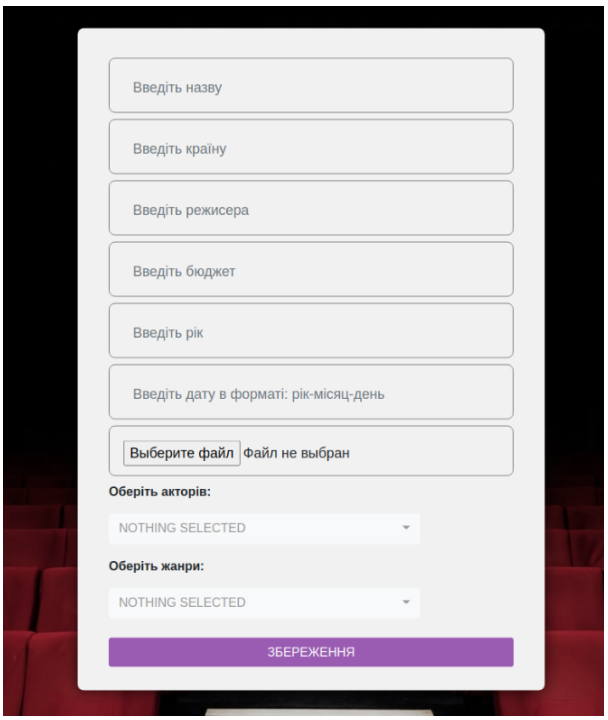

Рис. 2.34. Порожня форма для заповнення інформації про кінофільм

Після заповнення форма буде мати вигляд, який вказано на рис. 2.35. Після натискання на кнопку «Збереження», якщо всі введені дані коректні, адміністратора буде автоматично переправлено на сторінку переліку кінофільмів.

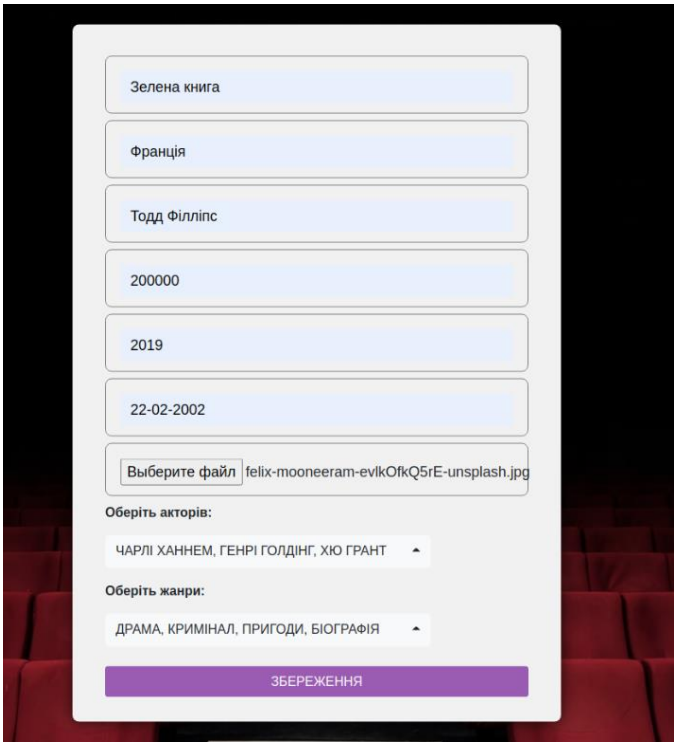

Рис. 2.35. Коректно заповнена форма

Для видалення кінофільму потрібно натиснути на кнопку «Видалити» та підтвердити видалення кінофільму, рис. 2.36.

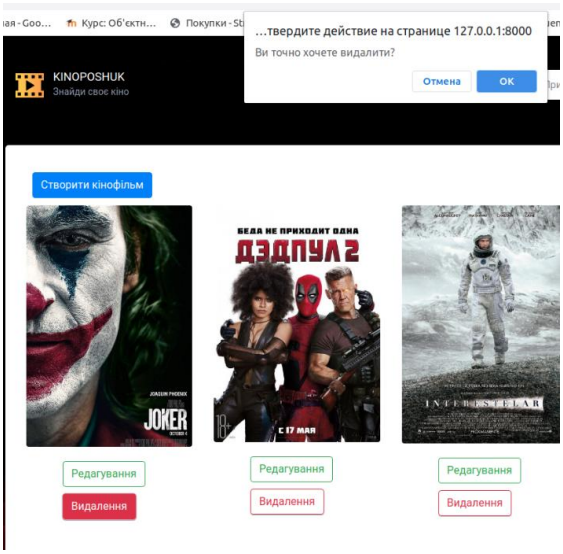

Рис. 2.36. Видалення кінофільму

Для редагування інформації про кінофільм потрібно натиснути на кнопку «Редагувати». Після цього перед адміністратором з’явиться заповнена форма про кінофільм, дані в якій він може змінювати, рис. 2.37.

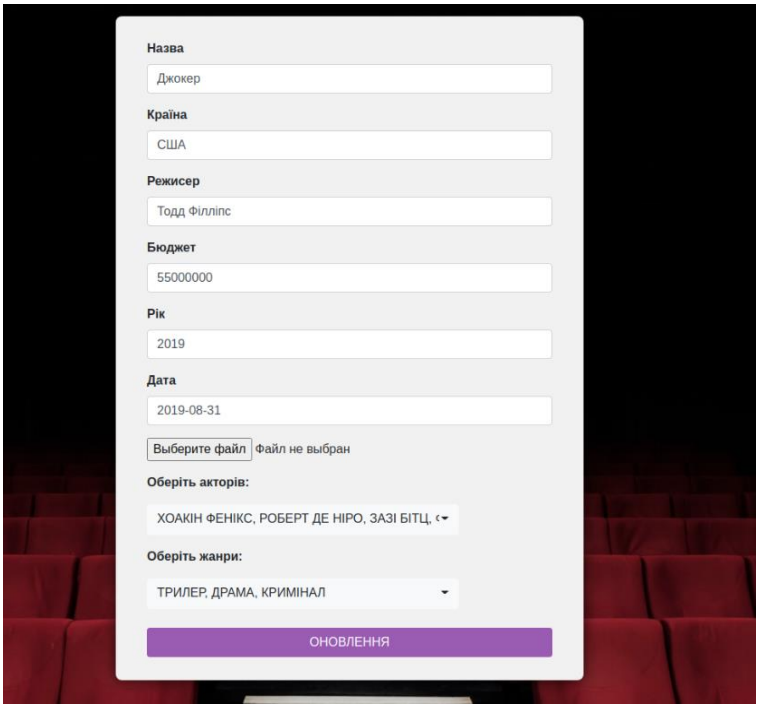

Рис. 2.37. Форма редагування

Натиснувши на кнопку «Інформація», адміністратору буде видано інформацію, щодо нових користувачів додатку, яка представлена як діаграма. рис. 2.38.

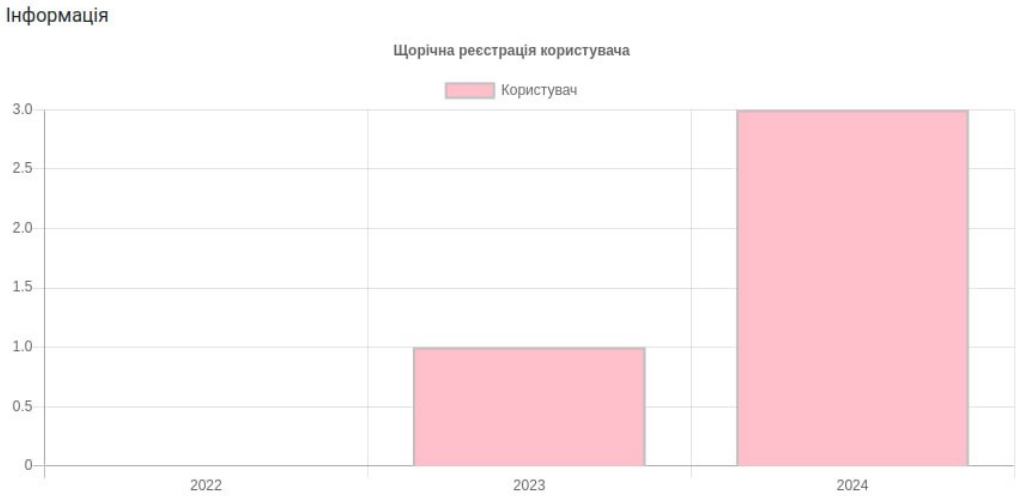

Рис. 2.38. Інформація щодо нових користувачів додатку

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1.Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. Передбачуване число операторів програми - 700;
2. Коефіцієнт складності програми (1,25...2,0) - 1,7;
3. Коефіцієнт корекції програми в ході її розробки (0,05...0,1) - 0,07;
4. Годинна заробітна плата програміста - 131 грн/год;

Згідно зі статистикою платформи найму "Djinni" на середину 2024 року, середня зарплата Junior PHP розробника в Україні варіюється в діапазоні від 300$ до 900$ на місяць. Виходячи з цих даних, можна розрахувати середню місячну заробітну плату, яка становить приблизно 600$ на місяць.

Ураховуючи офіційний курс валют Національного банку України на початок травня 2024 року, де один американський долар еквівалентний 38,50 гривням, середня місячна зарплата Junior PHP розробника в Україні складає 23100 гривень. Якщо припустити стандартний робочий графік з 176 годинами на місяць, то зарплата за годину становитиме приблизно 131 гривню.

5. Коефіцієнт збільшення витрат праці в наслідок недостатнього опису задачі (1,2...1,5) - 1,2;
6. Коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності (від 3-х до 5 років) – 1,1;
7. Вартість машино-години ЕОМ - 14 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{, людино-годин,} \qquad (3.1)$$

де t_o - витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \tag{3.2}$$

де q - передбачуване число операторів програми (q =700);

C - коефіцієнт складності програми (C =1,7);

p - коефіцієнт корекції програми в ході її розробки (p =0,07).

Звідси умовне число операторів в програмі:

$$Q = 700 \cdot 1{,}7 \cdot (1 + 0{,}07) = 1273{,}3.$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}\ \text{, людино-годин,} \tag{3.3}$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 3 до 5 років він складає 1,1.

Приймемо збільшення витрат праці в наслідок недостатнього опису завдання не більше 50% (B = 1.2). З урахуванням коефіцієнта кваліфікації k = 1,1, отримуємо витрати праці на вивчення опису завдання:

$$t_u = (1273{,}3 \cdot 1{,}2) / (80 \cdot 1{,}1) = 17{,}36 \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20\ldots25)\cdot k}\ \text{, людино-годин,} \qquad (3.4)$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.4), отримаємо:

$$t_a = 1273{,}3 / (21 \cdot 1{,}1) = 55{,}12 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20\ldots25)\cdot k}\ \text{, людино-годин,} \qquad (3.5)$$

$$t_n = 1273{,}3 / (25 \cdot 1{,}1) = 46{,}30 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5)\cdot k}\ \text{, людино-годин,} \qquad (3.6)$$

$$t_{отл} = 1273{,}3 / (5 \cdot 1{,}1) = 231{,}51 \text{ людино-годин.}$$

– за умови комплексного налагодження завдання:

$$t^k_{отл} = 1{,}5 \cdot t_{отл}\ \text{, людино-годин,} \qquad (3.7)$$

$$t^k_{отл} = 1{,}5 \cdot 231{,}51 = 347{,}26 \text{ людино-годин.}$$

Витрати праці на підготовку документації визначаються за формулою:

$$t_д = t_{др} + t_{до}\ \text{, людино-годин,} \qquad (3.8)$$

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису:

$$t_{др} = \frac{Q}{(15..20)\cdot k}\ \text{, людино-годин,} \qquad (3.9)$$

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0{,}75 \cdot t_{др}\ \text{, людино-годин.} \qquad (3.10)$$

Підставляючи відповідні значення, отримаємо:

$t_{др}$ = 1273,3 / (18 · 1,1) = 64,31 людино-годин.

$t_{до}$ = 0,75 · 64,31 = 48,23 людино-годин.

$t_{д}$= 64,31 + 48,23 = 112,54 людино-годин.

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

t = 50 + 17,36 + 55,12 + 46,30 + 347,26 + 112,54 = 628,59 людино-годин.

3.2. Рахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрати машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн.} \quad (3.11)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,}$$

де: t - загальна трудомісткість, людино-годин;

$С_{ПР}$ - середня годинна заробітна плата програміста, грн/година

З урахуванням того, що середня годинна зарплата програміста становить 131 грн / год, отримуємо:

$З_{ЗП}$ = 628,59 · 131 = 82345,29 грн.

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot С_{мч}, \text{ грн,} \quad (3.12)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$С_{мч}$ - вартість машино-години ЕОМ, грн/год (14 грн/год).

Підставивши в формулу (3.12) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$З_{мв}$= 231,51 · 14 = 3241,14 грн.

Звідси витрати на створення програмного продукту:

$$K_{ПО} = 82345,29 + 3241,14 = 85586,43 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \quad \text{міс.} \tag{3.13}$$

де B_k - число виконавців (дорівнює 1);

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин).

Звідси витрати на створення програмного продукту:

$$T = 628,59 / 1 \cdot 176 = 3,57 \text{ міс.}$$

Висновок

Програмне забезпечення призначене для реалізації веб-додатку з метою підбору кінофільмів. Вартість даного програмного забезпечення становить 85586,43 грн і включає витрати на заробітну плату виконавця та машинний час, необхідний для налагодження програми. Очікуваний час розробки складає приблизно 3,5 місяця. Цей термін обумовлений значним числом операторів і включає час на дослідження та розробку алгоритму вирішення поставленого завдання, програмування за готовим алгоритмом, налагодження програми і підготовку документації.

ВИСНОВКИ

Мета даної кваліфікаційної роботи полягала у створенні веб-додатку для підбору кінофільмів, що дозволяє користувачам знаходити фільми за різними критеріями та взаємодіяти між собою через систему коментарів та чатів.

Розроблений веб-додаток має наступні функції:

– додавання нових кінофільмів до бази даних;

– редагування існуючих записів про кінофільми;

– видалення кінофільмів з бази даних;

– моніторинг зареєстрованих користувачів;

– пошук фільмів за назвою, іменами акторів та режисерів з можливістю застосування додаткових параметрів пошуку, таких як жанр, країна виробництва та рік випуску;

– сортування інформації за різними параметрами: за назвою, за датою випуску фільму;

– застосування фільтрів, які дозволяють уточнити вибір за жанром, країною чи роком випуску для зручності користувачів;

– перегляд детальної інформації про кінофільми, включаючи сюжет, дату випуску, відгуки і рейтинги;

– можливість залишати оцінки та відгуки;

– ведення бесід у онлайн-чаті на сайті.

Актуальність розробки системи для підбору кінофільмів полягає у все зростаючому інтересі до перегляду кінофільмів у цифровому форматі. Користувачі стикаються зі складністю у пошуку та підборі фільмів за власними вподобаннями. Цей додаток забезпечує зручний інтерфейс та широкі можливості для таких користувачів.

В розробленій системі використовуються передові веб-технології:

– СУБД MySQL для збереження та управління всією інформацією, що стосується кінофільмів, користувачів, відгуків, рейтингів та адміністративних даних.

MySQL обрано через її надійність, швидкість обробки запитів, високий рівень безпеки та підтримку великої кількості паралельних операцій, що є важливим для динамічно змінюваних вмістів [22];

– серверна частина на PHP з використанням фреймворку Laravel для забезпечення модульності та легкості обслуговування;

– клієнтська частина використовує HTML5 і CSS3 для створення адаптивного дизайну, що забезпечує комфортне відображення на різних пристроях;

– JavaScript і бібліотека jQuery для реалізації інтерактивних елементів інтерфейсу

Розробка веб-додатку була проведена з урахуванням всіх сучасних вимог до безпеки та стандартів розробки програмного забезпечення.

В «Економічному розділі» визначено трудомісткість розробки програмного забезпечення (628,59 людино-годин), підраховані витрати на створення програмного забезпечення (85586,43 грн.) і гаданий період розробки (приблизно 3,5 місяці).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. IMDb [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/IMDb

2. IMDb [Електронний ресурс] // Офіційний сайт – Режим доступу до ресурсу: https://www.imdb.com

3. JustWatch [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/JustWatch

4. JustWatch [Електронний ресурс] // Офіційний сайт – Режим доступу до ресурсу: https://www.justwatch.com

5. OWASP [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/OWASP

6. SQL Injection [Електронний ресурс] // OWASP. – Режим доступу до ресурсу: https://owasp.org/www-community/attacks/SQL_Injection

7. Cross-Site Scripting (XSS) [Електронний ресурс] // OWASP. – Режим доступу до ресурсу: https://owasp.org/www-community/attacks/xss/

8. Cross-Site Request Forgery (CSRF) [Електронний ресурс] // OWASP. – Режим доступу до ресурсу: https://owasp.org/www-community/attacks/csrf

9. OWASP Top Ten Security Risks [Електронний ресурс] // OWASP. – Режим доступу до ресурсу: https://owasp.org/www-project-top-ten/

10. MVC (Model-View-Controller) [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Model-view-controller

11. SOLID Principles [Електронний ресурс] // WIкіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/SOLID_(object-oriented_design)

12. DRY Principle [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Don%27t_repeat_yourself

13. KISS Principle [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/KISS_principle

14. Laravel [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Laravel

15. Eloquent ORM [Електронний ресурс] // Офіційна документація Laravel – Режим доступу до ресурсу: https://laravel.com/docs/eloquent

16. PhpStorm [Електронний ресурс] // Офіційний сайт JetBrains – Режим доступу до ресурсу: https://www.jetbrains.com/phpstorm/ (дата звернення: 19.05.2024).

17. PhpStorm Documentation [Електронний ресурс] // JetBrains – Режим доступу до ресурсу: https://www.jetbrains.com/help/phpstorm/

18. HTML [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/HTML

19. CSS [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Cascading_Style_Sheets

20. Bootstrap [Електронний ресурс] // Офіційна документація – Режим доступу до ресурсу: https://getbootstrap.com/docs/4.0/getting-started/introduction/

21. PHP [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/PHP

22. MySQL [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/MySQL

ДОДАТОК А

ER-діаграма бази даних kino представлена на рис. А.1

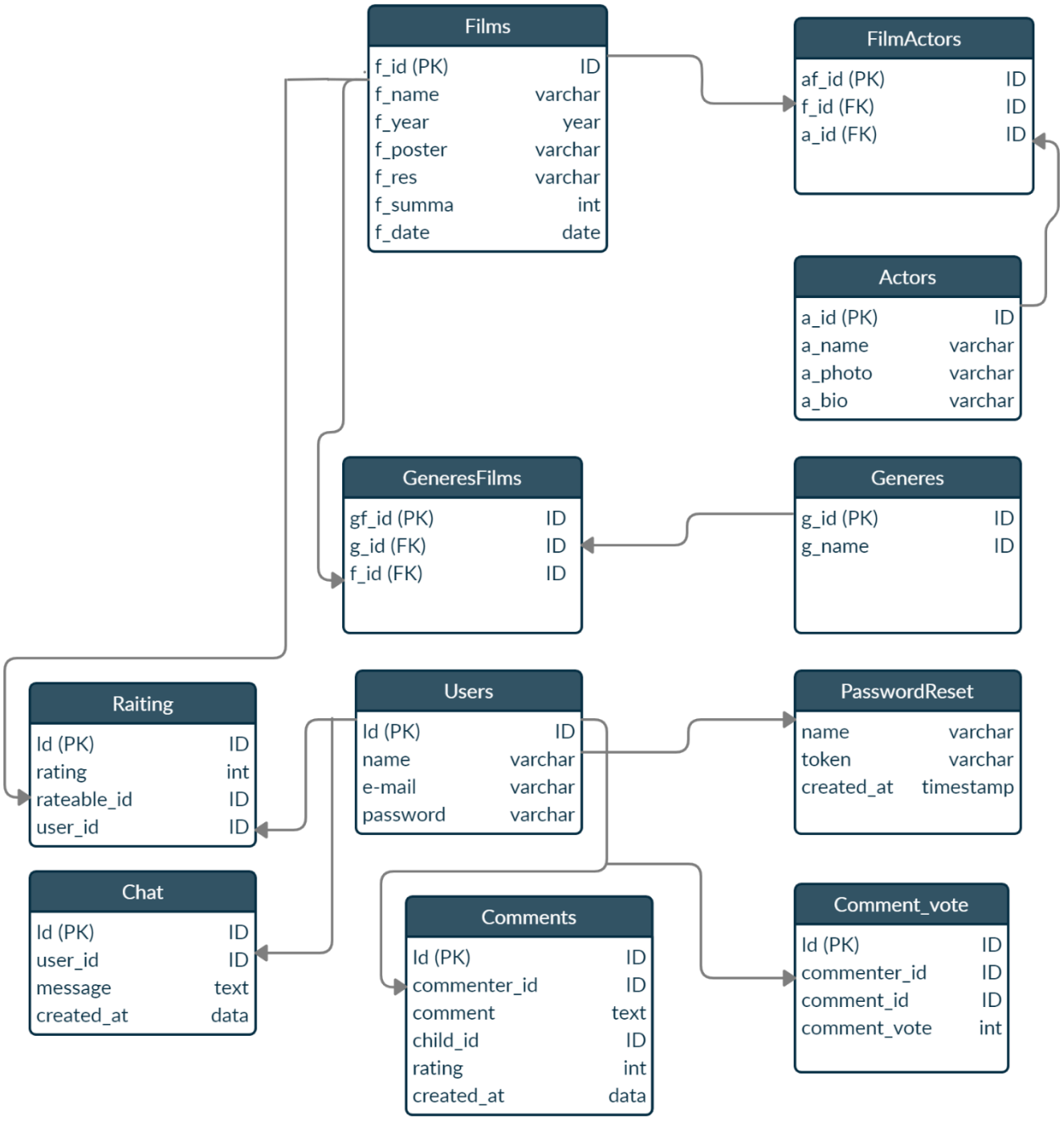

Рис. А.1. ER-діаграма бази даних kino

ДОДАТОК Б

КОД ПРОГРАМИ

Лістинг файлу Actors.php

```
class Actors extends Model
{

    public $timestamps = false;
    protected $primaryKey = 'a_id';
    use HasFactory;
    protected $fillable = [
        'a_name'
    ];

    public function films_models()
    {
        return $this->belongsToMany(FilmsModel::class);
    }
}
class ActorsFilmsModels extends Model
{
    protected $primaryKey = 'af_id';
    use HasFactory;
}
class FilmsModel extends Model implements ICommentable
{
    use Commentable;
    use Sortable;
    use Rateable;
    public $sortable = ['f_id','f_name','f_date', 'f_summa'];
    public $timestamps = false;
    protected $primaryKey = 'f_id';
    use HasFactory;
    protected $fillable = [
        'f_name',
        'f_country',
        'f_res',
        'f_summa',
        'f_year',
        'f_date',
    ];

    public function images()
    {
        return $this->hasOne(Image::class);
    }

    public function actors()
    {
```

```
        return $this->belongsToMany(Actors::class);

    }
public function generes()
    {
        return $this->belongsToMany(Generes::class);
    }
class FilmsModelGeneres extends Model
{
    use HasFactory;
}
class Generes extends Model
{
    public $timestamps = false;
    use HasFactory;
    protected $primaryKey = 'g_id';
    protected $fillable = [
        'g_name'
    ];
    public function films_models()
    {
        return $this->belongsToMany(FilmsModel::class);
    }
}
}
```

Лістинг файлу Users.php

```
class User extends Authenticatable
{

    public $timestamps = false;
    use Commenter;
    use Notifiable;
    use HasFactory;
    protected $fillable = [
        'name',
        'email',
        'password'
    ];

    public function messages(){

        return $this->hasMany(Chat::class);

    }
}
class PasswordReset extends Model
{
    use HasFactory;
}
class Comment extends laraComment
{
```

```
    use HasFactory;
}
class Chat extends Model
{
    use HasFactory;

    protected $guarded = [];

    public function user(){

        return $this->belongsTo(User::class);

    }

}
```

Лістинг файлу MainController.php

```
class MainController extends Controller
{


    private function getSelectedId(Request $req)
    {
        $selected_id = [];
        $selected_id['generes_id'] = $req->generes_id;
        $selected_id['country_id'] = $req->country_id;
        $selected_id['year_id'] = $req->year_id;
        return $selected_id;
    }

    public function review(Request $req)
    {
        $selected_id = $this->getSelectedId($req);
        $info = FilmsModel::with('images')->sortable()->paginate(8);
        return view('review', ['info' => $info, 'selected_id' =>
$selected_id]);
    }

    public function single($f_id, request $req)
    {
        $selected_id = $this->getSelectedId($req);
        $info = new FilmsModel();
        return view('single', ['info' => FilmsModel::query()-
>find($f_id), 'selected_id' => $selected_id]);
    }



    public function actorsOne($a_id, request $req)
    {
        $selected_id = $this->getSelectedId($req);
        $info = new Actors();
        return view('actors', ['info' => Actors::query()-
```

```
>find($a_id), 'selected_id' => $selected_id]);
    }

    public function filter(Request $req)
    {
        $info = FilmsModel::whereHas('generes', function ($query) use
($req) {
            return $req->generes_id ?
                $query->where('g_id', $req->generes_id) : '';
        })->where(function ($query) use ($req) {
            return $req->country_id ?
                $query->from('films_models')->where('f_country',
$req->country_id) : '';
        })->where(function ($query) use ($req) {
            return $req->year_id ?
                $query->from('films_models')->where('f_year', $req-
>year_id) : '';
        })
            ->with('generes')
            ->sortable()
            ->paginate(8);

        $selected_id = $this->getSelectedId($req);

        return view('review', compact('info', 'selected_id'));

    }


    public function searchname(Request $req)
    {

        $s = $req->s;
        $info = FilmsModel::where('f_name', 'LIKE', "%{$s}%")
            ->sortable()
            ->paginate(8);


        $selected_id = $this->getSelectedId($req);
        return view('review', ['info' => $info, 'selected_id' =>
$selected_id]);
    }


    public function searchFilter(Request $req)
    {

        $s = $req->s;
        $info = FilmsModel::WhereHas('actors', function ($q) use ($s)
{
            $q->where('a_name', 'like', '%' . $s . '%')-
>orWhere('f_res', 'LIKE', "%{$s}%");
        })->whereHas('generes', function ($query) use ($req) {
            return $req->generes_id ?
```

```
                $query->where('g_id', $req->generes_id) : '';
        })->where(function ($query) use ($req) {
            return $req->country_id ?
                $query->from('films_models')->where('f_country',
$req->country_id) : '';
        })->where(function ($query) use ($req) {
            return $req->year_id ?
                $query->from('films_models')->where('f_year', $req-
>year_id) : '';
        })
            ->with('generes')
            ->sortable()
            ->paginate(8);


        $selected_id = $this->getSelectedId($req);
        return view('review', ['info' => $info, 'selected_id' =>
$selected_id]);
    }

    public function destroy()
    {
        auth()->logout();
        return redirect()->to('/');
    }

}
public function index()
    {


        $info = FilmsModel::with('actors','generes')->paginate(8);

        return view('info.index', compact('info'));
    }


    public function create()
    {

        return
view('info.create',['a_name'=>Actors::all(),'g_name'=>Generes::all()]
);

    }



    use ImageUpload; //Using our created Trait to access inside trait
method

    public function store(Request $request)
    {
        $this->validate($request,[
```

```
            'image'        =>
'required|image|mimes:jpeg,png,jpg,gif|max:2048',
            'f_name' => 'required|regex:/^[\pL\s\0-
9]+$/u|min:3|max:20|unique:films_models',
            'f_country' => 'required|alpha|min:3|max:20',
            'f_res' => 'required|regex:/^[\pL\s\-]+$/u|min:5|max:20',
            'f_summa' => 'required|numeric|min:1|max:1000000000',
            'f_year' => 'required|date_format:Y',
            'f_date' => 'date_format:Y-m-d|required',
            'a_name'=>'required',
            'g_name'=>'required'
        ]);


        $info = FilmsModel::create([
            'f_name' => $request->get('f_name'),
            'f_country' => $request->get('f_country'),
            'f_res' => $request->get('f_res'),
            'f_summa' => $request->get('f_summa'),
            'f_year' => $request->get('f_year'),
            'f_date' => $request->get('f_date'),

        ]);


        $data = new Image;

        $data->i_image = $request->image;
        if($data->i_image){
            try {
                $filePath = $this->UserImageUpload($data->i_image);
//Passing $data->image as parameter to our created method
                $data->i_image = $filePath;

                $info->images()->save($data);


            } catch (Exception $e) {
                //Write your error message here
            }
        }

        foreach ($request->a_name as $tag) {   $info -> actors() ->
attach($tag);}
        foreach ($request->g_name as $tag) {   $info -> generes() ->
attach($tag);}


      $info->save();

        return redirect('/info')->with('success', 'info saved!');
    }
```

```
    public function show($f_id)
    {
        //
    }


    public function edit($f_id)
    {

        $info = FilmsModel::with('actors','generes')->find($f_id);

        return view('info.edit',
['info'=>$info,'a_name'=>Actors::all(), 'g_name'=>Generes::all()]);
    }


    public function update(Request $request, FilmsModel $info, Image
$data)
    {
        $request->validate([
            'image'        =>
'image|mimes:jpeg,png,jpg,gif|max:2048',
            'f_name' => 'required|regex:/^[\pL\s\0-
9]+$/u|min:3|max:20',
            'f_country' => 'required|alpha|min:3|max:20',
            'f_res' => 'required|regex:/^[\pL\s\-]+$/u|min:5|max:20',
            'f_summa' => 'required|numeric|min:1||max:1000000000',
            'f_year' => 'required|date_format:Y',
            'f_date' => 'date_format:Y-m-d|required',
            'a_name'=>'required',
            'g_name'=>'required'
        ]);
        $info->actors()->detach();
        $info->generes()->detach();
        $info->update([
            'f_name' => $request->f_name,
            'f_country' => $request->f_country,
            'f_res'=> $request->f_res,
            'f_summa'=> $request->f_summa,
            'f_year'=> $request->f_year,
            'f_date'=> $request->f_date,
        ]);

        $data->i_image = $request->image;
        if($data->i_image){
            try {
                $filePath = $this->UserImageUpload($data->i_image);
//Passing $data->image as parameter to our created method
                $data->i_image = $filePath;

                $info->images()->update(['i_image'=>$data->i_image]);


            } catch (Exception $e) {
```

```
                //Write your error message here
            }
        }

      foreach ($request->a_name as $tag) {   $info -> actors() ->
attach($tag);}
        foreach ($request->g_name as $tag) {   $info -> generes() ->
attach($tag);}


        return redirect('/info')->with('success', 'info updated!');


    }


    public function destroy($f_id)
    {

        $info = FilmsModel::find($f_id);
        $info->generes()->detach();
        $info->actors()->detach();
        $info->images()->delete();
        $info->delete();


        return redirect('/info')->with('success', 'info deleted!');
    }



}
```

Лістинг файлу HomeController.php

```
class HomeController extends Controller
{

    public function __construct()
    {
        $this->middleware('auth');
    }


    public function index()
    {

        return view('home');
    }
```

```
  private function getSelectedId(Request $req)
  {
      $selected_id = [];
      $selected_id['generes_id'] = $req->generes_id;
      $selected_id['country_id'] = $req->country_id;
      $selected_id['year_id'] = $req->year_id;
      return $selected_id;
  }

    public function posts(Request $req)
    {
        $selected_id = $this->getSelectedId($req);
        $info = FilmsModel::with('images')->sortable()->paginate(8);
        return view('review', ['info' => $info, 'selected_id' =>
$selected_id]);
    }

    public function postPost(Request $request,$f_id)
    {

        $info = FilmsModel::find($f_id);


        $rating = $info->ratings()->where('user_id', auth()->user()-
>id)->first();


        if(is_null($rating)){
            $rating = new Rating();
            $rating->rating =  $request['rate'];
            $rating->user_id = auth()->user()->id;
            $info->ratings()->save($rating);
            return redirect()->back();
        }

        else{
            return redirect()->back()->with('success', 'Вы уже
проголосовали! Дважды - нельзя.');
        }
    }

}
```

Лістинг файлу ChatController.php

```
class ChatController extends Controller
{
    public function __construct()
    {
        $this->middleware('auth');
    }
```

```
 public function index()
    {
        return view('chat');
    }

    public function fetchAllMessages()
    {
        return Chat::with('user')->get();
    }

    public function sendMessage(Request $request)
    {
        $chat = auth()->user()->messages()->create([
            'message' => $request->message
        ]);

        broadcast(new ChatEvent($chat->load('user')))->toOthers();

        return ['status' => 'success'];
    }
}

class ChatEvent implements ShouldBroadcast
{
    use Dispatchable, InteractsWithSockets, SerializesModels;

    public $chat;

    public function __construct(Chat $chat)
    {
        $this->chat = $chat;
    }

    public function broadcastOn()
    {
        return new PresenceChannel('chat');
    }
}
```

Лістинг файлу ChartJsController.php

```
class ChartJsController extends Controller
{
    public function index()
    {
        $year = ['2015','2016','2017','2018','2019','2020'];

        $user = [];
        foreach ($year as $key => $value) {
            $user[] = User::where(DB::raw("DATE_FORMAT(created_at,
'%Y')"),$value)->count();
        }
```

```
return view('charts')-
>with('year',json_encode($year,JSON_NUMERIC_CHECK))-
>with('user',json_encode($user,JSON_NUMERIC_CHECK));
    }
}
```

Лістинг файлу ConfirmPasswordController.php

```
namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Providers\RouteServiceProvider;
use Illuminate\Foundation\Auth\ConfirmsPasswords;

class ConfirmPasswordController extends Controller
{


    use ConfirmsPasswords;

    protected $redirectTo = RouteServiceProvider::HOME;

    public function __construct()
    {
        $this->middleware('auth');
    }
}
class ResetPasswordController extends Controller
{

    use ResetsPasswords;

    /**
     * Where to redirect users after resetting their password.
     *
     * @var string
     */
    protected $redirectTo = RouteServiceProvider::HOME;
}
class RegisterController extends Controller
{
    */

    use RegistersUsers;

    public function __construct()
    {
        $this->middleware('guest');
    }

    /**
     * Get a validator for an incoming registration request.
     *
     * @param  array  $data
     * @return \Illuminate\Contracts\Validation\Validator
```

```
     */
    protected function validator(array $data)
    {
        return Validator::make($data, [
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'email', 'max:255',
'unique:users'],
            'password' => ['required', 'string', 'min:8',
'confirmed'],
        ]);
    }

    /**
     * Create a new user instance after a valid registration.
     *
     * @param  array  $data
     * @return \App\Models\User
     */
    protected function create(array $data)
    {
        return User::create([
            'name' => $data['name'],
            'email' => $data['email'],
            'password' => Hash::make($data['password']),
        ]);
    }

}
class LoginController extends Controller
{

    use AuthenticatesUsers;
    protected $redirectTo = RouteServiceProvider::HOME;
    public function __construct()
    {
        $this->middleware('guest')->except('logout');
    }
}
class ForgotPasswordController extends Controller
{

    use SendsPasswordResetEmails;
}
```

Лістинг файлу Traits.php

```
<?php
namespace App\Traits;

use Illuminate\Http\UploadedFile;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Str;
trait ImageUpload
```

```
{
public function UserImageUpload($query) // Taking input image as
parameter
{
$image_name = Str::random(12);
$ext = strtolower($query->getClientOriginalExtension()); // You can
use also getClientOriginalName()
$image_full_name = $image_name.'.'.$ext;
$upload_path = 'image/';    //Creating Sub directory in Public folder
to put image
$image_url = $upload_path.$image_full_name;
$success = $query->move($upload_path,$image_full_name);

return $image_url; // Just return image
}
}
```

Лістинг файлу CommentsController.php

```
<?php

namespace tizis\laraComments\Http\Controllers;

use Auth;

class CommentsController extends Controller
{
use AuthorizesRequests;

protected $commentService;
protected $voteService;
protected $policyPrefix;

/**
* CommentsController constructor.
* @param VoteService $voteService
*/
public function __construct(VoteService $voteService)
{
$this->middleware(['web', 'auth'], ['except' => ['get']]);
$this->policyPrefix = config('comments.policy_prefix');
$this->voteService = $voteService;
}

/**
* Creates a new comment for given model.
* @param SaveRequest $request
* @return array|\Illuminate\Http\RedirectResponse
* @throws \Illuminate\Auth\Access\AuthorizationException
*/
public function store(SaveRequest $request)
{
$this->authorize($this->policyPrefix . '.store');
```

```
try {
$decryptedModelData = decrypt($request->commentable_encrypted_key);

$commentableId = $decryptedModelData['id'];
$modelPath = $decryptedModelData['type'];

} catch (DecryptException $e) {
throw new \DomainException('Decryption error');
}

if (!CommentService::modelIsExists($modelPath)) {
throw new \DomainException('Model don\'t exists');
}

if (!CommentService::isCommentable(new $modelPath)) {
throw new \DomainException('Model is\'t commentable');
}

$model = $modelPath::findOrFail($commentableId);

$comment = CommentService::createComment(
new Comment(),
Auth::user(),
$model,
CommentService::htmlFilter($request->message)
);

return $request->ajax()
? [
'success' => true,
'comment' => new CommentResource($comment)
]
: redirect()->to(url()->previous() . '#comment-' . $comment->id);
}

/**
* @param GetRequest $request
* @return array
*/
public function get(GetRequest $request): array
{
$decryptedModelData = decrypt($request->commentable_encrypted_key);

$modelId = $decryptedModelData['id'];
$modelPath = $decryptedModelData['type'];

$orderBy = CommentService::orderByRequestAdapter($request);

if (!CommentService::modelIsExists($modelPath)) {
throw new \DomainException('Model don\'t exists');
}

if (!CommentService::isCommentable(new $modelPath)) {
throw new \DomainException('Model is\'t commentable');
```

```
}

$model = $modelPath::where('id', $modelId)->first();

return [
'success' => true,
'comments' => CommentResource::collection(
$model->commentsWithChildrenAndCommenter()
->parentless()
->orderBy($orderBy['column'], $orderBy['direction'])
->get()
),
'count' => $model->commentsWithChildrenAndCommenter()->count()
];
}

/**
* @param Comment $comment
* @param Request $request
* @return array
*/
public function show(Comment $comment, Request $request): array
{
return [
'comment' => $request->input('raw') ? $comment : new
CommentResource($comment)
];
}

/**
* Updates the message of the comment.
* @param EditRequest $request
* @param Comment $comment
* @return array|\Illuminate\Http\RedirectResponse
* @throws \Illuminate\Auth\Access\AuthorizationException
*/
public function update(EditRequest $request, Comment $comment)
{
$this->authorize($this->policyPrefix . '.edit', $comment);

CommentService::updateComment(
$comment,
CommentService::htmlFilter($request->message)
);

return $request->ajax()
? ['success' => true, 'comment' => new CommentResource($comment)]
: redirect()->to(url()->previous() . '#comment-' . $comment->id);
}


array|\Illuminate\Http\RedirectResponse
* @throws \Illuminate\Auth\Access\AuthorizationException
*/
```

```
public function destroy(Request $request, Comment $comment)
{
$this->authorize($this->policyPrefix . '.delete', $comment);

try {
CommentService::deleteComment($comment);
$response = response(['message' => 'success']);
} catch (\DomainException $e) {
$response = response(['message' => $e->getMessage()], 401);
}

if ($request->ajax()) {
return $response;
}

return redirect()->back();
}


public function reply(ReplyRequest $request, Comment $comment)
{
$this->authorize($this->policyPrefix . '.reply', $comment);

$reply = CommentService::createComment(
new Comment(),
Auth::user(),
$comment->commentable,
CommentService::htmlFilter($request->message),
$comment
);

return $request->ajax()
? ['success' => true, 'comment' => new CommentResource($reply)]
: redirect()->to(url()->previous() . '#comment-' . $reply->id);
}}
```

Лістинг файлу blade

```
@extends('layout')
@section('main_content')
<div class="filters">
<form action="{{ route('filter') }}" method="GET" >
<select name="generes_id" id="input">
<option value="0">Жанр  </option>
@foreach (\App\Models\Generes::select ('g_id','g_name')-
>orderby('g_name','asc')->get() as $generes)
<option value="{{ $generes->g_id }}" {{ $generes->g_id ==
$selected_id['generes_id'] ? 'selected' : '' }}>
{{ $generes['g_name'] }}
</option>
@endforeach
</select>
<select name="country_id" id="input">
<option value="0">Країна</option>
@foreach (\App\Models\FilmsModel::select('f_country')-
```

```
>orderby('f_country','asc')->distinct()->get() as $fcountry)
<option value="{{ $fcountry->f_country }}" {{ $fcountry->f_country ==
$selected_id['country_id'] ? 'selected' : '' }}>
{{ $fcountry['f_country'] }}
</option>
@endforeach
</select>
<select name="year_id" id="input">
<option value="0">Рік </option>
@foreach (\App\Models\FilmsModel::select('f_year')-
>orderby('f_year','desc')->distinct()->get() as $fyear)
<option value="{{ $fyear->f_year }}" {{ $fyear->f_year ==
$selected_id['year_id'] ? 'selected' : '' }}>
{{ $fyear['f_year'] }}
</option>
@endforeach
</select>
<input type="submit" class="btn-sm" value="Фільтрувати">
<a href="/" class="btn btn-danger">Очистити</a>
</form>
<div style="margin-top: 20px;" class="form-group">

@sortablelink('f_id','За порядком')
@sortablelink('f_name','За назвою')
@sortablelink('f_date','За датою')
</div>

</div>

<div class="movie-list">

@forelse($info as $el)

<div class="movie">

<figure class="movie-poster"><img src="/{{$el->images->i_image}}"
></figure>

<div class="movie-title   "><a href="{{route('single-one',$el-
>f_id)}}">Детальніше</a></div>

</div>
@empty
<h1> Нічого не знайдено :( </h1>
@endforelse

</div> <!-- .movie-list -->
<div style="margin-left: 500px;" >
{{$info->appends(\Request::except('page'))-
>render('vendor.pagination.bootstrap-4') }}

</div>
</main>
```

```
@endsection

@extends('layout')
@extends('layout')

@section('main_content')
<link
href="https://netdna.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.c
ss" rel="stylesheet">

<div class="row text-center">
<div class="col-md-6">
<figure class="movie-poster"><img src="/{{ $info->images->i_image }}"
alt="#"></figure>
</div>

<div  class="col-md-6">

<ul style="margin-top: 20px;" class="movie-meta movie-title ">


<form action="{{ route('posts.post',$info->f_id)}}" method="POST">
{{ csrf_field() }}

<div class="rating">

            <a> Кількість оцінок:  <strong>{{$info->usersRated()}}
</strong></a>

<input id="input-1" name="rate" class="rating rating-loading" data-
min="0" data-max="5" data-step="1" value="{{ $info->AverageRating }}"
data-size="xs" data-show-clear="false">

                                                                <input
type="hidden" name="id" required="" value="{{ $info->id }}">

<button class="btn btn-success">Відправити оцінку</button>

</div>

</for <li><strong>Назва:</strong> {{$info->f_name}} </li>

<li><strong>Країна:</strong> {{$info->f_country}}</li>

<li><strong>Режисер:</strong> {{$info->f_res}}</li>

<li><strong>Бюджет:</strong> {{$info->f_summa}}$</li>
                                        <li><strong>Рік:</strong>
{{$info->f_year}}</li>
                                        <li><strong>Дата
виходу:</strong> {{$info->f_date}}</li>

<li><strong>Актори: @foreach ($info->actors as $item) <a
href="{{route('actors-one',$item->a_id)}}"> {{ $item->a_name  }}
```

```
</a>  @endforeach</strong> </li>

<li><strong>Жанр: </strong> @foreach ($info->generes as $item)
{{$item->g_name}}   @endforeach </li>
                                                                    <div
class="col-sm-12">

@if(session()->get('success'))

<div class="alert alert-success">

                    {{ session()->get('success') }}
                    </div>
@endif
                    </div>
                        </ul>

                    </div>
</div>

<script type="text/javascript">

$('#input-1').rating({
step: 1,
starCaptions: {1: '1', 2: '2', 3: '3', 4: '4', 5: '5'},

starCaptionClasses: {1: 'text-danger', 2: 'text-warning', 3: 'text-
info', 4: 'text-primary', 5: 'text-success'}
});
</script>

<x-comments :model="$info"/>

@endsection
@extends('base')
@section('main')
<div class="container">
<div class="card-header">{{ __('Інформація:') }}  <strong
style="color:green"> {{ __('Ви успішно війшли! ')   }} </strong>
</div>
<div class="card-body">
<div class="main-body">
<div class="row justify-content-center">
<div class="col-md-8">
<div class="card mb-3">
<div class="d-flex flex-column align-items-center text-center">
<img src="https://bootdey.com/img/Content/avatar/avatar7.png"
alt="Admin" class="rounded-circle" width="150">
<div class="mt-3">
</div>
</div>
<div class="card-body">

<div class="row">
```

```
<div class="col-sm-3">
<h6 class="mb-0">Нікнейм:</h6>
</div>
<div class="col-sm-9 text-secondary">
<strong>   {{ auth()->user()->name }}</strong>
</div>
</div>
<hr>
<div class="row">
<div class="col-sm-3">
<h6 class="mb-0">E-mail: </h6>
</div>
<div class="col-sm-9 text-secondary">
<strong>{{ auth()->user()->email }} </strong>
</div>
</div>
</div>
@can('isAdmin')
<div class="btn btn-success btn-lg" >
<a href="info">ВИ МАЄТЕ ПРАВА АДМІНІСТРАТОРА</a>
</div>
@endcan
</div>
</div>
</div>
</div>
@endsection

@extends('base')

@section('main')
<div class="container">
<div class="row">
<div class="col-md-10 offset-md-1">
<div class="panel panel-default">
<div class="panel-heading">Інформація</div>
<div class="panel-body">
<canvas id="canvas" height="280" width="600"></canvas>
</div>
</div>
</div>
</div>
</div>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.3/Chart.min.
js"></script>
<script>
var year = <?php echo $year; ?>;
var user = <?php echo $user; ?>;
var barChartData = {
labels: year,
datasets: [{
label: 'Користувач',
backgroundColor: "pink",
```

```
data: user
}]
};

window.onload = function() {
var ctx = document.getElementById("canvas").getContext("2d");
window.myBar = new Chart(ctx, {
type: 'bar',
data: barChartData,
options: {
elements: {
rectangle: {
borderWidth: 2,
borderColor: '#c1c1c1',
borderSkipped: 'bottom'
}
},
responsive: true,
title: {
display: true,
text: 'Щорічна реєстрація користувача'
}
}
});
};
</script>
@endsection


<!DOCTYPE html>
<html lang="uk">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-
scale=1.0,maximum-scale=1">

<title>KinoPoshuk</title>

<!-- Loading third party fonts -->
<link
href="https://fonts.googleapis.com/css?family=Roboto:300,400,700|"
rel="stylesheet" type="text/css">
<link href="{{ asset('/fonts/font-awesome.min.css') }} "
rel="stylesheet" type="text/css">

<!-- Loading main css file -->
<link rel="stylesheet" href="{{ asset('style.css') }}">

<!--[if lt IE 9]>
<script src="{{ asset('js/ie-support/html5.js') }}"></script>
<script src="{{ asset('js/ie-support/respond.js') }}"></script>
<![endif]-->
```

```
<link
href="https://netdna.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.c
ss" rel="stylesheet">

<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-star-
rating/4.0.2/css/star-rating.min.css" />

<script
src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.js"><
/script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-star-
rating/4.0.2/js/star-rating.min.js"></script>

<link href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css" rel="stylesheet">

<link
href="https://fonts.googleapis.com/css?family=Open+Sans:400,700"
rel="stylesheet">

</head>


<body>

<div id="site-content">
<header class="site-header">
<div class="container">
<a id="branding">
<img src="/logo.png" alt="" class="logo">
<div class="logo-copy">
<h1   class="site-title"><b> КіноПошук </b></h1>
<small . class="site-description">Знайди своє кіно</small>
</div>
</a> <!-- #branding -->

<div class="main-navigation">
<button type="button" class="menu-toggle"><i class="fa fa-
bars"></i></button>
<ul class="menu">

@if( auth()->check() )
<li class="menu-item"> <a class="nav-link" href="/home">Привіт, {{
auth()->user()->name }}</a></li>
@endif
<li class="menu-item current-menu-item"><a href="/">Головна</a></li>

@if( auth()->check() )
<li class="menu-item"> <a class="nav-link" href="/chats">Чат</a></li>
<li class="menu-item"> <a class="nav-link"
href="/logout">Вихід</a></li>
@else
```

```
<li class="menu-item"> <a class="nav-link"
href="/login">Вхід</a></li>
<li class="menu-item">     <a class="nav-link"
href="/register">Регестрація</a></li>
@endif

<div class="btn-group" role="group">
<div class="dropdown dropdown-lg ">
<form action="{{ route('searchname') }}" method="GET" class="search-
form">
<input  type="text" id="s" name="s" placeholder="Пошук..."
class="active" >
<button type="submit" class="btn btn-primary"><span class="fa fa-
search" aria-hidden="true"></span></button>
</form>

<form action="{{ route('searchfilter') }}" method="GET"
class="search-form">
<button type="button" class="btn btn-lg btn-secondary dropdown-toggle
dropdown-toggle-split" data-toggle="dropdown" aria-haspopup="true"
aria-expanded="false">
</button>
<div class=" dropdown-menu dropdown-menu-right" role="menu">

<div  class="filters" >
<label for="category" class="text-dark">Додаткові параметри пошуку
кінофільмів:</label>
<select name="generes_id" id="input">
<option value="0">Жанр</option>
@foreach (\App\Models\Generes::select ('g_id','g_name')-
>orderby('g_name','asc')->get() as $generes)
<option value="{{ $generes->g_id }}" {{ $generes->g_id ==
$selected_id['generes_id'] ? 'selected' : '' }}>
{{ $generes['g_name'] }}
</option>
@endforeach
</select>
<div style="margin-top: 5px;" >
<select name="country_id" id="input">
<option value="0">Країна </option>
@foreach (\App\Models\FilmsModel::select('f_country')-
>orderby('f_country','asc')->distinct()->get() as $fcountry)
<option value="{{ $fcountry->f_country }}" {{ $fcountry->f_country ==
$selected_id['country_id'] ? 'selected' : '' }}>
{{ $fcountry['f_country'] }}
</option>
@endforeach
</select>
<select name="year_id" id="input">
<option value="0">Рік</option>
@foreach (\App\Models\FilmsModel::select('f_year')-
>orderby('f_year','desc')->distinct()->get() as $fyear)
<option value="{{ $fyear->f_year }}" {{ $fyear->f_year ==
$selected_id['year_id'] ? 'selected' : '' }}>
```

```
{{ $fyear['f_year'] }}
</option>
@endforeach
</select>
<input style="margin-top: 15px;"  type="text" id="s" name="s"
placeholder="актери/режисери" class="active" >
<button><i class="fa fa-search"></i></button>

<a href="/" class="btn btn-danger">Очистити</a>
</div>
</div>
</div>

</form>
</div>
</div>
</ul> <!-- .menu -->
</div>
</div>

</header>
<main class="main-content">
<div class="container">

<div class="page">

@yield('main_content')

</div>
</div>
</main>

<footer class="site-footer">
<div class="container">
<div class="colophon"></div>

</div> <!-- .container -->

</footer>

</div>

<script src="{{ asset('/js/jquery-1.11.1.min.js') }}"></script>
<script src="{{ asset('/js/plugins.js') }}"></script>
<script src="{{ asset('/js/app.js') }}"></script>
```

ДОДАТОК В

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК Г

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файла	**Опис**
Пояснювальні документи	
Диплом_ Головко.doc	Пояснювальна записка. Документ Word.
Диплом_ Головко.pdf	Пояснювальна записка в форматі PDF.
Програма	
kino_project.rar	Архів. Містить коди програми і Програму.
Презентація	
Презентація_ Головко.pptx	Презентація.