

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавра

(ступінь вищої освіти)

на тему РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ РЕЦЕНЗУВАННЯ
ТЕЛЕСЕРІАЛІВ

SOFTWARE DEVELOPMENT FOR TV SERIES REVIEWS

Виконав: студент(ка) 4 курсу, групи КНТ-127
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Череватенко Д.С.

(прізвище та ініціали)

Керівник Федорончак Т.В.

(прізвище та ініціали)

Рецензент Казурова А.Є.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Череватенка Данила Станіславовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка програмного забезпечення рецензування телесеріалів

Software Development for TV Series Reviews

керівник проєкту (роботи) Федорончак Тетяна Василівна, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “30” березня 2021 року № 103

2. Строк подання студентом проєкту (роботи) 25.05.2021

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури програми. 3. Розробка програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Федорончак Т.В., доцент		
Нормоконтролер	Дейнега Л.Ю., ст. викладач		

7. Дата видачі завдання “17” березня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2-3 тижні	Розділ 1
3	Розробка архітектури програми	4 тиждень	Розділ 2
4	Розробка програми	5-6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	8 тиждень	Додатки
7	Захист роботи	9 тиждень	

Студент(ка)

(підпис) Череватенко Д.С.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Федорончак Т.В.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 80 с., 30 рис., 2 табл., 3 дод., 10 джерел.

ТЕЛЕСЕРІАЛ, РЕЦЕНЗІЯ, ПЕРЕЛІК ПЕРЕГЛЯДУ, ВЕБЗАСТОСУНОК, БАЗА ДАНИХ, DJANGO.

Об'єкт дослідження – процес рецензування телесеріалів. Предмет дослідження – програмне забезпечення рецензування телесеріалів.

Мета роботи – розроблення програмного забезпечення рецензування телесеріалів для підтримки індивідуалізації узагальнених результатів оцінювання та рецензування.

Матеріали, методи та технічні засоби: мова програмування Python, вебфреймворк Django, система керування базами даних MySQL.

Результати. Проведено аналіз предметної області з порівнянням програмних аналогів і виділенням особливостей програми, що розроблено. Виконано проєктування програми з визначенням функціональних вимог до програми, проєктування структури даних, визначенням архітектури застосунку. Описано створену структуру програмного проєкту та розроблені складові цієї структури.

Висновки. Розроблений програмний застосунок дозволяє працювати з ним віддалено на основі клієнт-серверної моделі, отримуючи дані про телесеріали, їх сезони, епізоди, акторів, режисерів, продюсерів, жанри та дозволяючи формувати переліки споріднених користувачів, що надає можливість зі всіх статистичних даних формувати підмножину, що стосується цієї групи людей, сприяючи індивідуалізації узагальнених результатів оцінювання та рецензування телесеріалів.

Галузь використання. Інформаційні бази для поціновувачів телесеріалів, стримінгові сервіси.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 80 pages, 30 figures, 2 tables, 3 appendixes, 10 sources.

TV SERIES, REVIEW, LIST OF VIEWING, WEB APPLICATION, DATABASE, DJANGO.

The object of research is the process of reviewing TV series. The subject of research is software for reviewing TV series.

The purpose of the work is to develop software for reviewing TV series to support the individualization of generalized evaluation and reviewing of TV series.

Materials, methods and technical means: Python programming language, Django web framework, MySQL database management system.

Results. The analysis of the subject area with the comparison of software analogues and highlighting the features of the software is developed. The design of the software with the definition of functional requirements for the program, the design of the data structure, the definition of the application architecture was performed. The created structure of the software project is described and the components of this structure are developed.

Conclusions. The developed software application allows to work with it remotely on the basis of client-server model, receiving data on TV series, its seasons, episodes, actors, directors, producers, genres and allowing to form lists of related users, which allows to form a subset of all statistics for the group of people. It contributes to the individualization of the generalized results of evaluation and reviewing of TV series.

Field of use. Information bases for fans of TV series, streaming services.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	9
1.1 Існуючі програмні засоби для рецензування телесеріалів.....	9
1.2 Визначення особливостей програми, що розробляється	14
1.3 Висновки за розділом 1	15
2 Розробка архітектури програми.....	17
2.1 Функціональні вимоги до програмного забезпечення	17
2.2 Вибір мови програмування	21
2.3 Проектування бази даних	21
2.4 Висновки за розділом 2	31
3 Розробка програми	32
3.1 Алгоритм функціонування програмного забезпечення	32
3.2 Структура розробленого програмного забезпечення	34
3.3 Опис прийнятих під час реалізації рішень	35
3.4 Висновки за розділом 3	41
4 Експлуатація, тестування та експериментальне дослідження програми	43
4.1 Призначення програми	43
4.2 Умови виконання програми	43
4.3 Виконання програми.....	43
Висновки	52
Перелік джерел посилання	53
Додаток А Технічне завдання	54
Додаток Б Текст програми	59
Додаток В Слайди презентації.....	73

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

UML – Unified Modeling Language;

БД – база даних.

ВСТУП

У сучасному світі телесеріали набрали великої популярності, що перевищує популярність повнометражних кінофільмів. Цьому сприяв розвиток стримінгових сервісів, з популярністю яких і зростала популярність телесеріалів.

Інформаційні технології отримали широке впровадження не тільки в просторі створення кінофільмів та телесеріалів, але і відносно поціновувачів фільмів, які бажали мати централізовані рішення для пошуку всієї інформації про їх улюблені кінофільми. На хвилі цієї необхідності і отримав велику популярність сервіс IMDb. На даний момент завдяки існуванню цього сервісу кожен користувач може знайти основну інформацію про будь-який фільм у світі, зокрема і про телесеріал. Окрім того велику значимість отримали рейтинги, які складають такі сервіси. Рейтинг IMDb є однією з основних оцінок для будь-якого фільму. Однак такі рейтинги є узагальненням серед всіх користувачів, в той час як деякий фільм, що має відносно невелику оцінку серед всіх користувачів, може мати відносно велику оцінку для певної групи людей. Ці люди можуть мати схожі смаки, уподобання і вони, сприяючи розповсюдженню улюблених телесеріалів, потребують отримання уявлення про оцінку кожного телесеріалу в такій групі людей. Даної особливості немає на сучасних вебсайтах, що існують у даній області. Зазвичай користувачі діляться цією інформацією в соціальних мережах, відкривають один одному свої переліки кінофільмів, але потреба в узагальненні результатів від різних користувачів у одній групі залишається. Саме на вирішення цієї проблеми і спрямовано розроблення програмного забезпечення у даній дипломній кваліфікаційній роботі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Існуючі програмні засоби для рецензування телесеріалів

Найбільшим джерелом даних про кінофільми у світі є IMDb, тому огляд існуючих програмних засобів для вирішення проблеми потрібно розпочати саме з неї.

IMDb [1] (рис. 1.1) надає велику кількість інформації як стосовно повнометражних фільмів, так і стосовно телесеріалів.

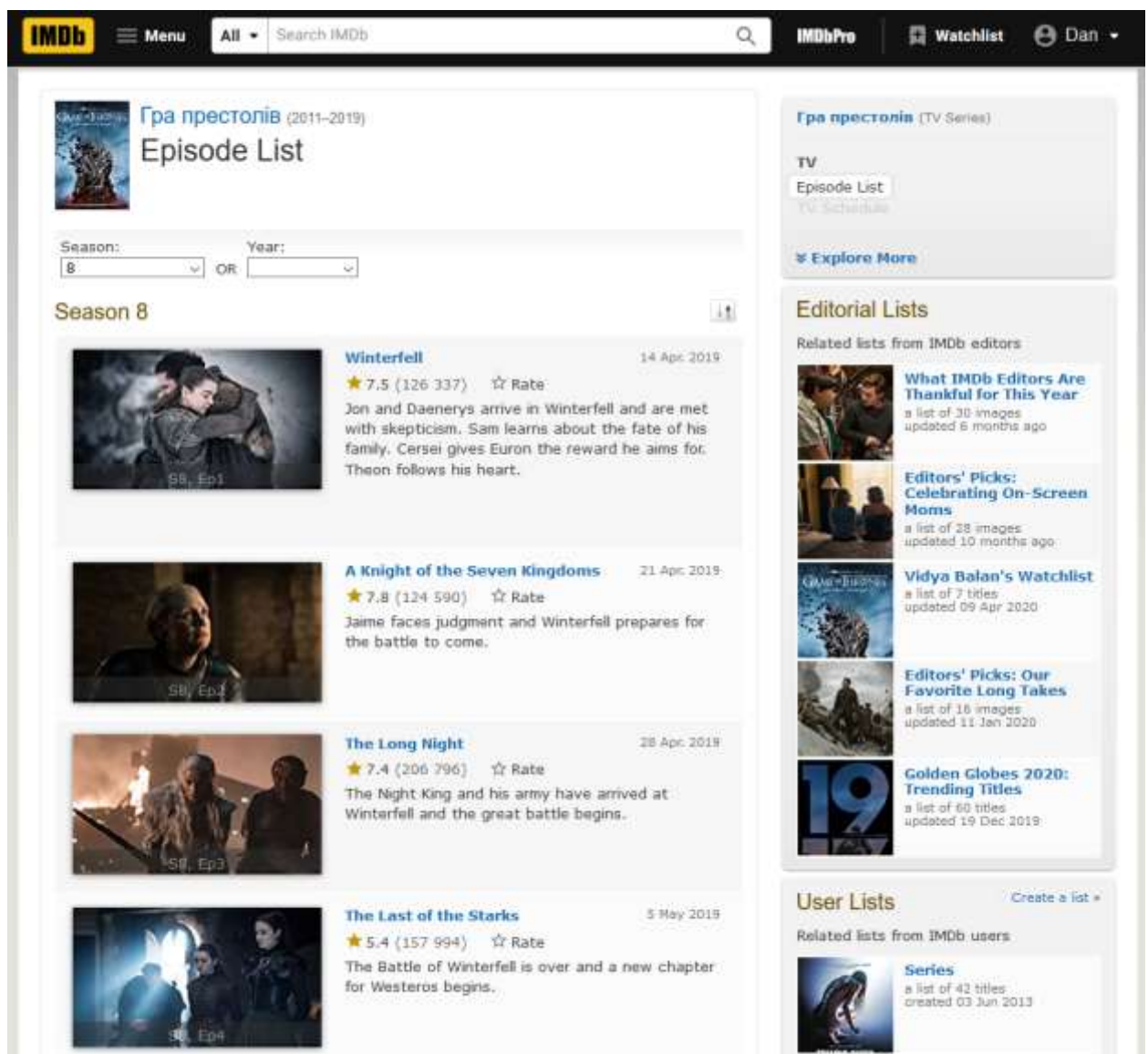


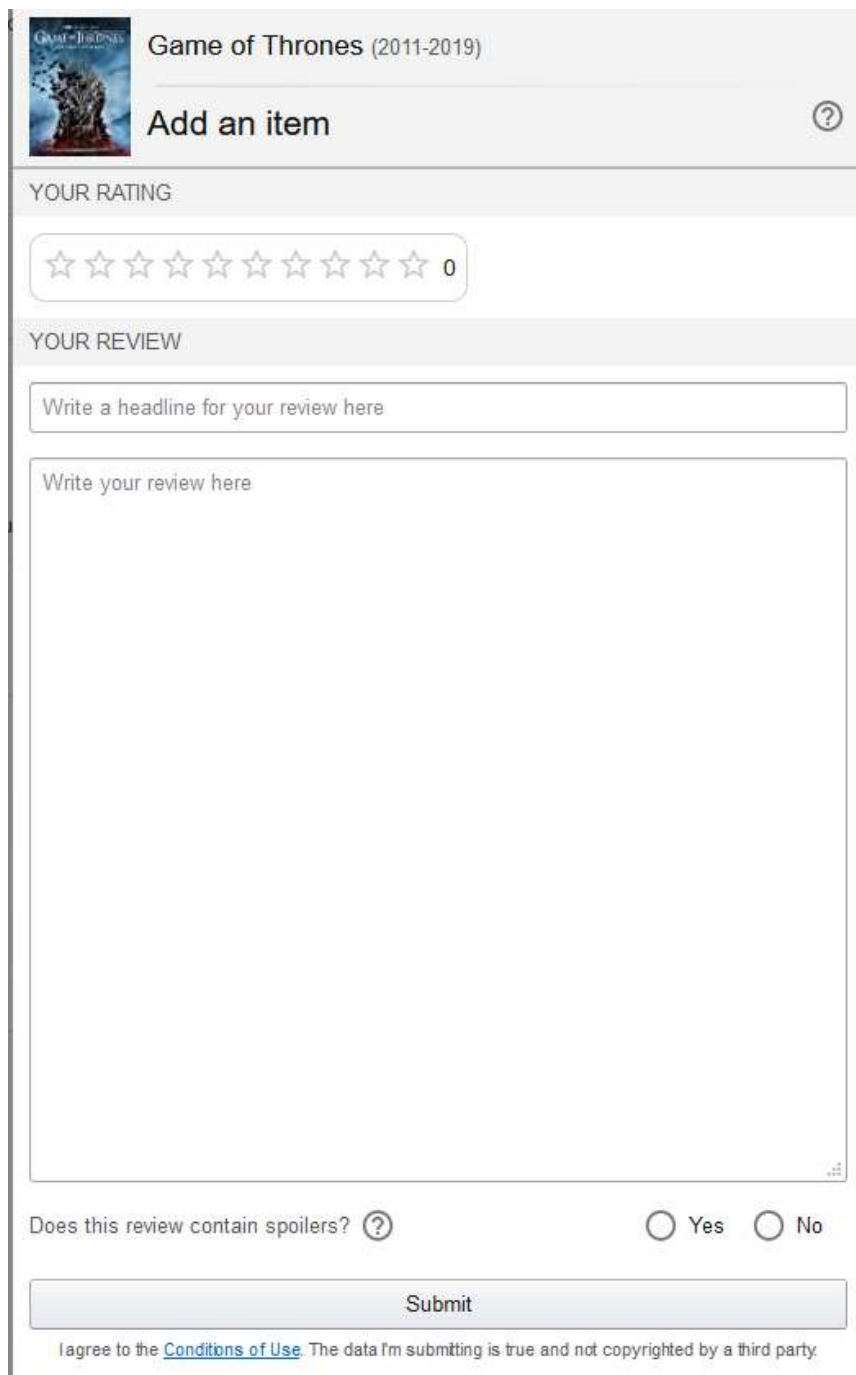
Рисунок 1.1 – Перегляд інформації про сезон телесеріалу в сервісі IMDb

Логіка представлення інформації однакова і для кінофільмів, і для телесеріалів, але при цьому для телесеріалів на сторінці відображається також переліків сезонів, що дозволяє в свою чергу переглянути перелік серій за кожним сезоном.

IMDB надає доступ до наступних функцій відносно телесеріалів:

- перегляд повної інформації про телесеріал загалом, його сезони, окремі серії;
- перегляд рецензій на телесеріал або епізоди з формуванням оцінки за серіал загалом або за окремою серією;
- формування переліку телесеріалів для подальшого перегляду: дана функція дозволяє відстежувати майбутні телесеріали, які користувач бажає переглянути, за замовчуванням перелік приватний, але можна зробити його загальнодоступним та поділитися ним із друзями, також його можна експортувати у формат електронних таблиць для подальшого стандартного використання;
- створення власних рецензій на телесеріал загалом або на окрему його серію (рис. 1.2), що включає визначення заголовку рецензії, виставлення оцінки, визначення тексту рецензії та повідомлення про наявність спойлерів у тексті рецензії;
- перегляд власних раніше виставлених оцінок;
- створення власних переліків телесеріалів: звичайні користувачі можуть створювати власні переліки, якими можна ділитися з іншими, але варто звернути увагу на те, що немає центральної сторінки, де можна переглядати переліки користувачів;
- перегляд власної минулої активності;
- перегляд телесеріалів за жанрами;
- перегляд рейтингів телесеріалів;
- перегляд телесеріалів, які найбільше переглядають користувачі;
- надається декілька динамічних переліків найкращого, за якими можна знаходити нові телесеріали для перегляду, але такі переліки

змінюються не дуже часто, тому після їх перегляду можуть знадобитись переліки «Останні переможці Оскара» та «Найпопулярніші за жанром» [2]-[4].



Game of Thrones (2011-2019)

Add an item

YOUR RATING

☆☆☆☆☆☆☆☆ 0

YOUR REVIEW

Write a headline for your review here

Write your review here

Does this review contain spoilers? ☐ Yes ☐ No

Submit

I agree to the [Conditions of Use](#). The data I'm submitting is true and not copyrighted by a third party.

Рисунок 1.2 – Створення рецензії на телесеріал на сервісі IMDb

Ще однією базою кінофільмів є TMDb [5] (рис. 1.3). Даний засіб відрізняється тим, що самі користувачі можуть редагувати вміст, що

призводить до того, що даний ресурс відзначається великою кількістю медіаресурсів, присвячених кінофільмам, включаючи фотографії, логотипи, постери, відеоматеріали. При цьому дане рішення має україномовний інтерфейс, що є позитивною особливістю, адже розширення кількості засобів локалізації завжди сприяє проникненню в середовище відповідної країни ширшим шляхом.

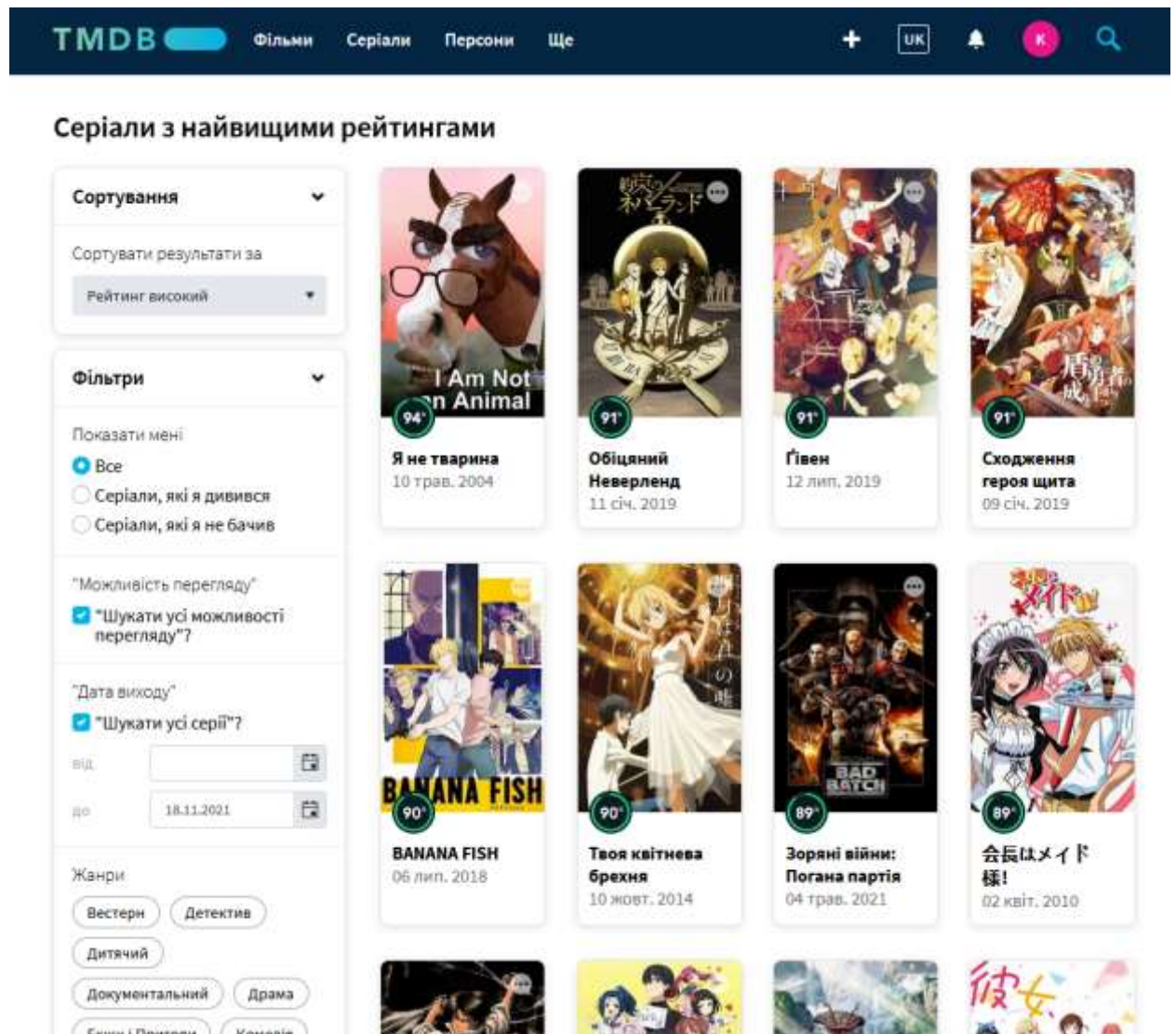


Рисунок 1.3 – Пошук телесеріалів на сервісі TMDb

TMDb розділяє всі ресурси на фільми та серіали, тобто існують окремі засоби для роботи з телесеріалами. Для телесеріалів можна переглянути перелік популярних, побачити програму телесеріалів на

сьогодні, програму в дану мить та рейтинг телесеріалів, який окрім відтворення переліку має ще й велику кількість фільтрів, що перетворюють його в повноцінний пошук, зокрема можна встановити обмеження рейтингу, визначити потрібні жанри.

Користувач може виставляти оцінки телесеріалам, може створювати власні переліки, а також створювати рецензії (рис. 1.4).

The screenshot shows a web form for submitting a review on the TMDb website. On the left, there is a grid of character portraits from the TV series 'Grey's Anatomy'. To the right of the grid, the form is titled 'A review' and includes the following fields: 'Title: Анатомія Грей (2005)' and 'Language: Український (uk)'. Below these fields are text formatting icons (bold, italic, list, link, quote) and a text area containing the placeholder text 'This is my first review.'. To the right of the main form is a 'Preview' section with the same placeholder text. At the bottom right of the form, there are two buttons: 'Cancel' and 'Submit'.

Рисунок 1.4 – Створення рецензії на телесеріал на сервісі TMDb

Щодо рецензій варто відзначити, що цей засіб недостатньо зручний і ефективний, оскільки рецензія зокрема не має заголовку, тобто відображати перелік рецензій у такому випадку достатньо складно, окрім того кожна рецензія має пройти перевірку і підтвердження зі сторони адміністрації.

Таке рішення щодо організації рецензування сприяє тому, що кількість рецензій на вебсайті достатньо обмежена, тому діляться власним враженням користувачі в основному за допомогою оцінок та в обговореннях фільмів та телесеріалів. Рецензії використовується головним чином для виділення особливих думок, а не для створення масового інструменту як це виконано в IMDb або подібних рішеннях.

1.2 Визначення особливостей програми, що розробляється

Порівнюючи розглянуті програмні аналоги, варто відзначити, що в якості основного прототипу для рішення, що розробляється, потрібно використовувати IMDb, оскільки TMDb має достатньо обмежені засоби рецензування.

В якості критеріїв, що розглядались, виділено:

- перегляд інформації про телесеріали: чи виділяє дане рішення телесеріали в окрему групу серед всіх фільмів;
- ефективність рецензування: якщо створені рецензії з невисокою ймовірністю потраплять в результаті на вебсайт, то ефективність є невисокою, відповідно невелика кількість користувачів буде створювати власні відгуки у формі рецензій;
- ефективність оцінювання: чи представлена система виставлення оцінок користувачами будь-якому телесеріалу і підведення загальної оцінки в результаті;
- україномовний інтерфейс: в Україні це може бути додатковою перевагою, як і локалізація або її можливість загалом;
- створення переліків телесеріалів: будь-який подібний сервіс буде неефективним, якщо дозволить тільки переглядати інформацію про фільми, але не дозволить упорядковувати таку інформацію;
- створення переліків споріднених користувачів: це функціональна особливість, яка є головною особливістю програмного забезпечення, що розробляється.

Створення переліків споріднених користувачів ґрунтується на тому, щоб виділити зі всіх оцінок і рецензій користувачів саме ті, думці яких довіряє користувач. Це дозволить виділити певну підмножину телесеріалів та їх оцінок, яка буде більш репрезентативною для користувача, а відповідно і додатково приверне його увагу до роботи з програмою за рахунок

персоналізації. Жодне з існуючих на даний момент рішень такої функціональної особливості не має.

Результати порівняння програми, що розробляється, з її програмними аналогами зведено в табл. 1.1.

Таблиця 1.1 – Порівняння програмних аналогів

	IMDB	TMDB	Нова програма
Перегляд інформації про телесеріали	Так	Так	Так
Ефективність рецензування	Висока	Обмежена	Висока
Ефективність оцінювання	Висока	Висока	Висока
Україномовний інтерфейс	Ні	Так	Так
Створення переліків телесеріалів	Так	Так	Так
Створення переліків споріднених користувачів	Ні	Ні	Так

1.3 Висновки за розділом 1

Завданнями дипломної роботи є:

- аналіз предметної області;
- проєктування програмного забезпечення;
- реалізація програмного забезпечення;
- тестування і налагодження програмного забезпечення.

У розділі було вирішено перше завдання роботи: проаналізовано існуючі програмні рішення щодо їх функціональних можливостей для

рецензування телесеріалів, виконано порівняння їх між собою та з програмою, що розробляється. Основною особливістю розроблюваної програми є створення переліків споріднених користувачів, що дозволять виділити зі всіх оцінок і рецензій користувачів саме ті, думці яких довіряє користувач.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Функціональні вимоги до програмного забезпечення

Для підтримки реалізації ідеї, описаної в першому розділі, програма повинна дозволяти виконувати наступні функції:

- внесення телесеріалів, їх сезонів та серій;
- внесення акторів, режисерів, продюсерів, компаній-виробників, жанрів;
- керування внесеними телесеріалами, їх сезонами, серіями, акторами, режисерами, продюсерами, компаніями-виробниками, жанрами;
- створення рецензій на телесеріали;
- створення рецензій на окремі серії телесеріалів;
- обчислення середньої оцінки телесеріалу за оцінками рецензій на нього;
- обчислення середньої оцінки сезону телесеріалу на основі середніх оцінок серій телесеріалу;
- обчислення середньої оцінки серії телесеріалу на основі рецензій;
- перегляд даних телесеріалу;
- перегляд даних сезонів телесеріалу;
- перегляд даних серій сезону телесеріалу;
- перегляд інформації про акторів, включаючи перелік їх серіалів;
- перегляд інформації про режисерів, включаючи перелік їх серіалів;
- перегляд інформації про продюсерів, включаючи перелік їх серіалів;
- пошук телесеріалів, їх сезонів та окремих серій за їх назвою;
- пошук режисерів, акторів, продюсерів;
- перегляд рецензій на телесеріал;
- перегляд рецензій на серію телесеріалу;
- додавання користувачів до власного переліку споріднених користувачів;

- вилучення користувачів з власного переліку споріднених користувачів;
- перегляд рецензій споріднених користувачів;
- перегляд рейтингу телесеріалів;
- перегляд рейтингу серій телесеріалу;
- перегляд рейтингу сезонів телесеріалу;
- перегляд рейтингу телесеріалів за рецензіями споріднених користувачів;
- відправлення поради для перегляду телесеріалу користувачу;
- реєстрація користувачів;
- авторизація користувачів.

Для забезпечення описаних функцій користувачів мають виконувати дві ролі, адміністратора та звичайного користувача. Звичайний користувач може бути авторизованим або неавторизованим. У результаті три ролі користувачів взаємодіють з програмою: адміністратор, неавторизований користувач та авторизований користувач. Для моделювання взаємодії користувачів з програмою було використано діаграми прецедентів Unified Modeling Language (UML) [6].

Адміністратор (рис. 2.1) має наступні сценарії роботи з програмою:

- внесення даних, що узагальнює сценарії внесення телесеріалів, виконавців, виробників, жанрів;
- керування внесеними даними, що узагальнює сценарії керування телесеріалами, виконавцями, виробниками, жанрами.

Неавторизований користувач (рис. 2.2) має наступні сценарії роботи:

- перегляд повних даних телесеріалу;
- перегляд інформації про виконавців;
- пошук даних, що узагальнює пошук телесеріалів та виконавців;
- перегляд рецензій;
- перегляд рейтингу телесеріалів;
- реєстрація.

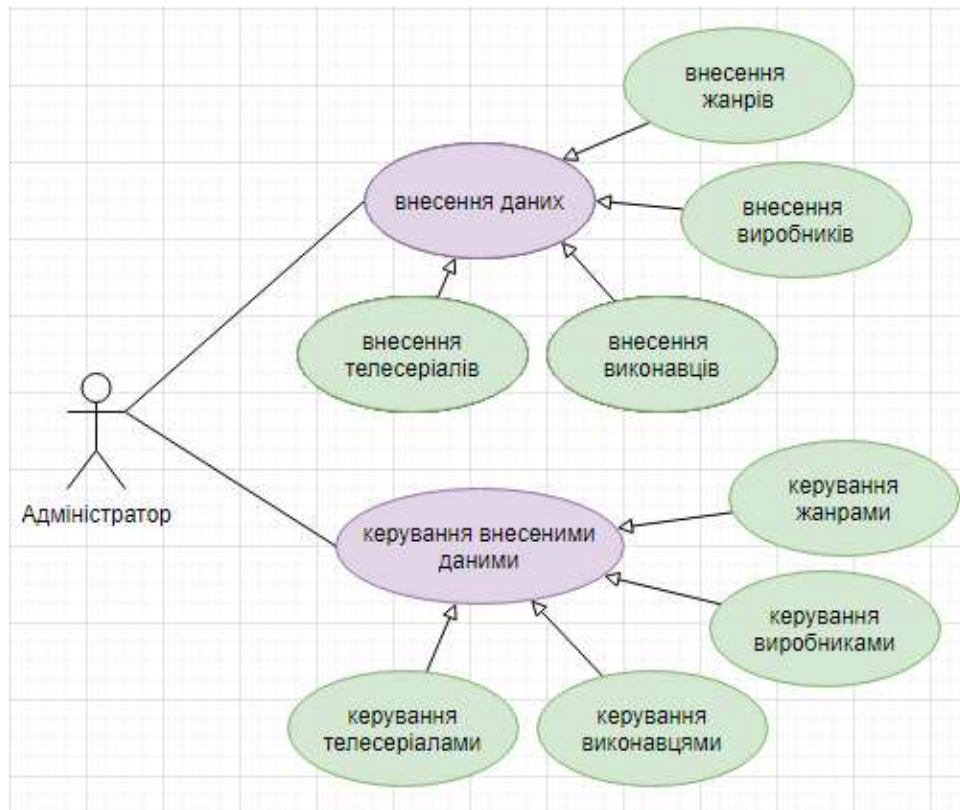


Рисунок 2.1 – Діаграма прецедентів UML адміністратора

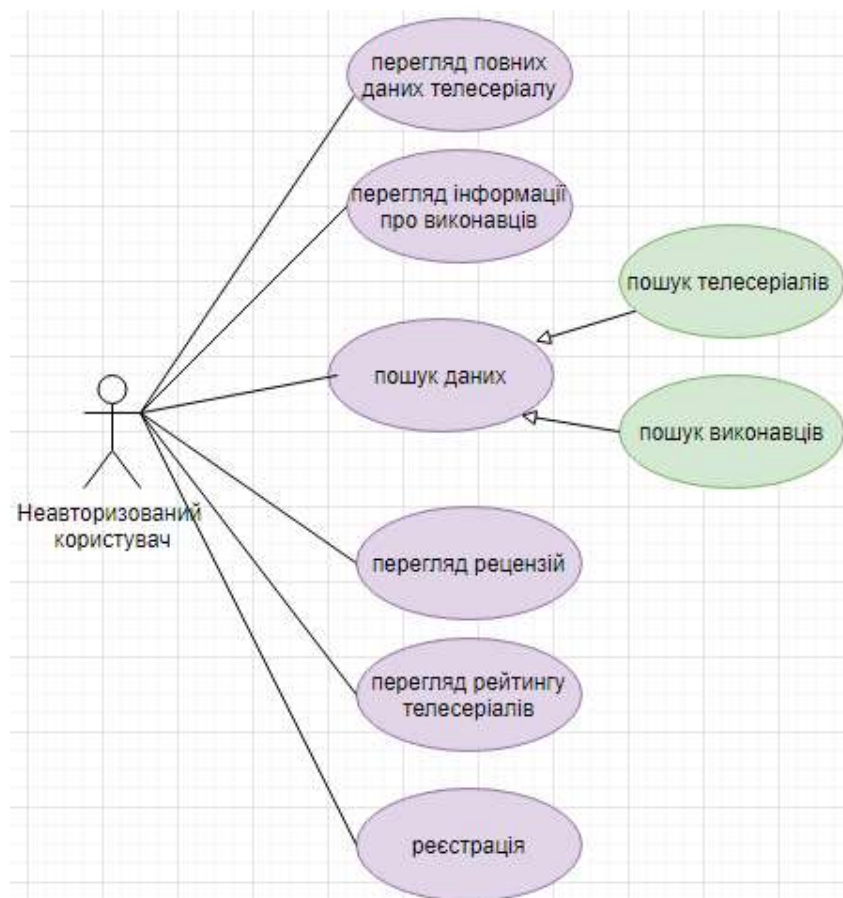


Рисунок 2.2 – Діаграма прецедентів UML неавторизованого користувача

Авторизований користувач (рис. 2.3) має наступні сценарії роботи з програмою:

- створення рецензій на телесеріали та окремі серії телесеріалів;
- керування власним переліком споріднених користувачів;
- перегляд рецензій споріднених користувачів;
- перегляд рейтингу телесеріалів за рецензіями споріднених користувачів;
- відправлення поради для перегляду телесеріалу користувачу;
- авторизація.

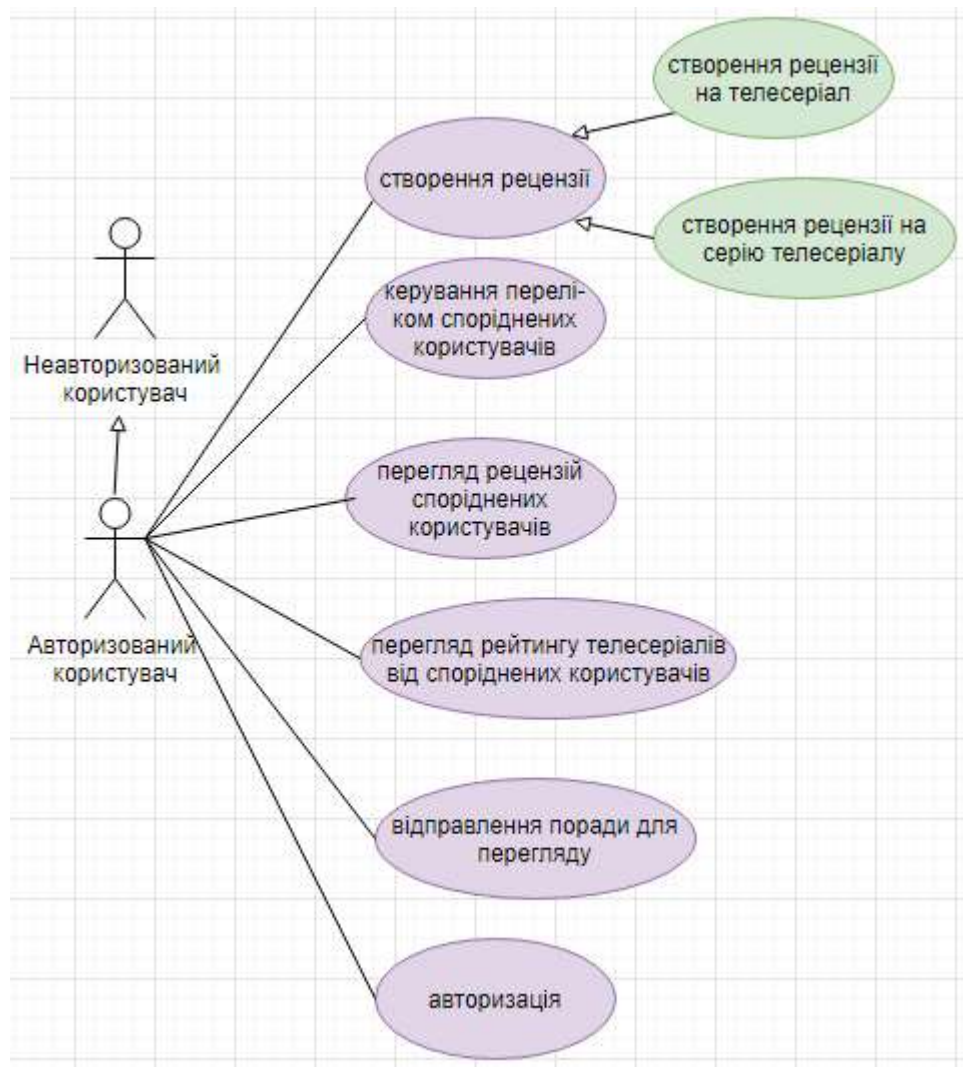


Рисунок 2.3 – Діаграма прецедентів UML авторизованого користувача

2.2 Вибір мови програмування

Для реалізації програмного забезпечення у відповідності з описаними функціональними вимогами було виконано вибір мови програмування серед мов, придатних для розробки вебзастосунків, Python [7]-[8] та Java (табл. 2.1).

Таблиця 2.1 – Вибір мови програмування

	Python	C#
Легкість написання коду	Підвищена	Середня
Вебфреймворки	Django	ASP.NET
Повністю безкоштовний набір засобів розробки	Так	Ні
Спільнота розробників	Велика і вільна	Велика

За результатами порівняння було обрано мову Python, враховуючи те, що вона дозволяє створювати менше коду, менше витратити часу на це, має повністю безкоштовні засоби, але до цього ж і підтримується великою спільнотою з розробки вільного програмного забезпечення, що дозволяє якісно і з належною підтримкою розробляти програмне забезпечення.

2.3 Проектування бази даних

Системою керування базами даних (БД) було визначено MySQL, оскільки вона є придатною на практиці при роботі в умовах високої завантаженості, до того ж є безкоштовною.

Першою основною сутністю предметної області є серіал, представлений сутністю TVSeries.

Важливою відмінністю саме телесеріалів від повнометражних кінофільмів є те, що кожен з телесеріалів повинен мати послідовність серій, при чому ці серії об'єднуються в сезон. Кожен серіал має хоча би один сезон. Тому наступними сутностями предметної області є сутності сезону телесеріалу TVSeason та серії телесеріалу Episode.

Сутність User визначає користувача, який працює з програмою і забезпечує індивідуалізацію рецензування.

Наступною важливою сутністю є рецензія. Однак у даному випадку рецензії, на відміну від повнометражних фільмів, можуть складатися на весь серіал цілком, а можуть складатися на окремі серії. Тому в результаті було створено рецензію на серію телесеріалу EpisodeReview та рецензію на серіал TVSeriesReview.

Сутність Company визначає компанію, що виробляє телесеріали.

Сутність Genre визначає жанри телесеріалів.

Сутність Participant визначає виконавця телесеріалу. Це сутність, яка об'єднує в собі режисера, продюсера та актора. Кожен з них розглядається як виконавець.

Сутність Media має забезпечити централізоване збереження медіафайлів, до яких можуть відноситися фотографії та відеофайли.

Сутність Advice має забезпечувати підтримку роботи функції особистих порад телесеріалів від користувача до користувача, тобто вона фактично визначає таку пораду.

Таблиця TVSeries забезпечує зберігання даних відповідної сутності за допомогою полів:

- id – номер серіалу;
- headline – назва серіалу;
- story – опис (сюжет) серіалу;
- site – вебсайт серіалу;
- release – дата випуску серіалу;
- companyId – номер компанії, яка випустила серіал;

- score – середня оцінка серіалу;
- reviews – кількість рецензій з оцінками, на основі яких виставлено дану середню оцінку.

Поле companyId таблиці є зовнішнім ключем. Він зв'язує таблицю з таблицею Company, між якими зв'язок один до багатьох від таблиці Company: одна компанія може виробляти багато серіалів, але кожен з них виробляється тільки однією компанією.

Таблиця TVSeason забезпечує зберігання даних відповідної сутності за допомогою полів:

- id – номер сезону серіалу;
- num – номер сезону;
- tvseriesId – номер серіалу;
- headline – назва сезону серіалу (може бути відсутня);
- period – період часу, протягом якого випускався даний сезон (використовується для короткого опису без точних дат);
- score – середня оцінка сезону серіалу, розрахована усередненням оцінок всіх серій цього сезону серіалу;
- reviews – кількість рецензій з оцінкою, на основі яких було отримано оцінку (сумарна кількість рецензій, написаних на всі серії сезону).

Поле tvseriesId таблиці є зовнішнім ключем. Він зв'язує таблицю з таблицею TVSeries, між якими зв'язок один до багатьох від таблиці TVSeries: один телесеріал може мати багато сезонів, але кожен конкретний сезон належить тільки одному телесеріалу.

Таблиця Episode забезпечує зберігання даних відповідної сутності за допомогою полів:

- id – номер серії;
- tvseasonId – номер сезону серіалу, до якого належить ця серія;
- headline – назва серії;
- story – опис (сценарій) серії;
- release – дата випуску серії;

- runtime – тривалість серії;
- score – середня оцінка, виставлена в рецензіях серії;
- reviews – кількість рецензій з оцінкою, яку отримала дана серія серіалу.

Поле tvseasonId таблиці є зовнішнім ключем. Він зв'язує таблицю з таблицею TVSeason, між якими зв'язок один до багатьох від таблиці TVSeason: один он може мати багато серій, але кожна серія належить тільки одному сезону.

Таблиця User забезпечує зберігання даних відповідної сутності за допомогою полів:

- id – номер облікового запису користувача;
- login – логін облікового запису користувача;
- password – пароль облікового запису користувача;
- activity – дата останньої авторизації користувача;
- photo – фотографія користувача.

Таблиця EpisodeReview забезпечує зберігання даних відповідної сутності за допомогою полів:

- id – номер рецензії на серію серіалу;
- episodeId – номер серії;
- userId – номер облікового запису користувача, який створив рецензію;
- release – дата і час створення рецензії;
- fullreview – повний текст рецензії;
- score – оцінка, виставлена в рецензії (може бути пропущена).

Поле episodeId таблиці є зовнішнім ключем. Він зв'язує таблицю з таблицею Episode, між якими зв'язок один до багатьох від таблиці Episode: одна серія може мати багато рецензій, але кожна рецензія серії має бути написана на одну конкретну серію.

Поле userId таблиці є зовнішнім ключем. Він зв'язує таблицю з таблицею User, між якими зв'язок один до багатьох від таблиці User: кожен

користувач може написати багато рецензій, але кожна рецензія може бути написана тільки одним користувачем.

Таблиця TVSeriesReview забезпечує зберігання даних відповідної сутності за допомогою полів:

- id – номер рецензії на серіал;
- tvseriesId – номер серіалу;
- userId – номер облікового запису користувача, який написав рецензію;
- release – дата і час створення рецензії;
- fullreview – повний текст рецензії;
- score – оцінка рецензії.

Поле tvseriesId таблиці є зовнішнім ключем. Він зв'язує таблицю з таблицею TVSeries, між якими зв'язок один до багатьох від таблиці TVSeries: один телесеріал може мати багато рецензій, але кожна рецензія телесеріалу може бути написана тільки на один конкретний телесеріал і не може об'єднувати одразу декілька.

Поле userId таблиці є зовнішнім ключем. Він зв'язує таблицю з таблицею User, між якими зв'язок один до багатьох від таблиці User: кожен користувач може написати багато рецензій, але кожна рецензія на телесеріал може бути написана тільки одним користувачем.

Таблиця Company забезпечує зберігання даних відповідної сутності за допомогою полів:

- id – номер компанії;
- headline – назва компанії.

Таблиця Genre забезпечує зберігання даних відповідної сутності за допомогою полів:

- id – номер жанру;
- headline – назва жанру.

Кожен жанр характеризує окремі епізоди телесеріалу та телесеріал загалом. В якості характеристик телесеріалу вибираються основні жанри. У

той же час окремі епізоди можуть мати в характеристиках жанри, які загалом серіал не характеризують.

Зв'язок між таблицями TVSeries і Genre багато до багатьох, оскільки кожен телесеріал може характеризуватися різними жанрами, а кожен жанр може характеризувати різні телесеріали. Для реалізації даного зв'язку було введено таблицю TVSeriesGenre.

Поля таблиці TVSeriesGenre:

- id – номер;
- tvseriesId – номер серіалу;
- genreId – номер жанру.

Поле tvseriesId таблиці є зовнішнім ключем. Він зв'язує таблицю з таблицею TVSeries, між якими зв'язок один до багатьох від таблиці TVSeries.

Поле genreId таблиці є зовнішнім ключем. Він зв'язує таблицю з таблицею Genre, між якими зв'язок один до багатьох від таблиці Genre.

Зв'язок між таблицями Episode і Genre багато до багатьох, оскільки кожна серія телесеріалу може характеризуватися різними жанрами, а кожен жанр може характеризувати різні серії. Для реалізації даного зв'язку було введено таблицю EpisodeGenre.

Поля таблиці EpisodeGenre:

- id – номер;
- episodeId – номер епізоду;
- genreId – номер жанру.

Поле episodeId таблиці є зовнішнім ключем. Він зв'язує таблицю з таблицею Episode, між якими зв'язок один до багатьох від таблиці Episode.

Поле genreId таблиці є зовнішнім ключем. Він зв'язує таблицю з таблицею Genre, між якими зв'язок один до багатьох від таблиці Genre.

Таблиця Participant забезпечує зберігання даних відповідної сутності за допомогою полів:

- id – номер виконавця;
- name – ім'я виконавця;

– photo – фотографія виконавця.

Зв'язок між таблицями Episode і Participant багато до багатьох, оскільки кожна серія телесеріалу може мати багато виконавців, а кожен виконавець може зніматися в різних серіях. Для реалізації даного зв'язку було введено таблицю EpisodeParticipant.

Таблиця EpisodeParticipant:

- id – номер;
- episodeId – номер епізоду;
- participantId – номер жанру;
- role – роль виконавця.

Поле episodeId таблиці є зовнішнім ключем. Він зв'язує таблицю з таблицею Episode, між якими зв'язок один до багатьох від таблиці Episode.

Поле participantId таблиці є зовнішнім ключем. Він зв'язує таблицю з таблицею Participant, між якими зв'язок один до багатьох від таблиці Genre.

Таблиця Media забезпечує зберігання даних відповідної сутності за допомогою полів:

- id – номер медіафайла;
- headline – назва, що використовується для опису файла;
- file – шлях до файла;
- type – тип медіафайла (фотографія або відео).

Оскільки будь-який медіафайл може бути характеристикою і телесеріалу загалом, і конкретного сезону телесеріалу, і конкретної серії, то було створено таблиці, які визначають роль медіафайла у взаємовідносинах з телесеріалом.

Таблиця EpisodeMedia визначає медіафайл як характеристику серії телесеріалу за допомогою полів:

- id – номер;
- episodeId – номер епізоду;
- mediaId – номер медіафайла.

Поле `episodeId` таблиці є зовнішнім ключем. Він зв'язує таблицю з таблицею `Episode`, між якими зв'язок один до багатьох від таблиці `Episode`.

Поле `mediaId` таблиці є зовнішнім ключем. Він зв'язує таблицю з таблицею `Media`, між якими зв'язок один до багатьох від таблиці `Media`.

Таблиця `TVSeriesMedia` визначає медіафайл як характеристики телесеріалу за допомогою полів:

- `id` – номер;
- `tvseriesId` – номер серіалу;
- `mediaId` – номер медіафайла.

Поле `episodeId` таблиці є зовнішнім ключем. Він зв'язує таблицю з таблицею `Episode`, між якими зв'язок один до багатьох від таблиці `Episode`.

Поле `tvseriesId` таблиці є зовнішнім ключем. Він зв'язує таблицю з таблицею `TVSeries`, між якими зв'язок один до багатьох від таблиці `TVSeries`.

Таблиця `TVSeasonMedia` визначає медіафайл як характеристики сезону телесеріалу за допомогою полів:

- `id` – номер;
- `tvseasonId` – номер сезону серіалу;
- `mediaId` – номер медіафайла.

Поле `tvseasonId` таблиці є зовнішнім ключем. Він зв'язує таблицю з таблицею `TVSeason`, між якими зв'язок один до багатьох від таблиці `TVSeason`.

Поле `mediaId` таблиці є зовнішнім ключем. Він зв'язує таблицю з таблицею `Media`, між якими зв'язок один до багатьох від таблиці `Media`.

Таблиця `UserList` визначає перелік споріднених користувачів для кожного користувача програми. Він дозволяє відслідковувати користувачу рецензії таких користувачів, оцінки, які вони виставляють телесеріалам. Таблиця має поля:

- `id` – номер;
- `userId` – номер облікового запису користувача (зовнішній ключ, таблиця `User`);

– user2Id – номер користувача, що присутній у переліку користувача userId (зовнішній ключ, таблиця User).

Таблиця Advice забезпечує зберігання даних відповідної сутності за допомогою полів:

- id – номер поради для перегляду;
- userListId – запис таблиці споріднених користувачів, який визначає від кого і кому направлено пораду;
- tvseriesId – номер серіалу, який користувач радить переглянути;
- release – дата і час створення поради.

Поле userListId таблиці є зовнішнім ключем. Він зв'язує таблицю з таблицею UserList, між якими зв'язок один до багатьох від таблиці UserList.

Поле tvseriesId таблиці є зовнішнім ключем. Він зв'язує таблицю з таблицею TVSeries, між якими зв'язок один до багатьох від таблиці TVSeries.

Переліки споріднених користувачів використовуються не тільки для перегляду рецензій користувачів з таких переліків, але і для отримання інформації про їх оцінку певному телесеріалу. Оскільки зі збільшенням кількості користувачів кількість таких записів буде значно збільшуватися, відповідно підрахунок такого середнього балу кожного телесеріалу для кожного переліку кожного разу буде вимагати надто багато часу, то таблиця TVSeriesList має дозволити зберігати такі узагальнення для телесеріалу від імені всіх споріднених користувачів для даного користувача.

Таблиця TVSeriesList має поля:

- id – номер;
- tvseriesId – номер серіалу, який входить до переліку користувача;
- userId – номер облікового запису користувача, для якого формується перелік споріднених користувачів;
- score – середня оцінка, яку було виставлено серіалу користувачами, які входять до переліку з номером listId;
- reviews – кількість рецензій з оцінками, на основі яких було обчислено середню оцінку.

На рис. 2.4 представлена структура БД.

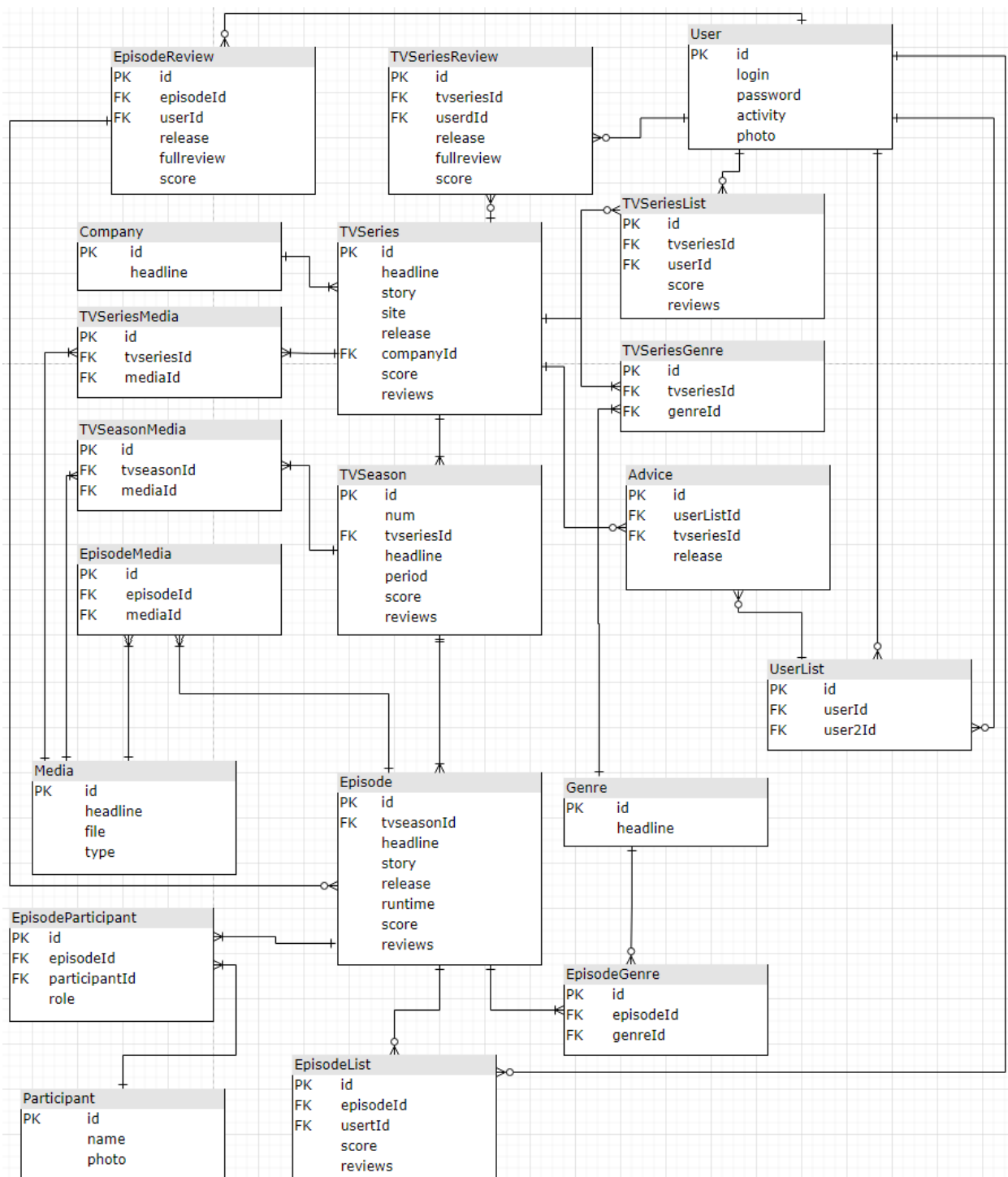


Рисунок 2.4 – Структура БД

Аналогічно таблиці TVSeriesList введено таблицю EpisodeList для узагальнення даних про серію телесеріалу за допомогою полів:

– id – номер;

- episodeId – номер серії, яка входить до переліку користувача;
- userId – номер користувача;
- score – середня оцінка, яку було виставлено даній серії користувачами, які входять до переліку користувача з номером userId;
- reviews – кількість рецензій з оцінками, на основі яких було обчислено середню оцінку.

2.4 Висновки за розділом 2

Визначено функціональні вимоги до програми, виділено сценарії роботи з програмою неавторизованим і авторизованим користувачами та адміністратором, виконано вибір мови Python для розробки програми, описано структуру даних у БД, обрано MySQL в якості системи керування БД.

3 РОЗРОБКА ПРОГРАМИ

3.1 Алгоритм функціонування програмного забезпечення

Програма може працювати або в стандартному режимі, або в режимі адміністратора. Режим адміністратора виражається адміністративною панеллю, з якої можна додавати і редагувати дані телесеріалів, їх сезонів, серій, жанрів, акторів, режисерів, продюсерів.

В стандартному режимі з програмою взаємодіє звичайний користувач.

Програма розпочинає взаємодію з користувачем з виведення за замовчуванням рейтингу телесеріалів (рис. 3.1). Після цього користувач за допомогою меню може обрати наступну групу функціональних можливостей, що включають:

- відображення всіх телесеріалів, що дозволяє переходити до конкретного обраного телесеріалу і переглядати про нього повну інформацію, рецензії;
- пошук, що включає загальний пошук, пошук телесеріалів, пошук сезонів телесеріалів, пошук серій телесеріалів, пошук акторів, пошук режисерів, пошук продюсерів;
- виведення рейтингів, що охоплює можливість виведення рейтингу телесеріалів, сезонів, серій;
- перегляд телесеріалів за жанрами.

Окрім того користувач може авторизуватися і додатково отримати доступ до наступних можливостей:

- робота над переліком споріднених користувачів, що дозволяє редагувати перелік користувачів, переглядати перелік рецензій;
- робота над порадами, що включає перегляд отриманих порад, а також надісланих порад.

Після авторизації виведення рейтингів має можливість виведення рейтингу телесеріалів від споріднених користувачів.

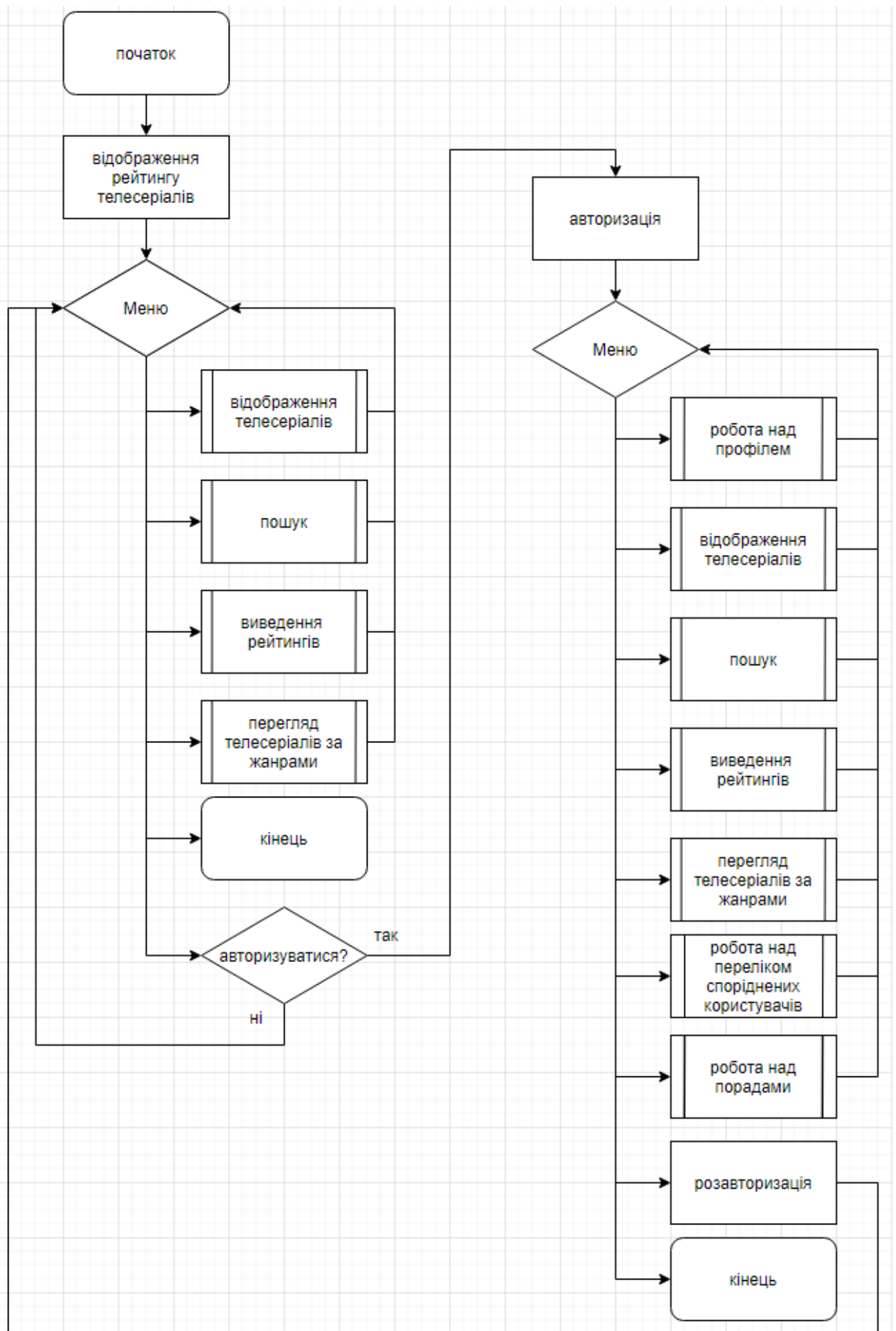


Рисунок 3.1 – Схема функціонування програми

3.2 Структура розробленого програмного забезпечення

Програмне забезпечення представлено програмним проєктом TVSeriesReviewer. До його основного складу (рис. 3.2) входять папки і файли:

- urls.py – менеджер URL;
- view – папка з представленнями застосунку;
- models.py – моделі застосунку;
- template – шаблони застосунку.

Усі представлення застосунку розділені на папки:

- listviews.py – представлення переліків;
- createviews.py – представлення створення записів;
- updateviews.py – представлення оновлення записів;
- detailsviews.py – представлення перегляду детальної інформації.

Шаблони представлені наступними групами:

- tvseries – шаблони, що стосуються телесеріалів: перегляду рецензій на телесеріал reviews.html, перегляду переляку всіх телесеріалів index.html, перегляду рейтингу телесеріалів rating.html, пошуку телесеріалів search.html, пошуку серед всіх даних searchall.html, перегляду телесеріалів за жанром genre.html, створення рецензії на телесеріал reviewadd.html, перегляду даних телесеріалу tvs.html;

- episode – шаблони, що стосуються серій телесеріалів;
- season – шаблони, що стосуються сезонів телесеріалів;
- ulist – шаблони, що стосуються переліків споріднених користувачів;
- part – шаблони, що стосуються виконавців;
- advice – шаблони, що стосуються порад для перегляду телесеріалів: створення нової поради adviceadd.html, перегляду порад, адресованих користувачу, tome.html, перегляду порад від користувача fromme.html.

Шаблони сторінок створено на основі безкоштовного шаблону [9].

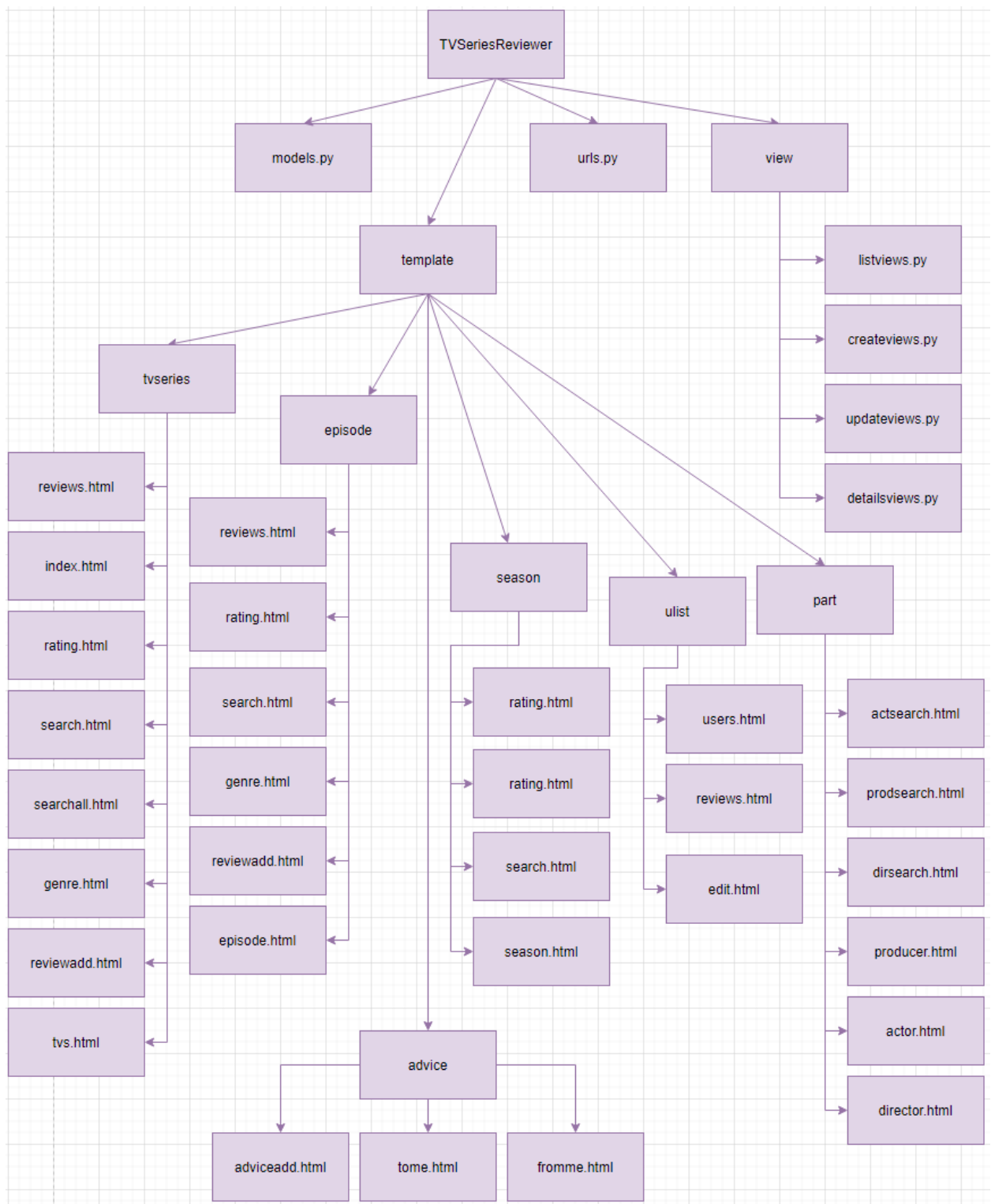


Рисунок 3.2 – Структура розробленої програми

3.3 Опис прийнятих під час реалізації рішень

Взаємодія з БД була виконана шляхом створення коду мовою Python, тобто запити мовою SQL до БД не виконувались вручну, а створювались за

допомогою об'єктно-реляційного відображення Django [10].

Класи моделей створені на основі наслідування класу `models.Model`.

Розроблені класи моделей:

- клас `TVSeries`, що представляє телесеріал і містить всередині набір жанрів та медіафайлів цього телесеріалу;
- клас `TVSeason`, що представляє сезон телесеріалу і містить перелік його медіафайлів;
- клас `Episode`, що представляє серію телесеріалу і містить перелік її жанрів та медіафайлів;
- клас `User`, що представляє користувача;
- клас `EpisodeReview`, що представляє рецензію на серію телесеріалу;
- клас `TVSeriesReview`, що представляє рецензію на телесеріал;
- клас `Company`, що представляє компанію-виробника телесеріалів;
- клас `Genre`, що представляє жанр телесеріалу;
- клас `Participant`, що представляє виконавця телесеріалу;
- клас `EpisodeParticipant`, що представляє інформацію про участь виконавця в епізодах телесеріалу (містить окрім зовнішніх ключів ще роль);
- клас `Media`, що представляє інформацію про медіафайл;
- клас `UserList`, що представляє інформацію про перелік споріднених користувачів;
- клас `Advice`, що представляє пораду для перегляду телесеріалу;
- клас `TVSeriesList`, що представляє дані телесеріалу, сформовані за переліком споріднених користувачів;
- клас `EpisodeList`, що представляє дані серії телесеріалу, сформовані за переліком споріднених користувачів.

Як видно з наведеного переліку, не всі таблиці БД були реалізовані у вигляді окремих класів, це пов'язано з тим, що завдяки використанню об'єктно-реляційного відображення Django зв'язок типу багато до багатьох автоматично перетворюється на окрему таблицю БД. Це дозволило спростити процес написання коду і працювати в інших класах застосунку,

маючи в моделі одразу перелік об'єктів, що відповідають моделями, пов'язаним зв'язком багато до багатьох з даною. Наприклад, замість того, щоб створювати окремий клас, який відповідає таблиці, що зв'язує телесеріал з жанром, було збережено перелік у структурі класу моделі телесеріалу, що містить відповідні жанри, аналогічно збережено медіафайли, що відповідають даному телесеріалу:

```
genres = models.ManyToManyField(Genre)
medias = models.ManyToManyField(Media)
```

Визначення типу кожного поля реалізується наступним чином:

```
headline = models.CharField(max_length=100)
story = models.TextField()
```

Використано наступні типи полів:

- CharField – символічне поле;
- TextField – поле, що містить довгий текст (зокрема текст рецензії);
- DateField – поле, що містить дату;
- DecimalField – поле, що містить десяткове число фіксованої точності;
- IntegerField – цілочисельне поле;
- SmallIntegerField – цілочисельне поле з малим діапазоном (зокрема ролі виконавці можуть виконувати лише 3 ролі (актор, режисер, продюсер));
- BinaryField – поле, що зберігає значення з заданою кількістю бітів (так зберігається логічне значення true або false);
- ForeignKey – поле, що відповідає зовнішньому ключу.

Для основних класів моделей визначено порядок їх сортування за замовчуванням, а також метод `__str__(self)` для перевантаження роботи з об'єктом даного класу як з рядком.

Усі класи представлень розроблені на основі використання відповідних стандартних класів представлень Django шляхом їх наслідування:

- ListView – представлення для організації відображення переліків (наборів даних): використовується для організації пошуку, відтворення рейтингів, рецензій телесеріалів з переліку споріднених користувачів;
- CreateView – представлення для створення запису даних: використовується для створення рецензій;
- UpdateView – представлення для оновлення даних об'єкта (запису таблиці БД): використовується для редагування переліку споріднених користувачів;
- DetailView – представлення для виведення повної інформації про об'єкт (запис таблиці БД): використовується для виведення на сторінці даних про телесеріал, серію телесеріалу, актора, продюсера, режисера.

Структура представлень була описана в попередньому підрозділі, тому в даному підрозділі приділено увагу процесу реалізації описаних представлень.

Клас TVSerieResView забезпечує формування даних, необхідних для відображення рецензій за телесеріалом. Для цього було визначено модель TVSeriesReview в якості основної моделі для даного представлення, визначено відповідний шаблон для створення вебсторінки та визначено відтворення рецензій по 10 на сторінці:

```
model = TVSeriesReview
template_name = 'tvseries/reviews.html'
paginate_by = 24
```

В основі даного класу перевизначено два методи класу ListView:

- метод `get_queryset(self)`, який забезпечує формування контексту моделі, з якої дані відображаються в представленні;

– метод `get_context_data(self, **kwargs)`, який повертає стандартний контекст моделі, включаючи в нього параметри GET-запиту, зокрема номер телесеріалу, для якого виводиться набір рецензій.

У першому методі реалізується отримання даних з параметра GET-запиту, встановлюючи номер телесеріалу, а потім відбувається отримання всіх об'єктів рецензій, які відповідають цьому номеру телесеріалу, а отримані дані сортуються за датою публікації за віддаленням від сьогоднішнього моменту:

```
tvserie = self.request.GET.get('tvserie')
reviews = TVserieReview.objects.filter(tvseries__id = tvserie).order_by('-release')
```

Клас `EpisodeResView` створює представлення перегляду рецензій до серії телесеріалу.

Клас `AdvicesView` виконує підготовку даних для відповідної вебсторінки перегляду порад, які були надіслані користувачу. Для цього підвантажуються всі поради, визначаються користувачі з переліку споріднених для авторизованого в даний момент користувача, потім відфільтровуються всі поради, які були створені користувачами з даного переліку:

```
advices = super(AdvicesView, self).get_queryset()
me = self.request.user
lists = UserList.objects.filter(user = me)
result = []
for us in lists:
    advices1 = advices.filter(userlist = us)
    result.append(advices1)
result = result.order_by('-release')
```

Клас `UserListResView` створює представлення перегляду рецензій до телесеріалів, створених користувачами з переліку споріднених користувачів для авторизованого користувача.

Клас `UserListView` створює представлення перегляду користувачів, які входять до переліку споріднених користувачів для авторизованого користувача.

Клас `TVSeriesRatingView` створює представлення перегляду рейтингу телесеріалів, що сформований за оцінками всіх користувачів шляхом виділення перших 100 найкращих:

```
tvseries = super(TVSeriesRatingView, self).get_queryset()
tvseries = tvseries.order_by('-score')[0:100]
```

Класи `SeasonsRatingView` і `EpisodesRatingView` формують відповідні рейтинги для сезонів телесеріалів та їх серій. Клас `ULTVSeriesRatingView` забезпечує створення представлення за рейтингом телесеріалів, сформованим за оцінками користувачів переліку споріднених користувачів для авторизованого користувача.

Клас `TVSearchView` виконує підготовку переліку даних за результатами пошуку, при чому пошук реалізується і серед телесеріалів, і серед серій телесеріалів, сезонів, а також виконавців. Для цього в отриманий словник результатів додається під відповідними ключами набір даних, що включає результати пошуку серед серій, сезонів тощо:

```
tvseries = super(TVSearchView, self).get_queryset()
tvseries = tvseries.filter(headline__contains=svalue.lower()).order_by(orderway)
tvseries['episodes'] = Episode.filter(headline__contains=svalue.lower()).order_by(orderway)
```

Сортування даних реалізується за алфавітним порядком за назвою або за датою публікації, або за оцінкою. Варіант сортування передається через параметр `orderby` GET-запиту.

Клас `GenreEpisodesView` створює представлення перегляду серій телесеріалів, що відповідають заданому через GET-запит жанру.

Клас `GenreTVSeriesView` створює представлення перегляду телесеріалів, що відповідають заданому через GET-запит жанру.

Клас `UserListEditView` забезпечує представлення редагування переліку споріднених користувачів.

Клас `TVSerieView` створює представлення перегляду інформації про телесеріал, включаючи перелік сезонів телесеріалу, деякі рецензії, а також перелік основних акторів.

Клас `SeasonView` створює представлення перегляду інформації про сезон телесеріалу.

Клас `EpisodeView` створює представлення перегляду даних серії телесеріалу, включаючи перелік медіафайлів, жанрів, акторів.

Клас `ActorView` створює представлення перегляду даних актора. Клас `ProducerView` створює представлення перегляду даних продюсера. Клас `DirectorView` створює представлення перегляду даних режисера.

Клас `EpisodeReviewCreateView` формує представлення створення рецензії на серію телесеріалу.

Клас `TVSeriesReviewCreateView` формує представлення створення рецензії на телесеріал.

Клас `AdviceCreateView` формує представлення створення поради для перегляду телесеріалу.

3.4 Висновки за розділом 3

Розглянуто результати, отримані в процесі реалізації програмного забезпечення у вигляді вебзастосунку: описано алгоритм функціонування

програми, побудовано схему функціонування програми, описано структуру програми і представлено структурну схему програми, описано класи, що були розроблені під час реалізації програми, а також особливості такої реалізації.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення програми

Програма призначена для отримання інформації про телесеріали, створення і перегляду рецензій за переглянутими телесеріалами, отримання даних про середні оцінки телесеріалів, сезонів телесеріалів, серій телесеріалів. Окрім того програма надає можливість узагальнювати результати оцінювання телесеріалів за групами споріднених користувачів.

4.2 Умови виконання програми

Серверна частина має відповідати наступним мінімальним вимогам:

- вільний простір на жорсткому диску – 1 Гб, що включає простір, необхідний для встановлення системи управління БД MySQL, фреймворку Django та самого застосунку, для збереження медіафайлів на власному сервері вимагається додатковий простір в обсягу, достатньому для розміщення всіх цих файлів;
- оперативна пам'ять – 4 Гб;
- процесор: 4 ядра, 2,8 ГГц частота;
- підключення до мережі інтернет.

4.3 Виконання програми

Спочатку потрібно запустити вебсервер, а далі користувачі можуть через браузер звертатися за адресою вебсервера.

За першого звернення користувач працює як неавторизований і йому відображається на основній сторінці рейтинг телесеріалів. Далі він може обрати потрібний йому варіант меню.

Всього йому доступні 4 пункти:

- «Телесеріали», де можна переглянути перелік всіх телесеріалів;
- «Жанри», де можна відобразити перелік всіх жанрів і перейти до перегляду телесеріалів за цим жанром;
- «Пошук», що виконує пошук даних і є пунктом, який можна розкривати;
- «Рейтинг», що відображає рейтинг телесеріалів і містить набір пунктів.

Пошук включає наступні варіанти підменю:

- «Всюди»;
- «Телесеріали»;
- «Сезони»;
- «Серії»;
- «Актори»;
- «Режисери»;
- «Продюсери».

Ці підпункти дозволяють виконувати пошук або у всіх варіантах одразу, або у відповідному варіанті даних.

У варіанті підменю «Всюди» потрібно, як і в інших пунктах, спочатку ввести в текстовий рядок потрібний текст для пошуку в назві (рис. 4.1), обрати напрямок сортування (за оцінкою, за алфавітом або за датою створення) і натиснути на кнопку «Шукати».

У результаті на сторінці послідовно будуть виведені всі знайдені телесеріали, актори, режисери, сезони телесеріалів, серії та продюсери.

Виведення дозволяє натиснути на назву, наприклад, телесеріалу і перейти до перегляду його повних даних.

Пункт меню «Рейтинг» містить підпункти «Телесеріали», «Сезони» та «Серії» для відображення 100 перших за загальним рейтингом відповідно телесеріалів, сезонів або серій телесеріалів.

Авторизованому користувачу після авторизації доступні також пункти «Користувачі», «Поради», «Профіль» та «Вийти». Пункт «Профіль» дозволяє змінювати власні особисті дані, включаючи логін, пароль та фотографію.

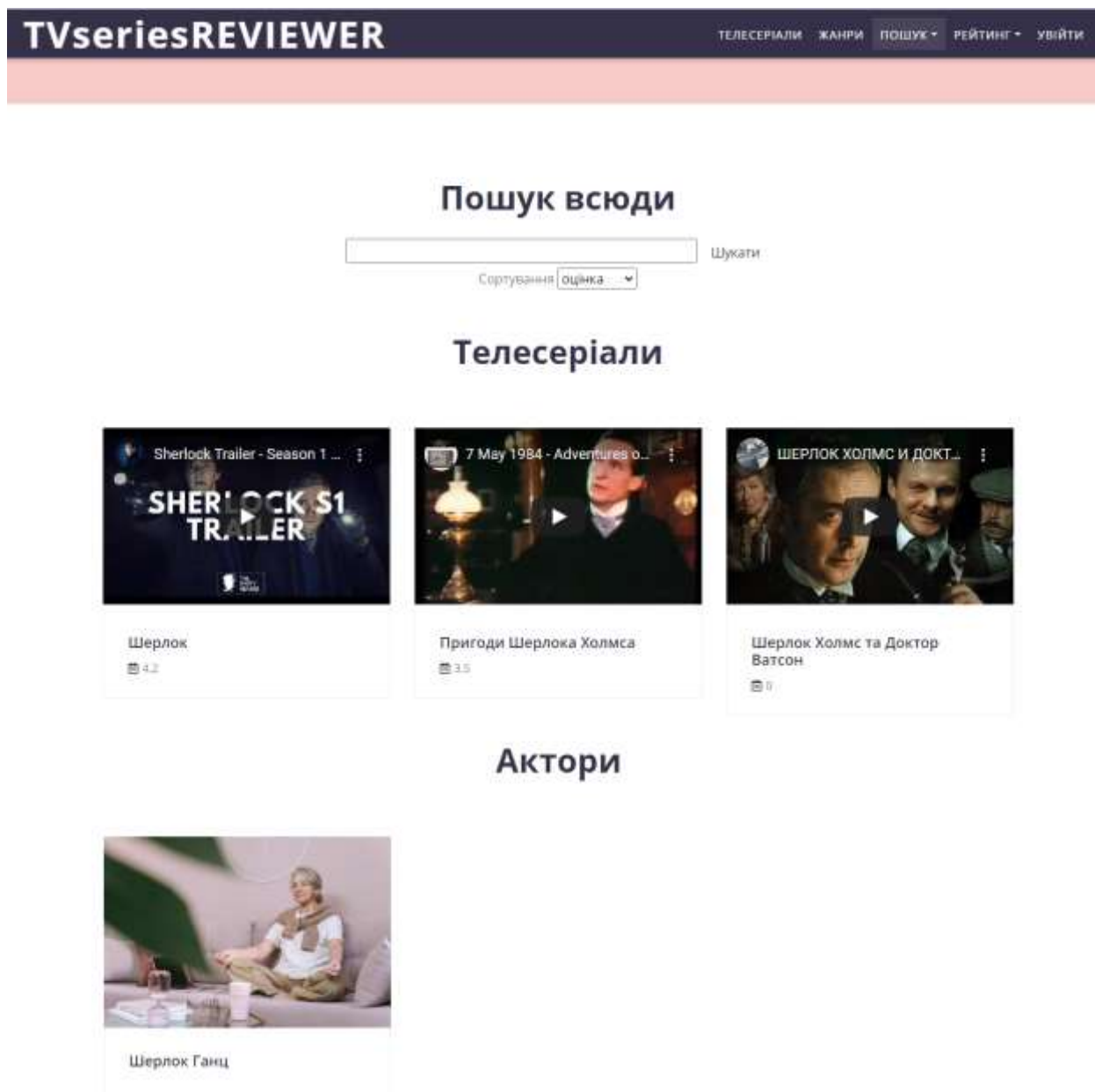


Рисунок 4.1 – Пошук за всіма основними сутностями

На сторінці перегляду телесеріалу, на яку можна потрапити практично з будь-якого іншого пункту меню, відображається повна інформація про телесеріал (рис. 4.2).

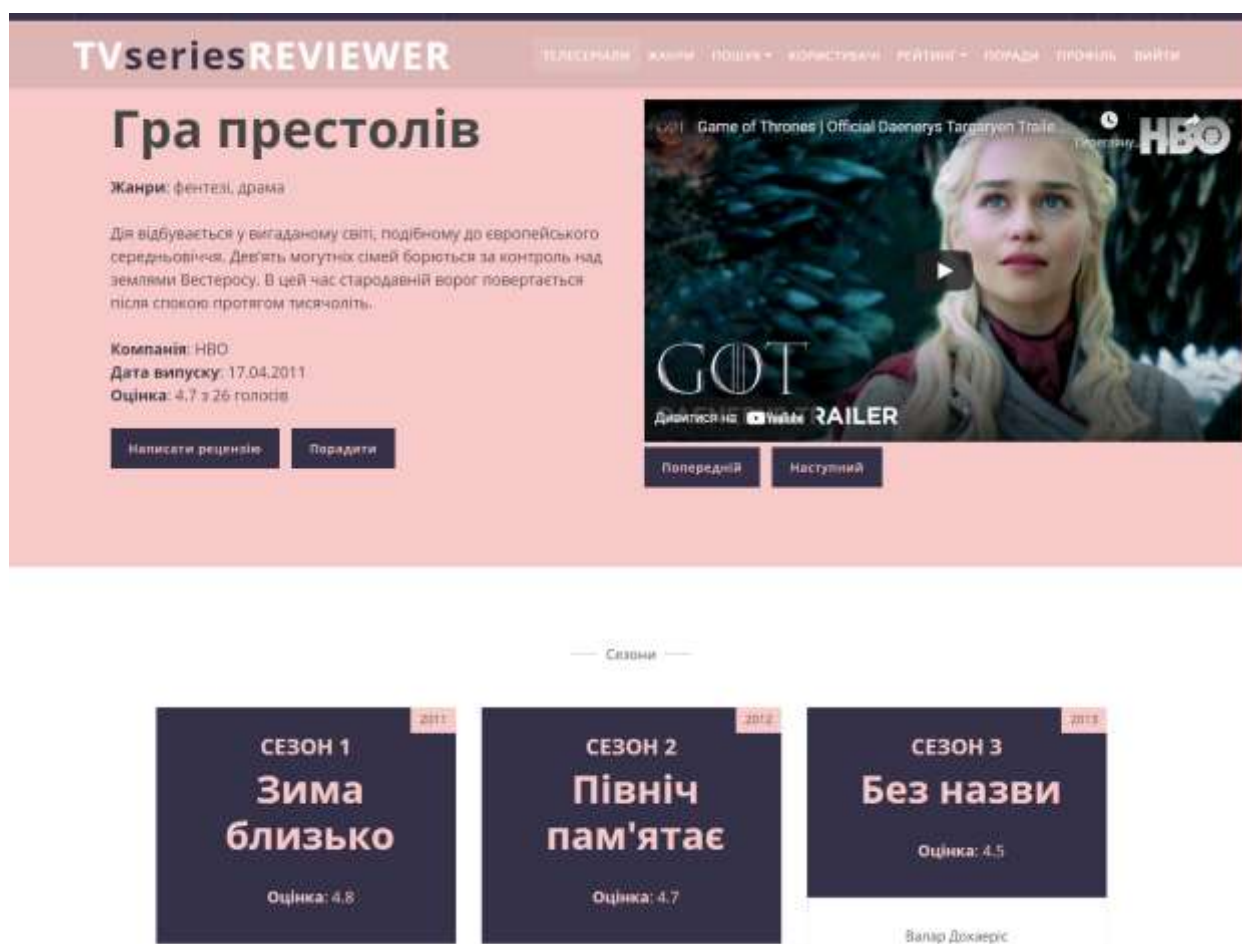


Рисунок 4.2 – Вебсторінка телесеріалу

Повна інформація включає назву телесеріалу (при натисканні на неї можна перейти на зовнішній вебсайт телесеріалу), перелік його жанрів (можна натиснути на будь-який з них та перейти до перегляду всіх фільмів цього жанру), опис телесеріалу, компанію, яка його виробляє, дату першого випуску, загальну оцінку за всіма користувача та кількість таких оцінок. Окрім того сюди входять перелік сезонів, набір акторів, режисерів, продюсерів, перелік рецензій, який є скороченим варіантом і дозволяє перейти на сторінку всіх рецензій.

В правій частині сторінки відображається набір медіафайлів, пов'язаних з телесеріалом. Переміщатися між ними можна за допомогою кнопок «Попередній» і «Наступний».

За допомогою кнопки «Написати рецензію» можна перейти до створення рецензії до телесеріалу. За допомогою кнопки «Порадити» можна

створити пораду користувачу для перегляду даного телесеріалу. Для цього відкривається окрема сторінка, де потрібно обрати, користувача, якому надійде порада. Обирати можна тільки серед користувачів, які раніше додали користувача до своїх переліків споріднених користувачів.

Перелік споріднених користувачів (рис. 4.3) відкривається при виборі пункту меню «Користувачі». У переліку відображаються всі користувачі, які були раніше додані до даного переліку авторизованим користувачем. Для кожного користувача можна або натиснути на кнопку з зображенням мінусу і таким чином видалити користувача з переліку, або натиснути на його логін та перейти до сторінки профілю користувача, на якій відображаються всі його рецензії. Це дозволяє отримувати інформацію про смаки користувача і за необхідності додавати його до переліку власних споріднених користувачів.

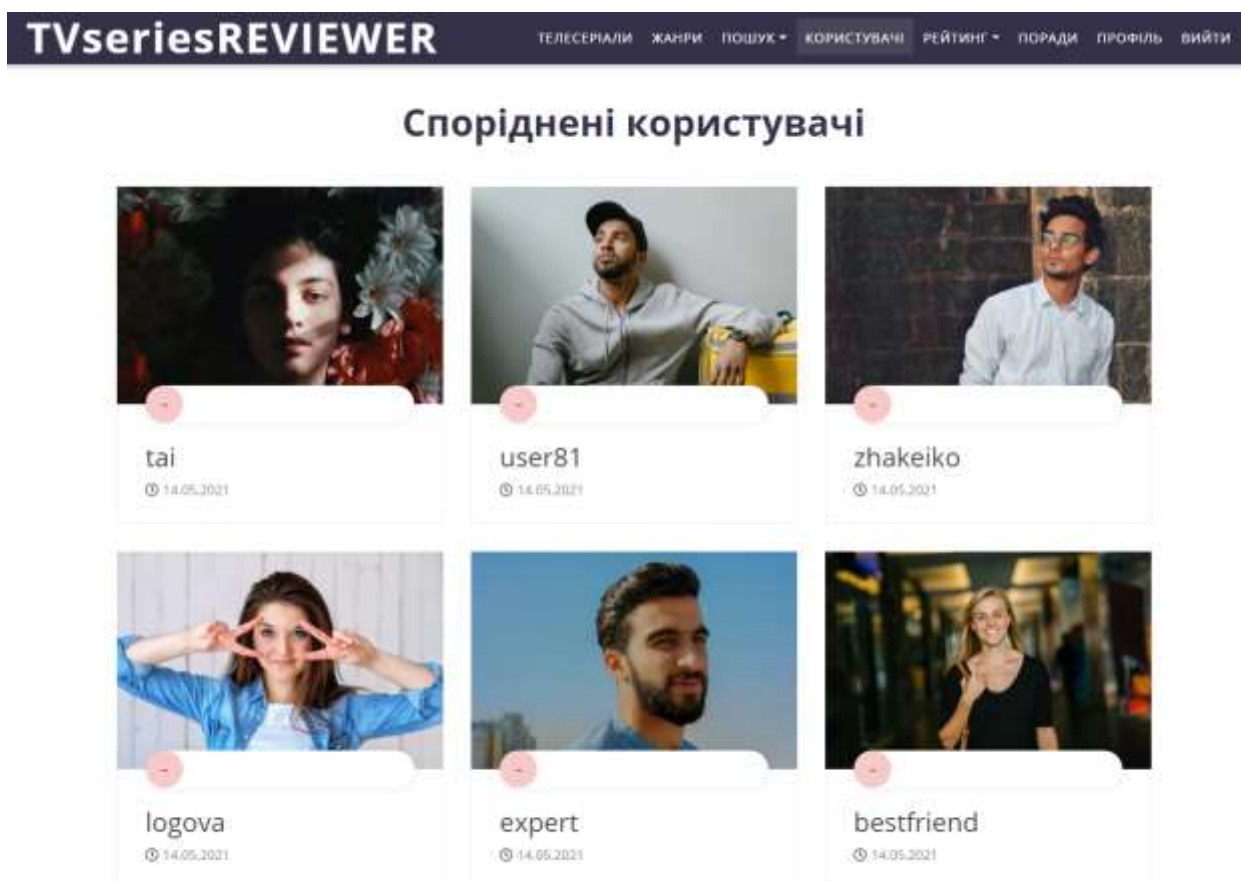


Рисунок 4.3 – Вебсторінка переліку споріднених користувачів

Пункт меню «Рейтинг» містить для авторизованого користувача

також підпункт «Споріднені» (рис. 4.4), який дозволяє відобразити рейтинг телесеріалів за оцінками, які їм були виставлені тільки спорідненими користувачами. За кожним телесеріалом відображається його перший медіафайл, назва та оцінка. За назвою можна перейти до сторінки телесеріалу.

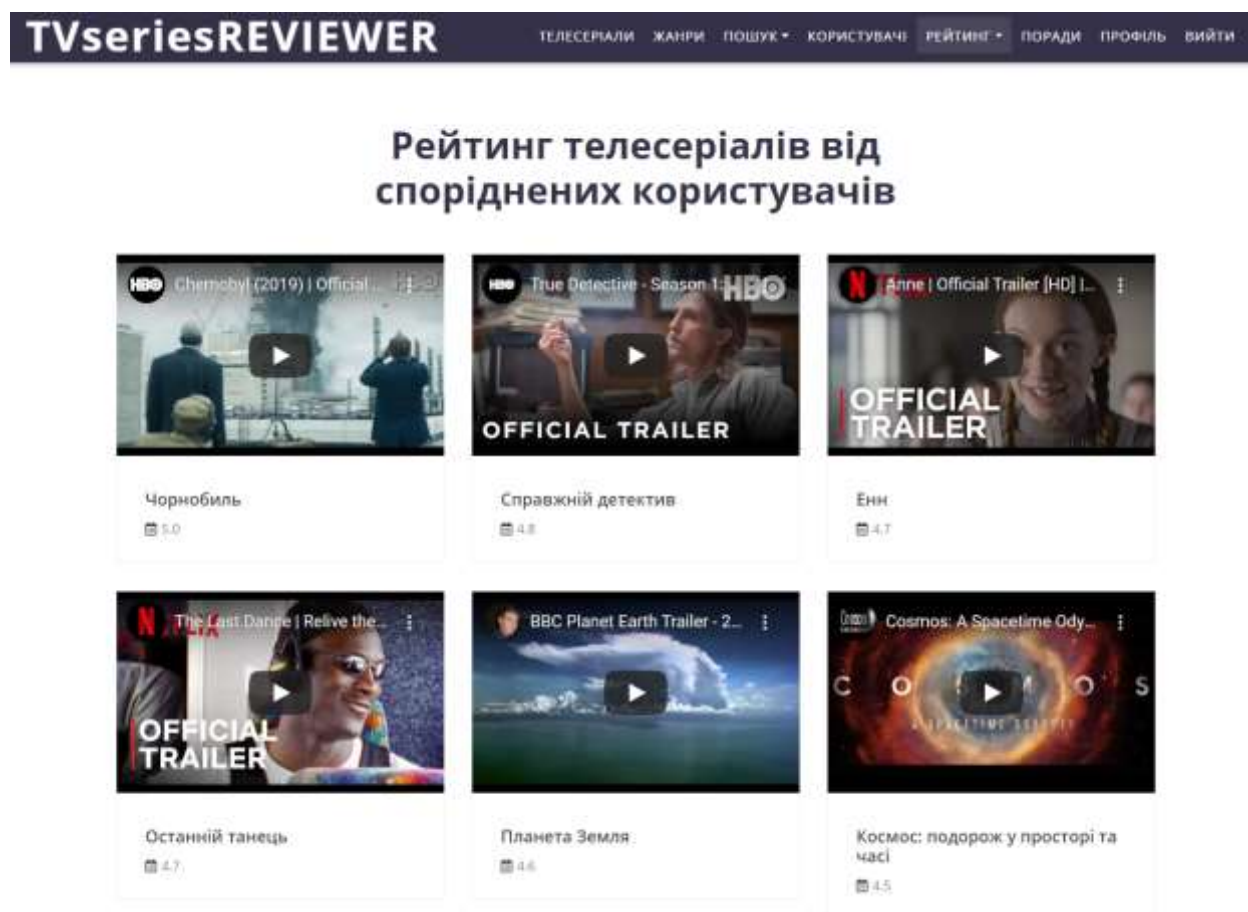


Рисунок 4.4 – Вебсторінка рейтинга телесеріалів від споріднених користувачів

Пункт меню «Поради» відкриває вебсторінку, на якій можна переглянути поради, адресовані користувачу або створені ним (рис. 4.5). Для переключення між цими варіантами відображення використовуються кнопки «Для мене» для відображення адресованих користувачу порад та «Мої» для відображення порад, які раніше створив авторизований користувач.

За кожною порадою відображається перший медіафайл телесеріалу, його назва, за якою можна перейти до сторінки телесеріалу, дата створення

поради. Також демонструється те, хто цю пораду створив. Якщо користувач вважає таку пораду неприйнятною і більше не хоче отримувати поради від даного користувача, то він може натиснути на знак мінус в рядку логіна користувача і вилучити користувача з власного переліку споріднених.

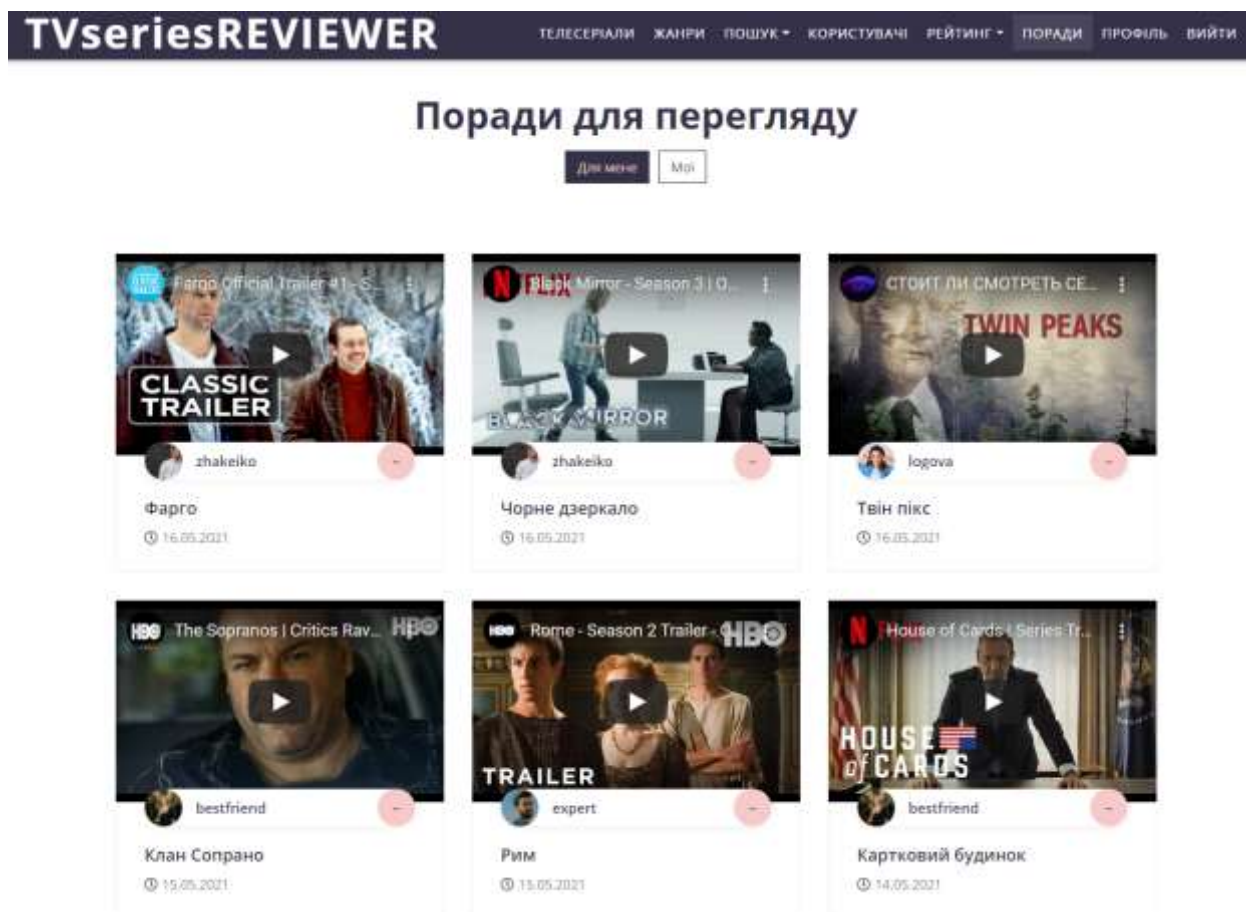


Рисунок 4.5 – Сторінка порад для перегляду

Якщо на сторінці перегляду детальної інформації про телесеріал в секції рецензій натиснути на кнопку «Повністю», то відкриється окрема сторінка, на якій будуть представлені всі рецензії, написані користувачами на даний телесеріал (рис. 4.6).

Для кожної рецензії відображається її автор, при чому фотографія профілю використовується як основне зображення рецензії. Внизу кожної фотографії відображається логін користувача. Якщо натиснути на нього, то можна перейти до профілю користувача, де будуть зокрема і відображені всі рецензії користувача, як на телесеріали, так і на серії телесеріалів.

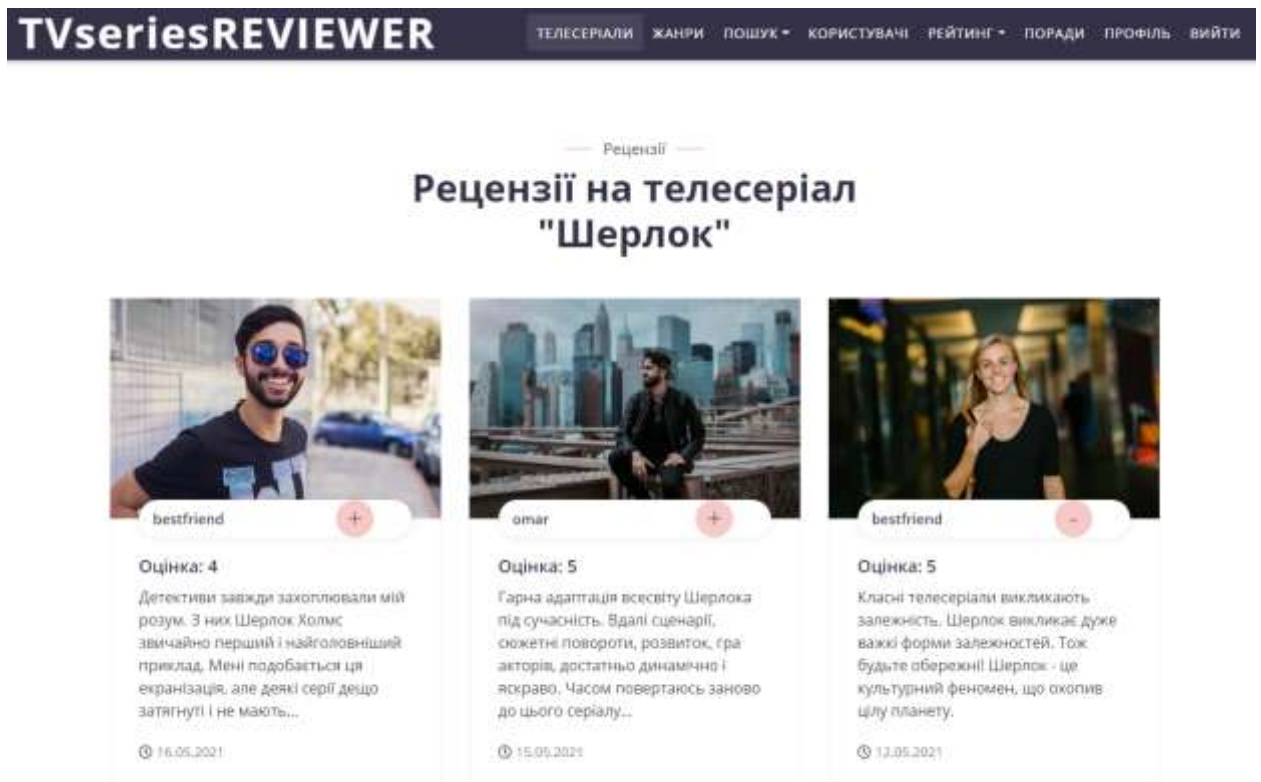


Рисунок 4.6 – Перегляд рецензій на телесеріал

Біля логіну кожного користувача відображається кнопка, яка має вигляд мінуса або плюса. Якщо кнопка представлена плюсом, то за натискання на неї цей користувач буде доданий до переліку споріднених користувачів. За натискання на кнопку мінус користувач навпаки буде вилучений з такого переліку. Тобто якщо користувач вже присутній у переліку, то йому відображається кнопка мінус.

На сторінці розташований перелік усіх рецензій, відсортований за датою публікації. Для кожної рецензії внизу відображається дата її публікації, а також оцінка і її текст. На цій сторінці текст рецензії обмежується до 175 символів. Щоб переглянути текст рецензії повністю, потрібно натиснути на її текст, тоді відкриється окрема сторінка, де будуть представлені всі дані рецензії.

Аналогічний вигляд має сторінка перегляду рецензій на серію телесеріалу.

Таким чином, авторизований користувач отримує можливість працювати з іншими користувачами, виділяючи для себе потрібну групу таких користувачів. Це призводить до виділення потрібних рецензій, оцінок на телесеріали, для чого потрібно тільки визначити учасників групи споріднених людей.

ВИСНОВКИ

У роботі розглянуто проблему розроблення програмного забезпечення рецензування телесеріалів. Проаналізовано існуючі програмні рішення щодо їх функціональних можливостей для рецензування телесеріалів, виконано порівняння їх між собою та з програмою, що розробляється. Основною особливістю програми, розробленої в роботі, є створення переліків споріднених користувачів, що дозволять виділити зі всіх оцінок і рецензій користувачів саме ті, думці яких довіряє користувач.

Під час проєктування програми визначено функціональні вимоги до програми, виділено сценарії роботи з програмою неавторизованим і авторизованим користувачами та адміністратором, виконано вибір мови Python для розробки програми, описано структуру даних у БД, обрано MySQL в якості системи керування БД.

Розглянуто результати, отримані в процесі реалізації програмного забезпечення у вигляді вебзастосунку: описано алгоритм функціонування програми, побудовано схему функціонування програми, описано структуру програми і представлено структурну схему програми, описано класи, що були розроблені під час реалізації програми, а також особливості такої реалізації.

Розроблено програмне забезпечення рецензування телесеріалів, що надає можливість отримувати інформацію про телесеріали, виконувати пошук телесеріалів, їх сезонів, серій, акторів, що беруть у них участь, режисерів, продюсерів, створювати і переглядати рецензії за телесеріалами, отримувати дані про середні оцінки телесеріалів, сезонів телесеріалів, серій телесеріалів, узагальнювати результати оцінювання телесеріалів за групами споріднених користувачів, надавати користувачами з переліку споріднених рекомендації для перегляду.

Завдання дипломної кваліфікаційної роботи в результаті повністю виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. IMDb : Ratings, Reviews, and Where to Watch the Best Movies & TV Shows [Electronic resource]. – Access mode : <https://www.imdb.com/>.
2. 7 IMDb Features You May Have Overlooked [Electronic resource]. – Access mode : <https://www.makeuseof.com/tag/useful-imdb-features/>.
3. IMDb TV Review [Electronic resource]. – Access mode : <https://www.reviews.com/entertainment/streaming/imdb-tv-review/>.
4. Shibu, S. IMDb TV Review [Electronic resource] / S. Shibu. – Access mode : <https://www.pcmag.com/reviews/imdb-tv>.
5. The Movie Database (TMDb) [Electronic resource]. – Access mode : <https://www.themoviedb.org/>.
6. Марголін, О. UML для бізнес-моделювання : для чого потрібні діаграми процесів [Електрон. ресурс] / О. Марголін. – Режим доступу : <https://evergreens.com.ua/ua/articles/uml-diagrams.html>.
7. Lubanovic, B. Introducing Python : Modern Computing in Simple Packages [Text] / B. Lubanovic. – Sebastopol, CA : O'Reilly Media, 2019. – 630 p.
8. Васильев, А. Программирование на Python в примерах и задачах [Текст] / А. Васильев. – М. : Бомбора, 2020. – 616 с.
9. Yoga Free Website Template [Electronic resource]. – Access mode : <https://www.free-css.com/free-css-templates/page170/yoga>.
10. The Web framework for perfectionists with deadlines [Electronic resource]. – Access mode : <https://www.djangoproject.com/>.

ДОДАТОК А
Технічне завдання

Вступ

Бази даних про кінематограф відомі і достатньо популярні. При чому їх традиційна функціональність також давно відома і потребує певного оновлення для того, щоб сучасні користувачі мали додаткову цікавість відносно використання таких програм окрім просто отримання інформації про телесеріали.

A.1 Підстави для розробки

Підставою для розробки було завдання на дипломну кваліфікаційну роботу за темою «Розробка програмного забезпечення рецензування телесеріалів» на основі наказу № 103 від 30 березня 2021 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Розробка призначена для отримання інформації про телесеріали, створення і перегляду рецензій за переглянутими телесеріалами, отримання даних про середні оцінки телесеріалів, сезонів телесеріалів, серій телесеріалів. Окрім того програма повинна надавати можливість узагальнювати результати оцінювання телесеріалів за групами споріднених користувачів.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Програма повинна забезпечувати наступні функціональні характеристики:

- внесення телесеріалів, їх сезонів та серій;

- внесення акторів, режисерів, продюсерів, компаній-виробників, жанрів;
- керування внесеними телесеріалами, їх сезонами, серіями, акторами, режисерами, продюсерами, компаніями-виробниками, жанрами;
- створення рецензій на телесеріали;
- створення рецензій на окремі серії телесеріалів;
- обчислення середньої оцінки телесеріалу за оцінками рецензій на нього;
- обчислення середньої оцінки сезону телесеріалу на основі середніх оцінок серій телесеріалу;
- обчислення середньої оцінки серії телесеріалу на основі рецензій;
- перегляд даних телесеріалу;
- перегляд даних сезонів телесеріалу;
- перегляд даних серій сезону телесеріалу;
- перегляд інформації про акторів, включаючи перелік їх серіалів;
- перегляд інформації про режисерів, включаючи перелік їх серіалів;
- перегляд інформації про продюсерів, включаючи перелік їх серіалів;
- пошук телесеріалів, їх сезонів та окремих серій за їх назвою;
- пошук режисерів, акторів, продюсерів;
- перегляд рецензій на телесеріал;
- перегляд рецензій на серію телесеріалу;
- додання користувачів до власного переліку споріднених користувачів;
- вилучення користувачів з власного переліку споріднених користувачів;
- перегляд рецензій споріднених користувачів;
- перегляд рейтингу телесеріалів;
- перегляд рейтингу серій телесеріалу;
- перегляд рейтингу сезонів телесеріалу;

- перегляд рейтингу телесеріалів за рецензіями споріднених користувачів;
- відправлення поради для перегляду телесеріалу користувачу;
- реєстрація користувачів;
- авторизація користувачів.

A.3.2 Вимоги до надійності

Надійність роботи програми має забезпечуватися надійною роботою вебсервера, системи керування БД та обробленням даних, отриманих від користувачів для уникнення некоректностей та спотворення даних, що вже зберігаються в БД.

A.3.3 Умови експлуатації

Програма має експлуатуватися шляхом розміщення вебзастосунку на сервері. Для підтримки роботи програми та системи керування базами даних має бути виділено системного адміністратора.

A.3.4 Вимоги до складу та параметрів технічних засобів

Серверна частина має відповідати наступним мінімальним вимогам:

- вільний простір на жорсткому диску – 1 Гб, що включає простір, необхідний для встановлення системи управління БД MySQL, фреймворку Django та самого застосунку, для збереження медіафайлів на власному сервері вимагається додатковий простір в обсягу, достатньому для розміщення всіх цих файлів;
- оперативна пам'ять – 4 Гб;
- процесор: 4 ядра, 2,8 ГГц частота;
- підключення до мережі інтернет.

A.4 Порядок контролю та приймання

Контроль за виконанням дипломної кваліфікаційної роботи покладається на керівника. Остаточне рішення приймається екзаменаційною комісією під час захисту роботи.

ДОДАТОК Б
Текст програми

Б.1 Текст файла models.py

```
from django.db import models
```

```
class TVSeries (models.Model):
    headline = models.CharField(max_length=100)
    story = models.TextField()
    site = models.CharField(max_length=255)
    release = models.DateField()
    company = ForeignKey (Company, on_delete=models.CASCADE)
    score = models.DecimalField(decimal_places=1)
    reviews = models.IntegerField()
    genres = models.ManyToManyField(Genre)
    medias = models.ManyToManyField(Media)

    def __str__(self):
        return self.headline

    class Meta:
        ordering = ['-score']

class TVSeason (models.Model):
    num = models.IntegerField (max_length=255)
    tvseries = ForeignKey (TVSeries, on_delete=models.CASCADE)
    headline = models.CharField(max_length=100)
    period = models.CharField(max_length=20)
    score = models.DecimalField(decimal_places=1)
    reviews = models.IntegerField()
    medias = models.ManyToManyField(Media)

    def __str__(self):
        if self.period:
            return self.period
        else:
            return 'Без назви'

    class Meta:
        ordering = ['-score']

class Episode (models.Model):
    num = models.IntegerField (max_length=255)
```

```

    tvseason = ForeignKey (TVSeason, on_delete=models.CASCADE)
    headline = models.CharField(max_length=100)
    story = models.TextField()
    release = models.DateField()
    runtime = models.IntegerField()
    score = models.DecimalField(decimal_places=1)
    reviews = models.IntegerField()
    genres = models.ManyToManyField(Genre)
    medias = models.ManyToManyField(Media)

    def __str__(self):
        return self.headline

    class Meta:
        ordering = ['-score']

class User (models.Model):
    login = models.CharField(max_length=30)
    password = models.CharField(max_length=30)
    activity = models.DateField()

    def __str__(self):
        return self.login

class EpisodeReview (models.Model):
    episode = ForeignKey (Episode, on_delete=models.CASCADE)
    user = ForeignKey (User, on_delete=models.CASCADE)
    release = models.DateField()
    fullreview = models.TextField()
    score = models.IntegerField ()

class TVSeriesReview (models.Model):
    tvseries = ForeignKey (TVSeries, on_delete=models.CASCADE)
    user = ForeignKey (User, on_delete=models.CASCADE)
    release = models.DateField()
    fullreview = models.TextField()
    score = models.IntegerField ()

class Company (models.Model):
    headline = models.CharField(max_length=100)

```

```

def __str__(self):
    return self.headline

class Meta:
    ordering = ['headline']

class Genre (models.Model):
    headline = models.CharField(max_length=100)

    def __str__(self):
        return self.headline

    class Meta:
        ordering = ['headline']

class Participant (models.Model):
    name = models.CharField(max_length=100)
    photo = models.CharField(max_length=255)

    def __str__(name):
        return self.headline

    class Meta:
        ordering = ['name']

class EpisodeParticipant (models.Model):
    episode = ForeignKey (Episode, on_delete=models.CASCADE)
    participant      =      ForeignKey      (Participant,
on_delete=models.CASCADE)
    role = models.SmallIntegerField ()

class Media (models.Model):
    headline = models.CharField(max_length=100)
    file = models.CharField(max_length=255)
    type = models.BinaryField(max_length=1)

class UserList (models.Model):
    user = ForeignKey (User, on_delete=models.CASCADE)
    user2 = ForeignKey (User, on_delete=models.CASCADE)

```

```

class Advice (models.Model):
    tvseries = ForeignKey (TVSeries, on_delete=models.CASCADE)
    userlist = ForeignKey (UserList, on_delete=models.CASCADE)
    release = models.DateField()

    class Meta:
        ordering = ['-release']

class TVSeriesList (models.Model):
    tvseries = ForeignKey (TVSeries, on_delete=models.CASCADE)
    user = ForeignKey (User, on_delete=models.CASCADE)
    score = models.DecimalField(decimal_places=1)
    reviews = models.IntegerField()

class EpisodeList (models.Model):
    episode = ForeignKey (Episode, on_delete=models.CASCADE)
    user = ForeignKey (User, on_delete=models.CASCADE)
    score = models.DecimalField(decimal_places=1)
    reviews = models.IntegerField()

```

Б.2 Текст файла listviews.py

```

from django.views.generic import ListView
from django.contrib.auth.mixins import LoginRequiredMixin
from .models import TVSeriesReview
from .models import EpisodeReview
from .models import Advice
from .models import UserList
from .models import TVSeries
from .models import TVSeason
from .models import Episode
from .models import TVSeriesList
from .models import Participant

class TVSerieResView (ListView):
    model = TVSeriesReview
    template_name = 'tvseries/reviews.html'
    paginate_by = 24

    def get_queryset(self):

```

```

        tvserie = self.request.GET.get('tvserie')
        reviews = TVserieReview.objects.filter(tvseries__id =
tvserie).order_by('-release')
        return reviews

```

```

    def get_context_data(self, **kwargs):
        reviews = super(TVserieResView,
self).get_context_data(**kwargs)
        reviews ['tvserie'] = self.request.GET.get('tvserie')
        return reviews

```

```

class EpisodeResView (ListView):
    model = EpisodeReview
    template_name = 'episode/reviews.html'
    paginate_by = 24

```

```

    def get_queryset(self):
        episode = self.request.GET.get('episode')
        reviews = TVserieReview.objects.filter(
            episode__id = episode
        ).order_by('-release')
        return reviews

```

```

    def get_context_data(self, **kwargs):
        reviews = super(EpisodeResView,
self).get_context_data(**kwargs)
        reviews ['episode'] = self.request.GET.get('episode')
        return reviews

```

```

class AdvicesView (LoginRequiredMixin, ListView):
    model = Advice
    template_name = 'advice/tome.html'
    paginate_by = 24

```

```

    def get_queryset(self):
        advices = super(AdvicesView, self).get_queryset()
        me = self.request.user
        lists = UserList.objects.filter(user = me)
        result = []
        for us in lists:
            advices1 = advices.filter(userlist = us)
            result.append(advices1)
        result = result.order_by('-release')

```

```
return result
```

```
class FromAdvicesView (LoginRequiredMixin, ListView):
    model = Advice
    template_name = 'advice/fromme.html'
    paginate_by = 24

    def get_queryset(self):
        advices = super(FromAdvicesView, self).get_queryset()
        me = self.request.user
        lists = UserList.objects.filter(user2 = me)
        result = Advice.objects.filter(id = -1)
        for us in lists:
            advices1 = advices.filter(userlist = us)
            result.append(advices1)
        result = result.order_by('-release')
        return result

class UserListResView (LoginRequiredMixin, ListView):
    model = TVSeriesReview
    template_name = 'ulist/reviews.html'
    paginate_by = 24

    def get_queryset(self):
        tvseries = super(UserListResView, self).get_queryset()
        me = self.request.user
        lists = UserList.objects.filter(user = me)
        result = tvseries.filter(id = -1)
        for us in lists:
            u = us.user2
            rev = tvseries.filter(user = u)
            result.append(rev)
        result = result.order_by('-release')
        return result

class UserListView (LoginRequiredMixin, ListView):
    model = UserList
    template_name = 'ulist/users.html'

    def get_queryset(self):
        users = super(UserListView, self).get_queryset()
        me = self.request.user
```

```

users = users.filter(user = me)
result = []
for us in users:
    u = us.user2
    result.append(u)
result = result.order_by('login')
return result

```

```

class TVSeriesIndexView (ListView):
    model = TVSeries
    template_name = 'tvseries/index.html'
    paginate_by = 24

```

```

class GenresIndexView (ListView):
    model = Genre
    template_name = 'tvseries/genres.html'

```

```

class TVSeriesRatingView (ListView):
    model = TVSeries
    template_name = 'tvseries/rating.html'

    def get_queryset(self):
        tvseries = super(TVSeriesRatingView,
self).get_queryset()
        tvseries = tvseries.order_by('-score')[0:100]
        return tvseries

```

```

class SeasonsRatingView (ListView):
    model = TVSeason
    template_name = 'season/rating.html'

    def get_queryset(self):
        seasons = super(SeasonsRatingView, self).get_queryset()
        seasons = seasons.order_by('-score')[0:100]
        return seasons

```

```

class EpisodesRatingView (ListView):
    model = Episode
    template_name = 'episode/rating.html'

```

```

    def get_queryset(self):
        episodes = super(EpisodesRatingView,
self).get_queryset()
        episodes = episodes.order_by('-score')[0:100]
        return episodes

class ULTVSeriesRatingView (ListView):
    model = TVSeriesList
    template_name = 'ulist/rating.html'

    def get_queryset(self):
        tvseries = super(ULTVSeriesRatingView,
self).get_queryset()
        me = self.request.user
        tvseries = tvseries.filter(user = me)
        tvseries = tvseries.order_by('-score')[0:100]
        return tvseries

class ActorSearchView (ListView):
    model = Participant
    template_name = 'part/actsearch.html'
    paginate_by = 24

    def get_queryset(self):
        actorname = self.request.GET.get('name')
        participants = super(ActorSearchView,
self).get_queryset()
        participants = participants.filter(headline__contains =
name.lower())
        for p in participants:
            actors = EpisodeParticipant.objects.filter(participant = p, role = 0)
            if not actors:
                participants.remove(p)
        return participants

    def get_context_data(self, **kwargs):
        participants = super(ActorSearchView,
self).get_context_data(**kwargs)
        participants ['name'] = self.request.GET.get('name')
        return participants

```

```

class TVSeriesSearchView (ListView):
    model = TVSeries
    template_name = 'tvseries/search.html'
    paginate_by = 24

    def get_queryset(self):
        svalue = self.request.GET.get('svalue')
        orderway = self.request.GET.get('orderway')
        if orderway == 0:
            orderway = 'headline'
        elif orderway == 1:
            orderway = 'release'
        else:
            orderway = 'score'
        tvseries = super(TVSeriesSearchView,
self).get_queryset()
        tvseries = tvseries.filter(headline__contains =
svalue.lower()).order_by(orderway)
        return tvseries

    def get_context_data(self, **kwargs):
        tvseries = super(TVSeriesSearchView,
self).get_context_data(**kwargs)
        tvseries ['svalue'] = self.request.GET.get('svalue')
        tvseries ['orderway'] = self.request.GET.get('orderway')
        return tvseries

class SeasonsSearchView (ListView):
    model = TVSeason
    template_name = 'season/search.html'
    paginate_by = 24

    def get_queryset(self):
        svalue = self.request.GET.get('svalue')
        orderway = self.request.GET.get('orderway')
        if orderway == 0:
            orderway = 'headline'
        elif orderway == 1:
            orderway = 'release'
        else:
            orderway = 'score'
        seasons = super(SeasonsSearchView, self).get_queryset()
        seasons = seasons.filter(headline__contains =
svalue.lower()).order_by(orderway)

```

```

        return seasons

    def get_context_data(self, **kwargs):
        seasons = super(SeasonsSearchView,
self).get_context_data(**kwargs)
        seasons ['svalue'] = self.request.GET.get('svalue')
        seasons ['orderway'] = self.request.GET.get('orderway')
        return seasons

class ProducerSearchView (ListView):
    model = Participant
    template_name = 'part/prodsearch.html'
    paginate_by = 24

    def get_queryset(self):
        actorname = self.request.GET.get('name')
        participants = super(ProducerSearchView,
self).get_queryset()
        participants = participants.filter(headline__contains =
name.lower())
        for p in participants:
            producers = EpisodeParticipant.objects.filter(participant = p, role = 2)
            if not producers:
                participants.remove(p)
        return participants

    def get_context_data(self, **kwargs):
        participants = super(ProducerSearchView,
self).get_context_data(**kwargs)
        participants ['name'] = self.request.GET.get('name')
        return participants

class EpisodesSearchView (ListView):
    model = Episode
    template_name = 'episode/search.html'
    paginate_by = 24

    def get_queryset(self):
        svalue = self.request.GET.get('svalue')
        orderway = self.request.GET.get('orderway')
        if orderway == 0:
            orderway = 'headline'

```

```

        elif orderway == 1:
            orderway = 'release'
        else:
            orderway = 'score'
        episodes = super(EpisodesSearchView,
self).get_queryset()
        episodes = episodes.filter(headline__contains =
svalue.lower()).order_by(orderway)
        return episodes

    def get_context_data(self, **kwargs):
        episodes = super(EpisodesSearchView,
self).get_context_data(**kwargs)
        episodes ['svalue'] = self.request.GET.get('svalue')
        episodes ['orderway'] = self.request.GET.get('orderway')
        return episodes

class DirectorSearchView (ListView):
    model = Participant
    template_name = 'part/dirsearch.html'
    paginate_by = 24

    def get_queryset(self):
        actorname = self.request.GET.get('name')
        participants = super(DirectorSearchView,
self).get_queryset()
        participants = participants.filter(headline__contains =
name.lower())
        for p in participants:
            directors =
EpisodeParticipant.objects.filter(participant = p, role = 1)
            if not directors:
                participants.remove(p)
        return participants

    def get_context_data(self, **kwargs):
        participants = super(DirectorSearchView,
self).get_context_data(**kwargs)
        participants ['name'] = self.request.GET.get('name')
        return participants

```

```

class TVSearchView (ListView):
    model = TVSeries
    template_name = 'tvseries/searchall.html'
    paginate_by = 24

    def get_queryset(self):
        svalue = self.request.GET.get('svalue')
        orderway = self.request.GET.get('orderway')
        if orderway == 0:
            orderway = 'headline'
        elif orderway == 1:
            orderway = 'release'
        else:
            orderway = 'score'
        tvseries = super(TVSearchView, self).get_queryset()
        tvseries = tvseries.filter(headline__contains =
svalue.lower()).order_by(orderway)
        tvseries['episodes'] =
Episode.filter(headline__contains = svalue.lower()).order_by(orderway)
        if orderway != 'release':
            tvseries['seasons'] =
TVSeason.filter(headline__contains =
svalue.lower()).order_by(orderway)
        else:
            tvseries['seasons'] =
TVSeason.filter(headline__contains = svalue.lower()).order_by('score')
            tvseries['participants'] =
Participant.filter(headline__contains = name.lower())
        return tvseries

    def get_context_data(self, **kwargs):
        tvseries = super(TVSearchView,
self).get_context_data(**kwargs)
        tvseries ['svalue'] = self.request.GET.get('svalue')
        tvseries ['orderway'] = self.request.GET.get('orderway')
        return tvseries

class GenreEpisodesView (ListView):
    model = Episode
    template_name = 'episode/genre.html'
    paginate_by = 24

    def get_queryset(self):
        genrename = self.request.GET.get('genre')

```

```

        episodes = super(GenreEpisodesView, self).get_queryset()
        for ep in episodes:
            genreepis = EpisodeGenre.objects.filter(genre =
genrename, episode = ep)
            if not genreepis:
                episodes.remove(ep)
        episodes = episodes.order_by('score')
        return episodes

    def get_context_data(self, **kwargs):
        episodes = super(GenreEpisodesView,
self).get_context_data(**kwargs)
        episodes ['genre'] = self.request.GET.get('genre')
        return episodes

class GenreTVSeriesView (ListView):
    model = TVSeries
    template_name = 'tvseries/genre.html'
    paginate_by = 24
    def get_queryset(self):
        genrename = self.request.GET.get('genre')
        tvseries = super(GenreTVSeriesView, self).get_queryset()
        for tv in tvseries:
            genretv = TVSeriesGenre.objects.filter(genre =
genrename, episode = ep)
            if not genretv:
                tvseries.remove(tv)
        tvseries = tvseries.order_by('score')
        return tvseries

    def get_context_data(self, **kwargs):
        tvseries = super(GenreTVSeriesView,
self).get_context_data(**kwargs)
        tvseries ['genre'] = self.request.GET.get('genre')
        return episodes

```

ДОДАТОК В
Слайди презентації

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ РЕЦЕНЗУВАННЯ ТЕЛЕСЕРІАЛІВ

SOFTWARE DEVELOPMENT FOR TV SERIES REVIEWS

Виконав
студент групи КНТ-127

Д.С. Череватенко

Керівник
к.т.н., доцент

Т.В. Федорончак

2021

Рисунок В.1 – Слайд № 1

Об'єкт дослідження – процес рецензування телесеріалів.

Предмет дослідження – програмне забезпечення рецензування телесеріалів.

Мета роботи – розроблення програмного забезпечення рецензування телесеріалів для підтримки індивідуалізації узагальнених результатів оцінювання та рецензування.

Рисунок В.2 – Слайд № 2

Завдання

- ✓ аналіз предметної області;
- ✓ проєктування програмного забезпечення;
- ✓ реалізація програмного забезпечення;
- ✓ тестування і налагодження програмного забезпечення.

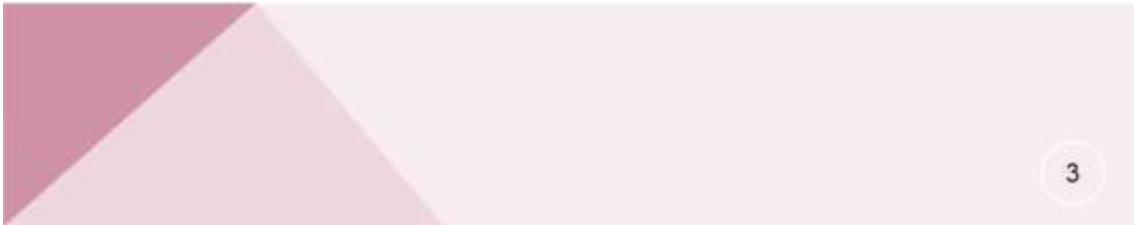


Рисунок В.3 – Слайд № 3

Порівняння програмних аналогів

	IMDB	TMDB	Нова програма
Перегляд інформації про телесеріали	Так	Так	Так
Ефективність рецензування	Висока	Обмежена	Висока
Ефективність оцінювання	Висока	Висока	Висока
Україномовний інтерфейс	Ні	Так	Так
Створення переліків телесеріалів	Так	Так	Так
Створення переліків споріднених користувачів	Ні	Ні	Так

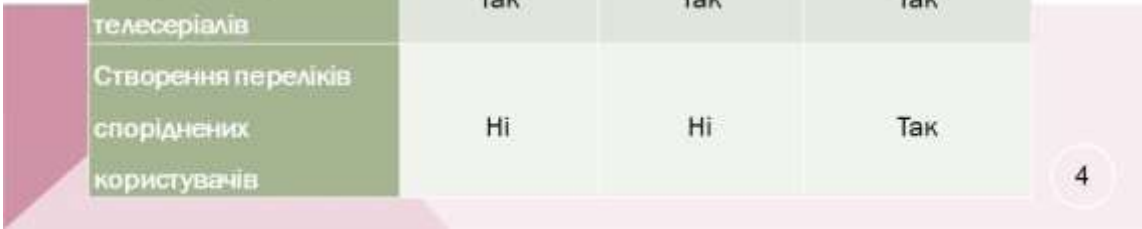


Рисунок В.4 – Слайд № 4

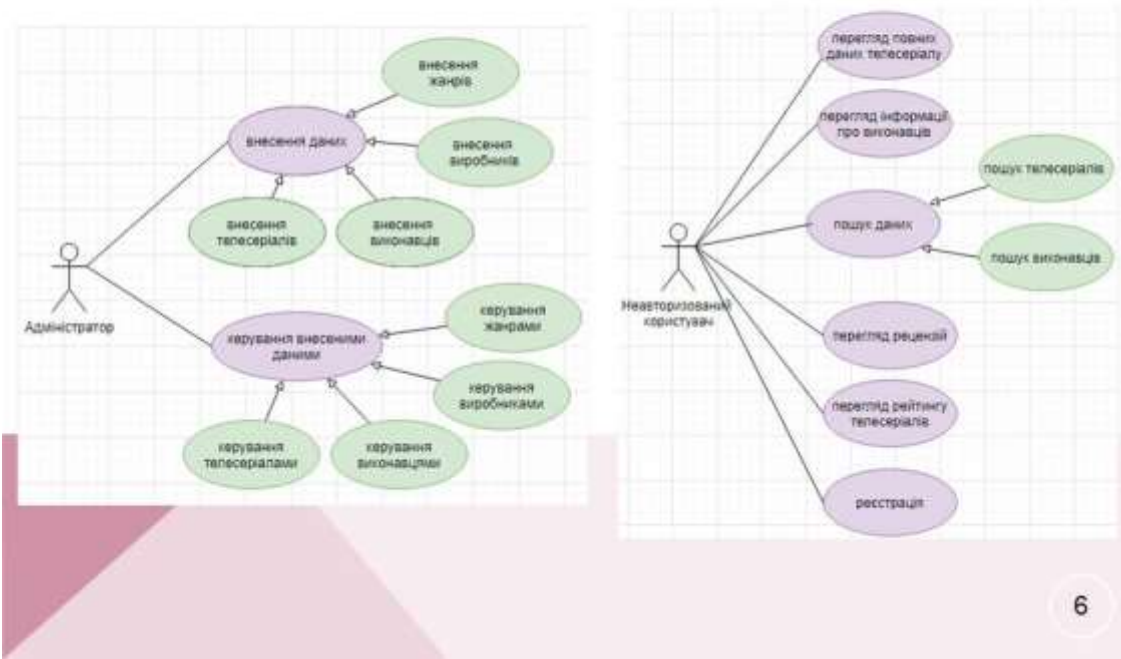
Вибір мови програмування

	Python	C#
Легкість написання коду	Підвищена	Середня
Вебфреймворки	Django	ASP.NET
Повністю безкоштовний набір засобів розробки	Так	Ні
Спільнота розробників	Велика і вільна	Велика

5

Рисунок В.5 – Слайд № 5

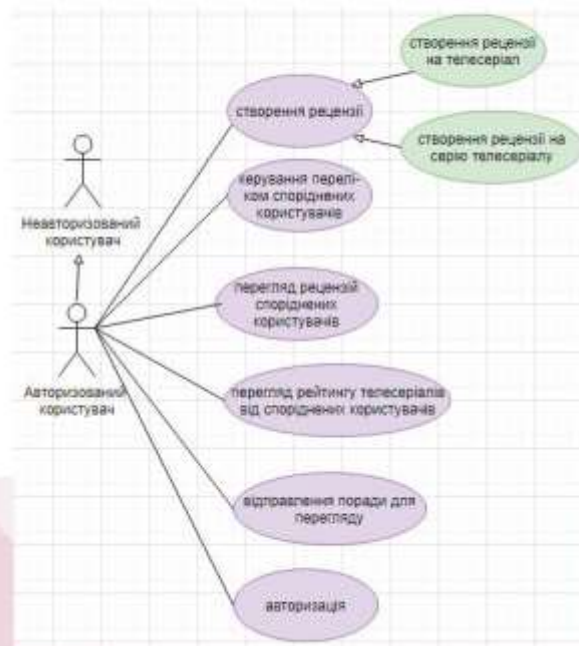
Діаграми прецедентів адміністратора та неавторизованого користувача



6

Рисунок В.6 – Слайд № 6

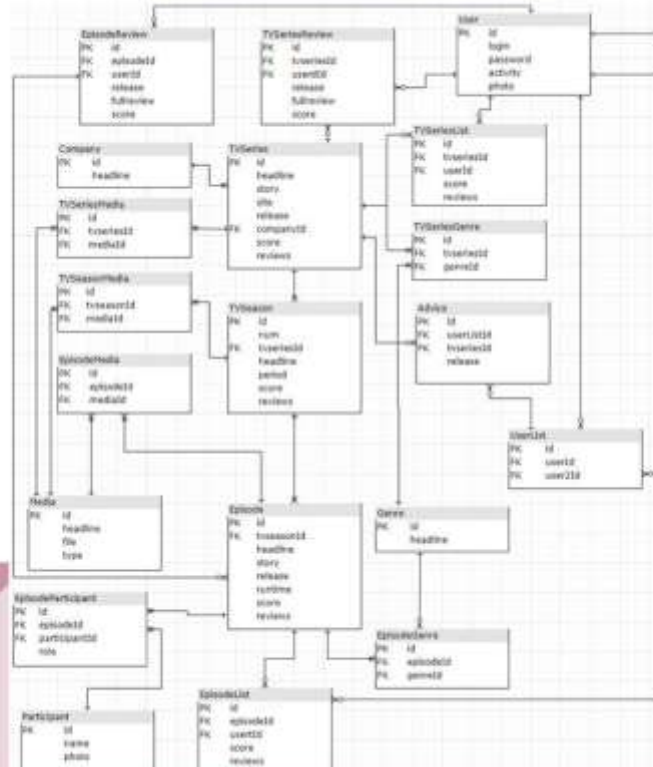
Діаграми прецедентів авторизованого користувача



7

Рисунок В.7 – Слайд № 7

Схема бази даних



8

Рисунок В.8 – Слайд № 8

Структура програми

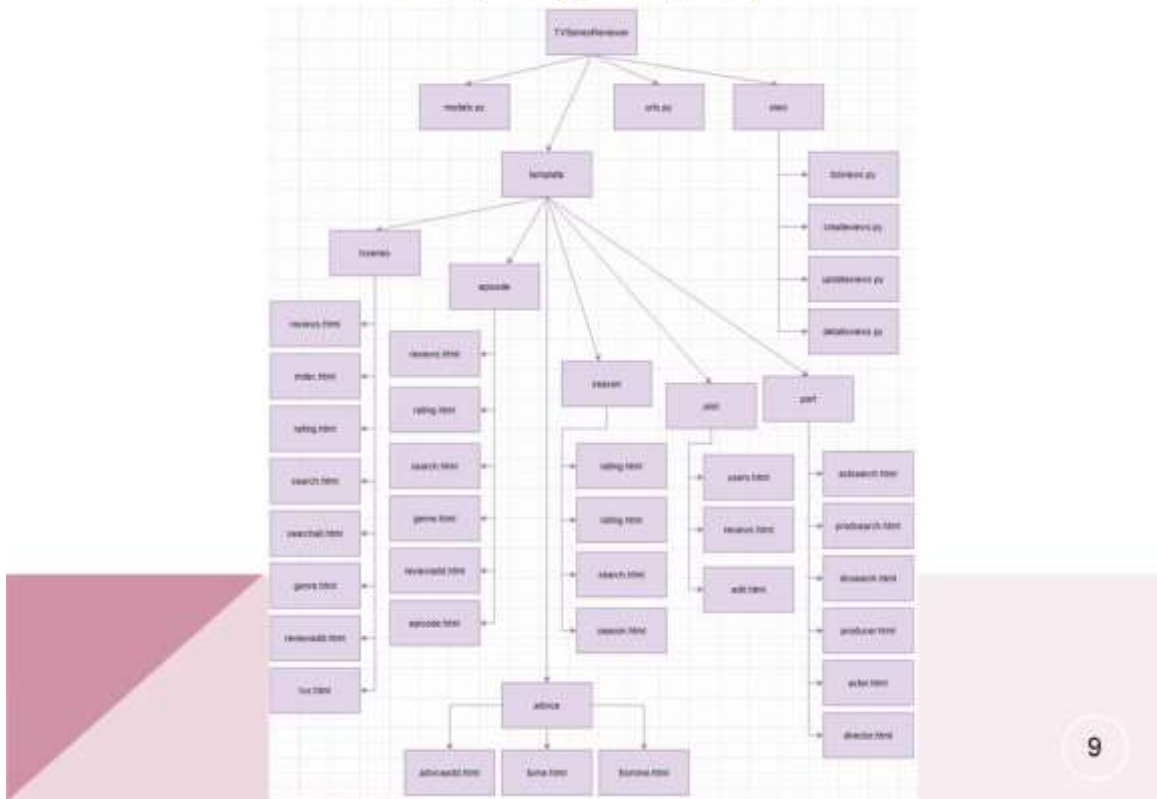


Рисунок В.9 – Слайд № 9

Схема функціонування програми

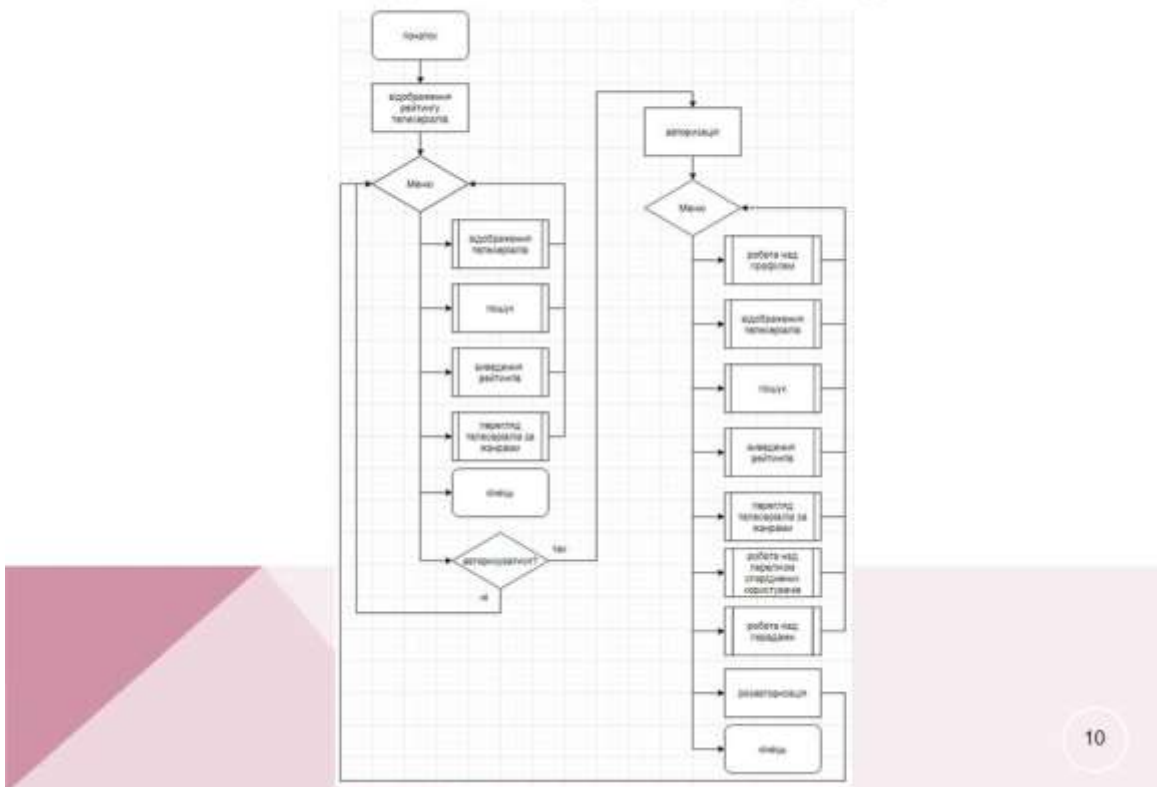


Рисунок В.10 – Слайд № 10

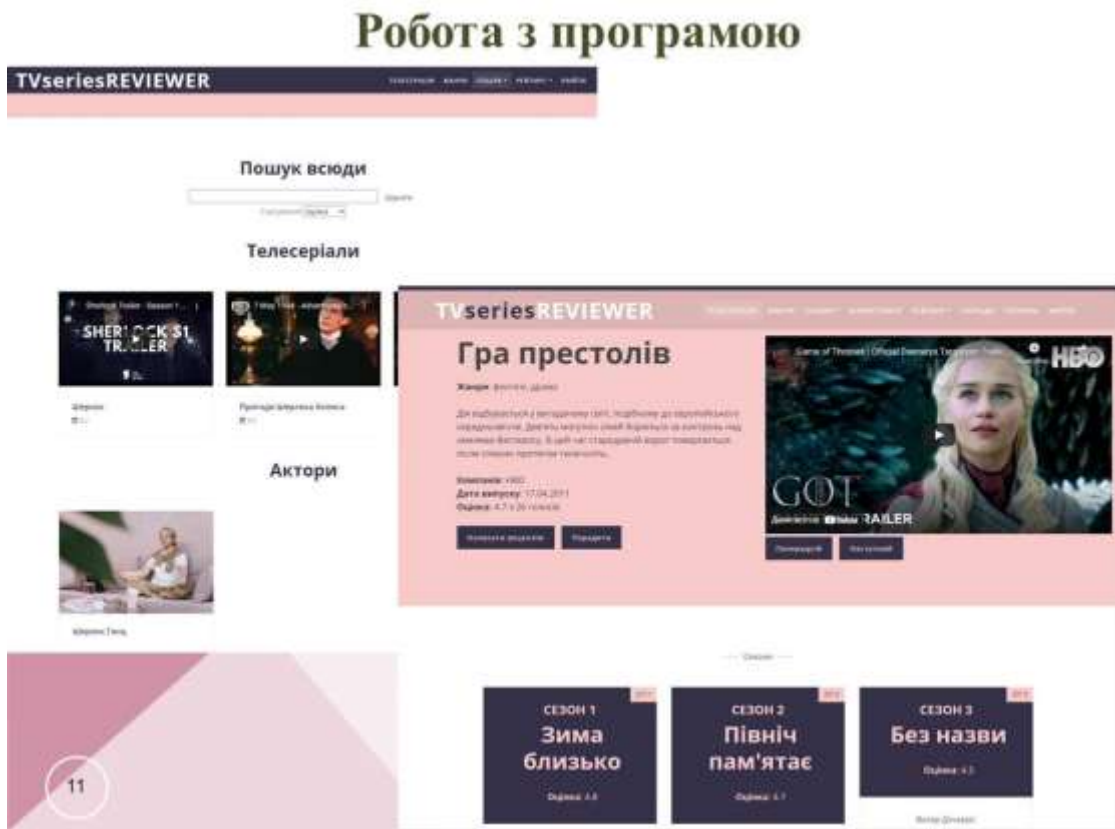


Рисунок В.11 – Слайд № 11

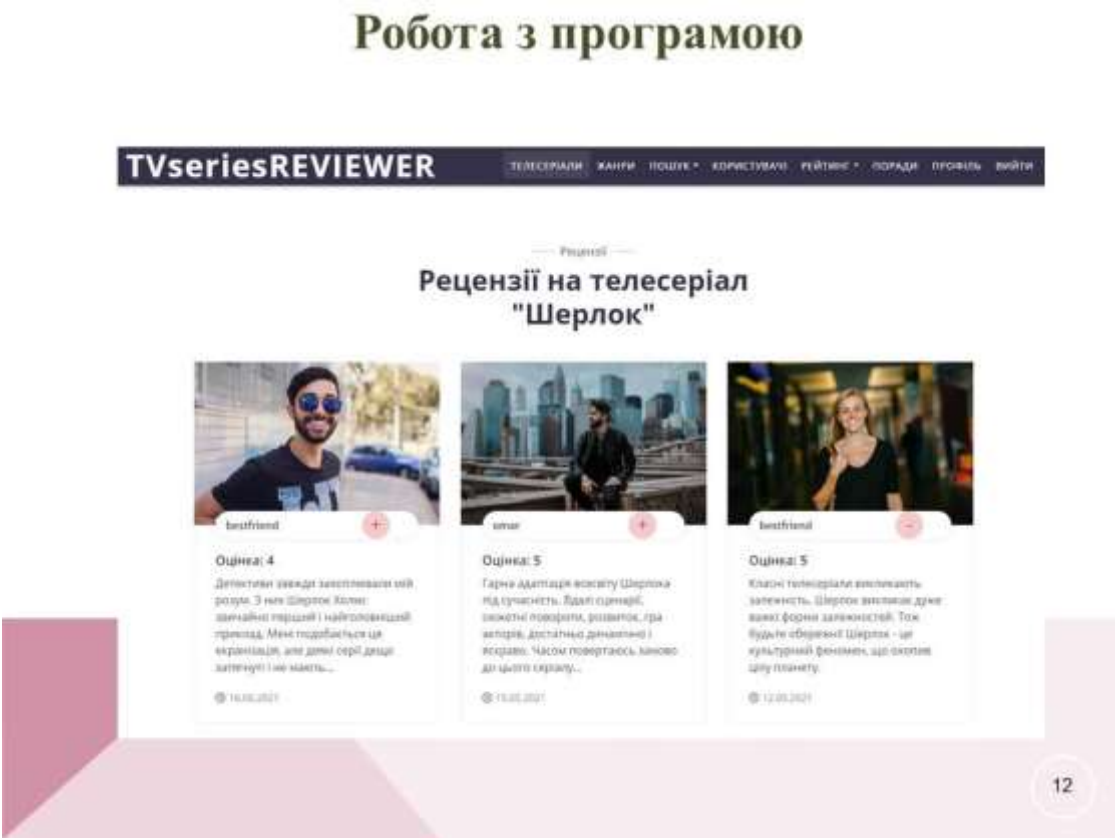


Рисунок В.12 – Слайд № 12

Робота з програмою

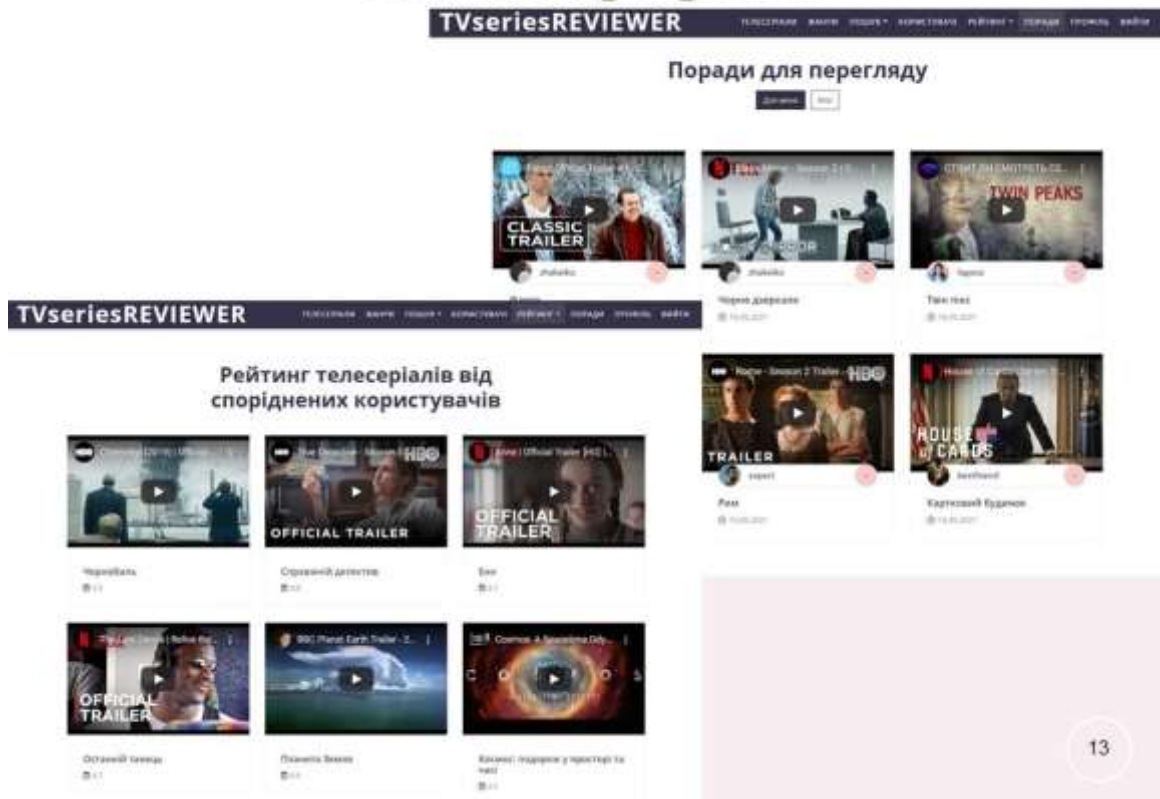


Рисунок В.13 – Слайд № 13

Висновки

У роботі розглянуто проблему розроблення програмного забезпечення рецензування телесеріалів.

Під час проєктування програми визначено функціональні вимоги до програми, виділено сценарії роботи з програмою неавторизованим і авторизованим користувачами та адміністратором, виконано вибір мови Python для розробки програми, описано структуру даних у базі даних, обрано MySQL в якості системи керування базами даних.

Розглянуто результати, отримані в процесі реалізації програмного забезпечення у вигляді вебзастосунку: описано алгоритм функціонування програми, побудовано схему функціонування програми, описано структуру програми і представлено структурну схему програми, описано класи, що були розроблені під час реалізації програми, а також особливості такої реалізації.

Завдання дипломної кваліфікаційної роботи в результаті повністю виконано.

Рисунок В.14 – Слайд № 14