Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка методу та засобів сервісу генерації відеоконтенту на основі фільтрів»

Виконав: студент II курсу групи 1ПІ-20м
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Мазур О. В.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Черноволик Г. О.
(прізвище та ініціали)
«____» _____ 2021 р.
Опонент: к.т.н., доц. каф. КН Крилик Л.В.
(прізвище та ініціали)
«____» _____ 2021 р.

Допущено до захисту
Завідувач кафедри ПЗ
проф. Романюк О. Н.
«____» _____ 2021 р

Вінниця ВНТУ – 2021 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
«13» вересня 2021 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Мазуру Олегу Віталійовичу

1. Тема роботи**:** Розробка методу та засобів сервісу генерації відеоконтенту на основі фільтрів.

Керівник роботи: к.т.н., доц. каф. ПЗ Черноволик Г.О. , затверджені наказом вищого навчального закладу від «24» вересня 2021 р. № 277

2. Строк подання студентом роботи 1 грудня 2021 р.

3. Вихідні дані до роботи: веб-браузер Chrome (або будь-які інші його аналоги); середовище розробки – Intellij IDEA; база даних – PostgerSQL; мова програмування – Java; фреймворк для серверної частини – Spring; фреймворки для клієнтської частини – Freemarker, Bootstrap; розподілена система управління версіями – Git.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз та постановка задачі; розробка методів та засобів додатку; розробка інтерфейсу; розробка додатку; тестування додатку; висновки; перелік посилань; додатки.

5. Перелік графічного матеріалу: аналоги додатку; структура інтерфейсу додатку; діаграми класів додатку; фрагменти коду генерації; загальний алгоритм роботи додатку; результати тестування додатку.

6. Консультанти розділів магістерської кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Черноволик Г.О., к.т.н., доц. каф. ПЗ		
5	Ратушняк О. Г., к.т.н, доц. каф. ЕПВМ		

7. Дата видачі завдання 14 вересня 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану питання та постановка задачі дослідження	15.09.2021 р. – 25.09.2021 р.	Вик.
2	Розробка методів для генерації відеоконтенту	25.09.2021 р. – 13.10.2021 р.	Вик.
3	Розробка сервісу генерації відеоконтенту	13.10.2021 р. – 17.11.2021 р.	Вик.
4	Тестування програмного продукту	17.11.2021 р. – 24.11.2021 р.	Вик.
5.	Економічна частина	24.11.2021 р. – 30.11.2021 р.	Вик.

Студент _____ **Мазур О.В.**

(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Черноволик Г.О.**

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 621.374.415

Мазур О. В. Сервіс генерації відеоконтенту на основі фільтрів. Магістерська кваліфікаційна робота зі спеціальності 121 – Інженерія програмного забезпечення, освітня програма – програмна інженерія. Вінниця: ВНТУ, 2021. 113 с.

На укр. мові. Бібліогр.: 17 назв; рис.: 40; табл. 8.

Результатом виконання магістерської кваліфікаційної роботи є програмний продукт для генерації відеконтенту на основі фільтрів. Розроблено метод генерації відеоконтенту. Було описано класи, структури, графічний інтерфейс програмного додатку та архітектуру.

При створенні програмного продукту та серверної частини було використано мову програмування Java, а також фреймворк Spring – для серверної частини і бібліотеку Bootsrap та Freemarker – для клієнтської частини. Сервіс генерації відповідає вимогам, має сучасний зручний та зрозумілий інтерфейс.

Розробка продукту була виконана у середовищі програмування Intellij IDEA.

ABSTRACT

UDC 621.374.415

Mazur O. V. Video content generation service based on filters. Master's thesis in specialty 121 - Software Engineering, educational program - software engineering. Vinnytsia: VNTU, 2021. 113 p.

In Ukrainian language. Bibliogr .: 17 titles; fig .: 40; table 8.

The result of the master's qualification work is a software product for the generation of video content based on filters. A method of video content generation has been developed. Classes, structures, graphical application interface and architecture were described.

The software and server part were created using the Java programming language, as well as the Spring framework for the server part and the Bootsrap and Freemarker libraries for the client part. The generation service meets the requirements and has a modern, convenient and clear interface.

Product development is performed in the Intellij IDEA programming environment.

ЗМІСТ

ВСТУП 8
1 АНАЛІЗ сучасного СТАНУ ПИТАННЯ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ 10
1.1 Аналіз сучасного стану питання 11
1.2 Порівняльний аналіз аналогів 15
1.3 Аналіз методів і засобів реалізації програмного продукту 18
1.4 Постановка задач магістерської кваліфікаційної роботи 21
1.5 Висновки 21
2 РОЗРОБКА МЕТОДУ ДЛЯ ГЕНЕРАЦІЇ ВІДЕОКОНТЕНТУ 23
2.1 Розробка структурної схеми програмного засобу сервісу генерації відеоконтенту 23
2.2 Розробка методу генерації відеоконтенту 28
2.3 Розробка алгоритму програмного продукту 29
2.4 Висновки 34
3 РОЗРОБКА СЕРВІСУ ГЕНЕРАЦІЇ ВІДЕКОНТЕНТУ НА ОСНОВІ ФІЛЬТРІВ 35
3.1 Обгрунтування вибору мови розробки 35
3.2 Обгрунтування вибору середовища розробки 41
3.3 Програмна реалізація засобу сервісу генерації відеоконтенту, із використанням розширених фільтрів 48
3.4 Висновки 62
4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАСОБУ 63
4.1 Методи тестування 63
4.2 Тестування програмного засобу 67
4.3 Висновки 73
5 ЕКОНОМІЧНА ЧАСТИНА 74
5.1 Оцінювання комерційного потенціалу розробки 74
5.2 Прогнозування витрат на виконання науково-дослідної роботи 77
5.3 Розрахунок економічної ефективності науково-технічної розробки 82
5.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності . 83
5.5 Висновки 86

ВИСНОВКИ 87
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 88
ДОДАТКИ 90
Додаток А. Технічне завдання 91
Додаток Б. Протокол на плагіат 92
Додаток В. Лістинг програмного застосунку 93
Додаток Г. Ілюстративна частина 108

ВСТУП

З усіх видів мистецтва кіно займає унікальне місце в сучасному світі, а відповідно і в житті людини. Кінематограф впливає на життя суспільства, формуючи свідомості глядача. Кінофільм формує світогляд людини, збагачує або обкрадає його духовно, емоційно насичує. Звідси вимальовується ще одна родзинка кінематографа — психологічний вплив на людину.

Ми живемо у добу пікового контенту. Розважального продукту створюють стільки і в такому темпі, що проблеми "що подивитись", здається, більше не існує. Згідно з аналізом дослідженого матеріалу з’ясовано, що кінематограф має велику вагу у житті людини, як розважальна функція; він несе також і пізнавальну та розвивальну функцію. Кінематограф допомагає глибше проникнути в драматургічні, прозові твори великих класиків і ознайомлює з творами сучасних авторів.

Згідно з аналізом дослідженого матеріалу з’ясовано, що кінематограф має велику вагу у житті людини, як розважальна функція; він несе також і пізнавальну та розвивальну функцію. Кінематограф допомагає глибше проникнути в драматургічні, прозові твори великих класиків і ознайомлює з творами сучасних авторів[1].

Актуальність даної роботи полягає у оптимізації процесу вибору фільмів, шляхом реалізації генератору відеоконтенту на основі фільтрів.

Мета та завдання дослідження. Основною метою роботи є зменшення часу процесу вибору кінофільмів користувачами, шляхом генерації відеоконтенту за допомогою різноманітних фільтрів.

Відповідно до поставленої мети потрібно виконати такі задачі:

- Розробити метод генерації відеоконтенту;
- Розробити та реалізувати способи фільтрації кінофільмів;
- Розробити програмні модулі сервісу генерації відеоконтенту;
- Провести тестування програмного продукту.

Об’єкт дослідження – процес реалізації генерації та візуалізації відеоконтенту із допомогою програмного засобу.

Предмет дослідження – методи та засоби сервісу генерації відеоконтенту на основі фільтрів.

Методи дослідження:

– метод проектування програмного сервісу для фільтрації відео контенту;

– метод теорії алгоритмів для розробки алгоритмів і розробки сервісу;

– метод генерації випадкових чисел, для генерації послідовності номерів.

Наукова новизна отриманих результатів полягає в наступному:

Подальшого розвитку набув метод генерації відеоконтенту, який відрізняється від існуючого, використанням розширених фільтрів, що дозволяє розширити функціонал пошуку для користувача.

Практична цінність одержаних результатів полягає у тому, що на основі проведених теоретичних досліджень і отриманих наукових результатів розроблено програмний засіб для актуального та багатофункціонального сервісу генерації відео контенту.

Особистий внесок здобувача. Усі наукові результати отримано автором самостійно. У праці, опублікованій у співтоваристві, здобувачу належить запропонований аналіз методів і засобів генерації.

Апробація матеріалів магістерської кваліфікаційної роботи. Основні положення й результати досліджень було подано на:

1. XIV Міжнародна науково-практична конференція «Інформаційні технології і автоматизація - 2021»[2] - 15 жовтня 2021 р.

2. Електронні інформаційні ресурси: створення, використання, доступ. Пам’яті Олексія Петровича Стахова [3]. Збірник матеріалів Міжнародної науково-практичної Інтернет конференції 9-10 листопада 2021 р. – Суми/Вінниця: НІКО/ВНТУ, 2021. – 223 с.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ПИТАННЯ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

Кіно – це дивовижне досягнення людства. За допомогою кінематографа людина на певний час може перенестись в інший світ та пережити масу яскравих емоцій. Такі емоції можуть бути найрізноманітнішими, оскільки на даний момент існує чимала кількість кіножанрів, здатних змусити людину плакати, сміятися, помирати від страху або поєднати все це в єдиному пориві.

Навряд чи можна знайти людину, якій процес перегляду кіно не приносив би задоволення. У момент емоційного підйому і певного настрою дуже важливо знайти джерело «натхнення», щоб підкріпити почуття, що відчуваються на даний момент, а це більш ніж реально зробити за допомогою сучасного кіно.

Користь від перегляду правильної картини може бути дуже великою. Багато вчених припускають, що у такий спосіб можна впоратися з низкою найскладніших емоційних проблем.

Кіно може навчити нас багато чому: як правильно поводитися у певній ситуації, як реагувати на ті чи інші події в житті, сприймати та аналізувати те, що відбувається. Є чимало фільмів, які після перегляду стають знаковими, сильно впливаючи на наше життя, змінюючи його на краще.

Фільми можуть значним чином створити правильний настрій і дають ґрунт для переосмислення всього, що відбувається. У разі ми ніби дивимося він із боку, у результаті відбувається переоцінка цінностей і ідеалів.

За допомогою фільмів можна формувати характер та поведінкову лінію дитини, її свідомість. Сімейний або груповий перегляд фільмів вкрай зближує, тому це дозвілля варто практикувати якнайчастіше.

Кіно це цілий світ. Світ прекрасний і жахливий, веселий та сумний, добрий та злий, кольоровий. Це світ історії та сучасності, реальності та казки, любові та ненависті. Кіно люблять всі від малого до великого, адже перегляд фільму - процес захоплюючий і пізнавальний. Кіно приносить задоволення, прикрашає дозвілля, піднімає настрій або, навпаки, наводить на сумні думки. Кіно здатне викликати у глядача різні емоції — сміх і сльози, смуток і

радість[4].

1.1 Аналіз сучасного стану питання

Питання впливу кіно на глядачів почало цікавити вчених та психологів із самого початку розвитку цього виду мистецтва. Поступово накопичена інформація та проведені дослідження дозволили розробити методи лікування, що базуються на ефекті, що отримується від перегляду фільмів. Завдяки можливості аналізувати поведінку героїв на екрані та проектувати побачену ситуацію на своє життя кіно терапія стала цінним інструментом, який не тільки чинить терапевтичну дію, а й мотивує на розвиток здібностей та підвищення професійних навичок.

Ефективність фільмів як психологічного інструменту обумовлена такими факторами. Інтенсивність: історії, герої та сценарій зібрані разом та «упаковані» в обмежений за часом формат. Тривалість: фільм за тривалістю порівняний з великим сеансом психотерапії. Навчання: фільми, подібно до притч і казок, здатні доносити інформацію у вигляді алегорії. Ідентифікація: багато хто з нас ідентифікував себе з героями фільму. Увага: завдяки візуальному впливу фільми сприяють концентрації уваги на різних образах. Соціальна користь: перегляд фільму разом із друзями та подальше обговорення збільшують його цінність, сприяють розвитку відносин та підвищують ефективність терапії.

Кінотерапія надає безліч відчутних переваг - ось лише деякі з них. Відновлення - перегляд фільмів може сприяти відновленю дію завдяки тому, що дозволяє відволіктися, розслабитися і добре провести час. Робота зі страхами – за допомогою окремих сцен або цілих фільмів кінотерапія дає можливість дивитися своїм страхам в обличчя, щоби вчитися перемагати їх. Усвідомлення проблем - співпереживаючи героям фільму, багато хто усвідомлює власні проблеми, наявність яких до цього не було очевидним. Можливість пережити ситуацію - фільми є своєрідним симулятором реальності, дозволяючи пережити ситуацію, не стикаючись з її реальними наслідками.

Фільми є потужним джерелом мотивації - під час перегляду можна побачити героїв, приклад яких надихне нас у реальному житті. Розвага - позитивні емоції, які приносять фільми, не минають без користі для психічного та фізичного здоров'я. Сльози та плач – сумні фільми можуть вивести на поверхню глибоко сховані емоції. Переосмислення негативних думок – через фільми можна побачити свою ситуацію з іншого боку, що дозволить замислитися та змінити до неї ставлення. Розвиток креативності - різні точки зору, представлені у фільмах, можуть змінити шаблони мислення та підштовхнути до більш творчого, гнучкого та інноваційного підходу. Фільми допомагають впоратися з втратою - розбите серце та втрата можуть бути пом'якшені фільмами, які допоможуть пережити ці періоди[5].

Кіно грає глобальну роль життя людей, воно пов'язані з ними набагато сильніше, ніж скажемо театр. А пізніше, коли з'явився телевізор, можна навіть не ходити в кінотеатри, а насолоджуватися улюбленими фільмами домашньої обстановки. Фільми, з дитячих років вчать дітей розрізняти добро і зло, випробувати почуття жалості до когось або радіти витівкам героїв. А якщо це документальне кіно, то в ньому розповідається теж чимало цікавого та пізнавального, про історію, традиції та культуру різних країн світу або ж це передачі про тварин, яких дуже багато, і кожен глядач обов'язково знайде для себе щось до душі[6].

Кіноіндустрія перебуває у постійному зростанні, нині для виробництва фільму використовуються новітні технології, сучасні спецефекти, графіка зачаровують глядачів. А якщо це 3Д, то взагалі відчуття, що ти потрапив разом з героями або на безлюдний острів, або на батискаф підводного корабля.

В даний час прогрес дійшов до того, що можливо навіть завантажувати фільми, і дивитися навіть раніше, ніж вони вийшли в маси. Основна мета кінематографа це внести фарби в повсякденне життя і додати до неї позитиву. І з цією метою кінематограф він чудово справляється.

Сьогодні складно знайти людину, яка не любить подивитися вдома захоплюючий фільм чи серіал, або ознайомитися з якоюсь касовою новинкою у

кінотеатрі. Деякі люди просто іноді коротають таким чином дозвілля. Є такі «кіномани», які дня не можуть прожити без яскравих емоційних подій та оригінальних сюжетів, якими просякнутий сучасний кінематограф.

Правильний приклад, який викладає улюблений герой, може крайнім чином змінити людину на краще. Дати йому мотивацію на якісне покращення власного життя. Також більшість дівчат та молодих людей можуть копіювати стиль одягу та поведінки, прагнути бути більш стильними та сучасними.

Крім того, перегляд комедії, а саме веселий заразливий сміх, здатний сприятливо впливати на загальний стан здоров'я та тривалість життя.

Фільми – це відображення нашої реальності, а люди ніколи не втомляться спостерігати за собою. Кіно ніколи не перестане бути актуальним, оскільки воно змінюється паралельно з нашим життям і завжди буде цікавим для великої кількості глядачів.

Пошукова система — це програмна програма, яка допомагає людям знаходити інформацію, яку вони шукають в Інтернеті, за допомогою ключових слів або фраз.

Пошукові системи можуть швидко відображати результати — навіть із мільйонами веб-сайтів у мережі — безперервно скануючи Інтернет та індексуючи кожну знайдену сторінку

Пошукові системи дозволяють користувачам шукати вміст в Інтернеті за допомогою ключових слів. Хоча на ринку домінують деякі, існує багато пошукових систем, які люди можуть використовувати. Коли користувач вводить запит до пошукової системи, повертається сторінка результатів пошукової системи (SERP), яка ранжує знайдені сторінки в порядку їх релевантності. Спосіб визначення цього рейтингу відрізняється в різних пошукових системах.

Пошукові системи часто змінюють свої алгоритми (програми, які оцінюють результати), щоб покращити роботу користувачів. Вони прагнуть зрозуміти, як користувачі шукають, і дати їм найкращу відповідь на їхні запити. Це означає надавати пріоритет найякіснішим і най бажаним сторінкам.

Для роботи більшості пошукових систем необхідно виконати три основні кроки:

Сканування – пошукові системи використовують програми, які називаються павуками, ботами або сканерами, для перегляду Інтернету. Вони можуть робити це кожні кілька днів, тому вміст може бути застарілим, доки вони знову не сканують ваш веб-сайт.

Індексування – пошукова система намагатиметься зрозуміти та класифікувати вміст веб-сторінки за допомогою «ключових слів». Дотримання найкращих методів SEO допоможе пошуковій системі зрозуміти ваш вміст, щоб ви могли оцінювати відповідні пошукові запити.

Рейтинг – результати пошуку ранжуються на основі ряду факторів. Вони можуть включати щільність ключових слів, швидкість і посилання. Мета пошукової системи – надати користувачеві найбільш релевантний результат.

Хоча більшість пошукових систем дадуть поради щодо покращення рейтингу вашої сторінки, точні алгоритми, які використовуються, добре охороняються та часто змінюються, щоб уникнути неправильного використання. Але дотримуючись найкращих методів пошукової оптимізації (SEO), ви можете переконатися, що:

Пошукові системи можуть легко сканувати ваш веб-сайт. Ви також можете запропонувати їм сканувати новий вміст. Ваш вміст індексується за потрібними ключовими словами, тому він може відображатися для релевантних пошукових запитів.

Під час пошуку даних різниця між швидким додатком і повільнішим полягає в точному використанні алгоритму пошуку. Алгоритми пошуку є основним, фундаментальним кроком у обчисленні, який виконується за допомогою покрокового методу для визначення місця розташування конкретних даних серед колекції даних.

Усі алгоритми пошуку використовують ключ пошуку для завершення процедури. І очікується, що вони повернуть статус успішного або невдалого (у логічному значенні true або false).

1.2 Порівняльний аналіз аналогів

На сьогоднішні день існує багато різних вебсайтів генерації фільмів з своїми недоліками і перевагами. Нижче наведено

Free-generator – (Рис. 1.1) - генератор дає змогу швидко знаходити цікаві фільми. Для чого необхідно виконати всього 3 простих кроки:

– виберіть жанр: один або всі;

– виберіть країну;

– встановіть період.

Щоб отримати результат, потрібно натиснути "сгенерировать фильм". Ви маєте побачити всю необхідну інформацію: назву, рік випуску, тривалість, опис сюжету, а також рейтинг на «Кінопошуку» та IMDb. [7]. Недоліком даного веб сайту є відсутність інтерфейсу українською мовою та не доцільно працююча генерація фільмів.

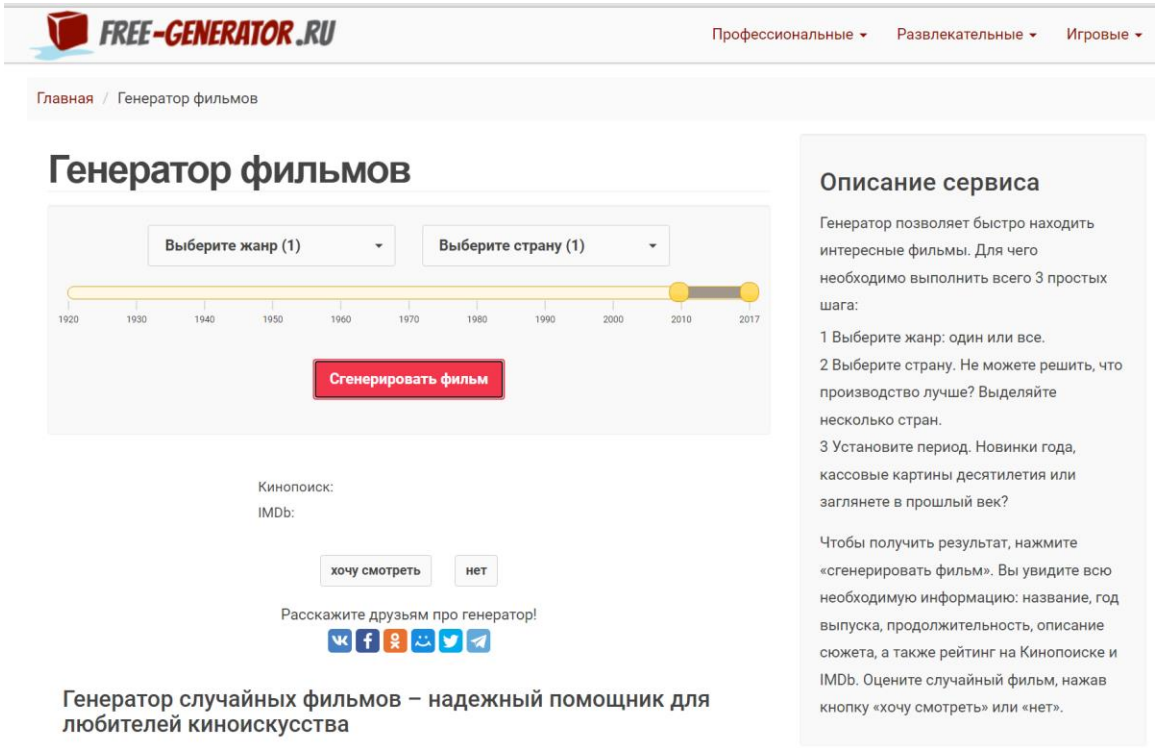

Рисунок. 1.1 - Інтерфейс Free-generator

Generator-online (Рис.1.2) – генератор фільмів онлайн. Програма підбере варіант з високим рейтингом з десятків тисяч кінострічок в 19ти жанрах за більш ніж 100 років історії світового кінематографа.

Простий спосіб знайти хороший фільм - задати параметри фільму в сервісі. Встановіть параметри, такі як жанр і рік релізу, натиснувши «Генерувати результат» і ви отримаєте картку з коротким описом відповідного кіно. [8]. Недоліком є застарілий інтерфейс та база даних, яка видає фільми низького рейтингу. Також мала кількість інформації про згенерований фільм, та неможливо подивитись трейлер до фільму

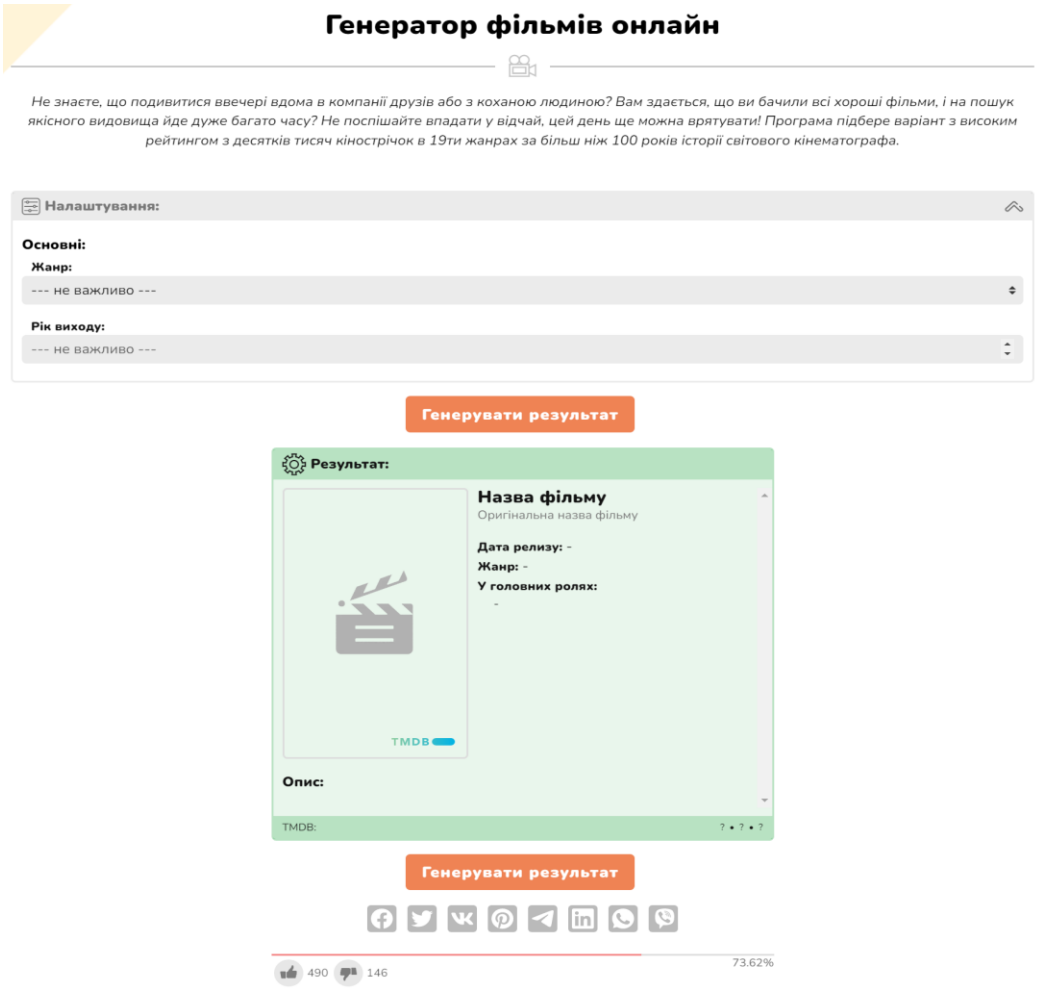

Рисунок 1.2. - Інтерфейс Generator-online

Coolgenerator (Рис.1.3) - це генератор випадкових фільмів. В ньому

зібрано 5000 найкращих фільмів з 1902 року до сьогодні. Ці фільми містять майже всі жанри фільмів, пригод, бойовиків, комедій тощо. За допомогою цього генератора можна отримати випадковий фільм.

Користуватися цим генератором дуже просто, спочатку обираються фільтри, такі як: виберіть жанр фільму, рік виходу, рейтинги. У списку показаних фільмів кожен фільм містить назву фільму, рік випуску, рейтинг, огляд фільму, які можуть допомогти вам обрати фільм. Недоліком є застарілий візуальний інтерфейс, англомовна розкладка, замала кількість фільтрів.

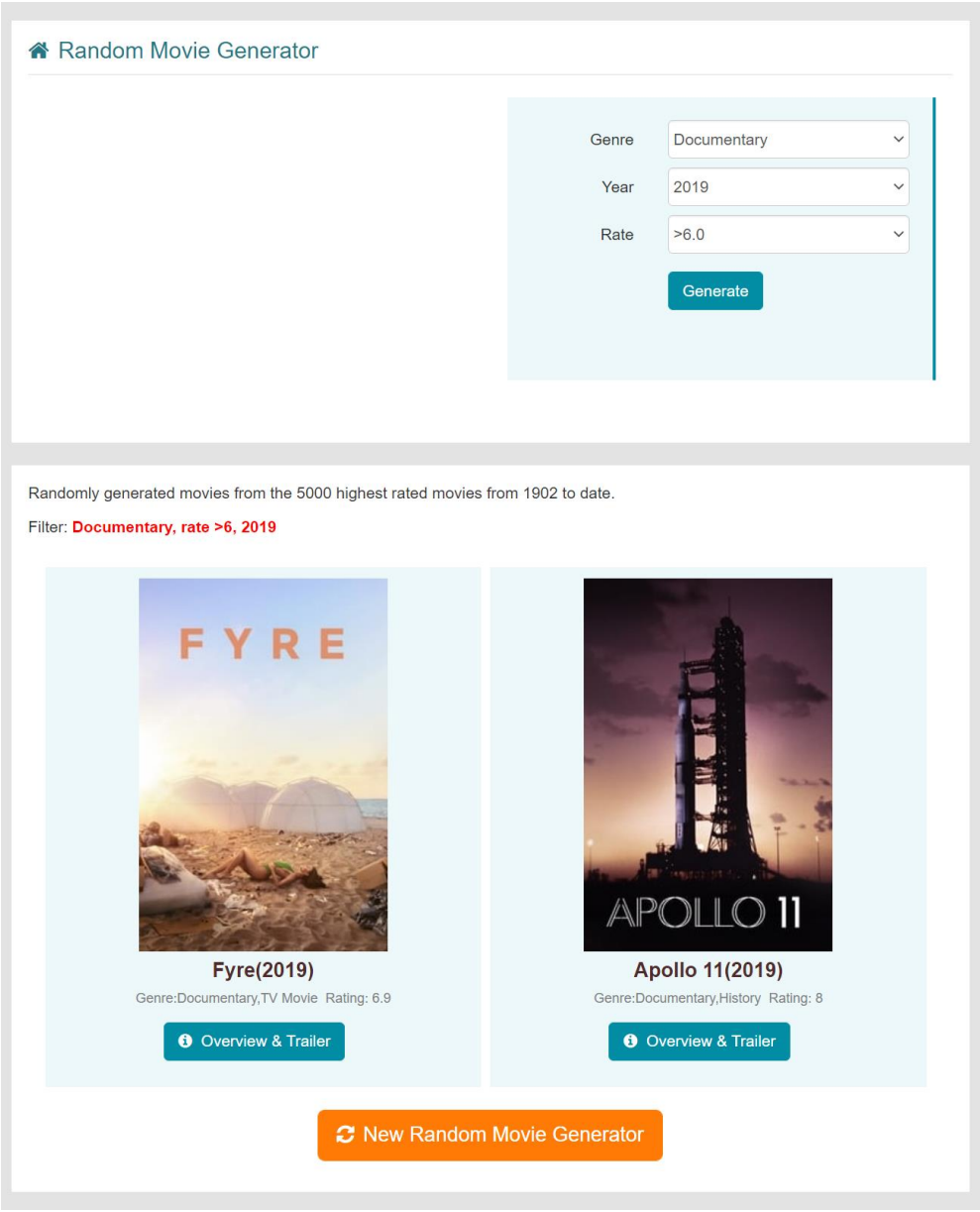

Рис. 1.3 - Інтерфейс Cool generator

Порівняння аналогів з розроблюваним сервісом генерації описано в табл. 1.1.

Таблиця 1.1 – Порівняльні характеристики програмних продуктів

Умова	Free-generator	Generator-online	Cool generator	Movie Generator
Доступність	–	+	+	+
Надійність	–	+	–	+
Український інтерфейс	–	+	–	+
Сучасний простий інтерфейс	–	–	–	+
Народне голосування	–	–	–	+
Якісне швидке генерування	–	+	–	+

Враховуючи недоліки зазначених вище веб сайтів можна зробити висновок, що розробка власної реалізації є доцільною. Буде розроблено веб-сайт українською мовою для генерації відеконтенту на основі різноманітних фільтрів.

1.3 Аналіз методів і засобів реалізації програмного продукту

Випадкові номери зазвичай використовуються для створення таких додатків, як кістки для настільних ігор, азартних програм тощо. Зазвичай генерація випадкових чисел займає багато часу. Але в мові програмування Java цього можна досягти, використовуючи три способи. Вони розглядаються нижче в розділі Функції генератора випадкових чисел на Java.

У Java випадкові номери можна генерувати, використовуючи 3 способи:

– math. випадковий метод;

– java.util.Random class;

– клас ThreadLocalRandom.

Клас Java Math пропонує ряд методів для роботи над обчисленнями, такі як логарифми, середнє значення, експоненція тощо. Random () - один із методів серед них, який повертає додатне подвійне значення в межах 0, 0 та 1, 0, де 0, 0 включно і 1.0 є ексклюзивним. Цей метод можна використовувати з використанням параметрів або без них. Якщо параметри задані, генерується випадкове число буде в межах заданого параметра.

Клас Java.util.Random генерує випадкові числа різних типів даних, таких як float, long, integer, double, Boolean тощо. Також можливо передавати діапазон чисел як аргументи, щоб у цьому діапазоні генерувалося випадкове число . Для використання цього класу необхідно імпортувати клас Random java.util (java.util.Random). Після імпорту цього класу створіть екземпляр і зателефонуйте на такі методи, як next long (), nextInt () тощо, використовуючи цей екземпляр.

Клас ThreadLocalRandom - це спеціалізований тип класу Random, який вводиться у версії Java 1.7. ThreadLocalRandom.current (). NextInt () - один із поширених методів, який використовується для генерації випадкових чисел. Зазвичай використовується в багатопотокових програмах.

Java містить безліч функцій, які можна використовувати в програмах. Це допомагає скоротити час обробки та рядки коду. Генерація випадкових чисел - одне завдання, де ми можемо використовувати деякі з цих функцій. Цей документ охоплює різні методи досягнення цього[9].

Рандомізація — розташування або вибір об'єктів у випадковому порядку. Для випадкового вибору номерів дослідів можна використовувати таблицю випадкових чисел або лотерею.

Застосовується, наприклад, для вибору порядку чергування окремих дослідів при плануванні експериментів тощо.

Рандомізація дозволяє нівелювати систематичні впливи факторів, що не контролюються, а також забезпечити об'єктивність при виборі об'єкта.

При плануванні експериментів окремі досліди повинні виконуватися у послідовності, яка встановлюється за допомогою таблиці випадкових чисел, або будь-якої процедури, що забезпечує випадковий характер проведення дослідів. Рандомізація дозволяє нівелювати систематичні впливи факторів, що не контролюються[10].

Послідовність називається псевдовипадковою, якщо вона виглядає, як безсистемна і випадкова, хоча насправді вона створювалась з допомогою суто детермінованого процесу, відомого під назвою псевдовипадкового генератора. Подібні генератори переважно задаються деяким початковим значенням і за допомогою певних алгоритмів отримують з нього випадкові послідовності. В цьому сенсі псевдовипадкові генератори можна розглядати як розповсюджувачі випадковості. Комп'ютери є детермінованими машинами, що завжди роблять саме те на що вони запрограмовані і це усуває можливість звертатися до комп'ютерів як до джерела істинної випадковості. Саме краще, на що здатний комп'ютер - це згенерувати псевдовипадкову послідовність, яка хоча і виглядає випадковою, але, насправді, такою не є. Згенерувати дійсно випадкову послідовність можна лише при апаратній реалізації генератора, який би для отримання випадкових чисел використовував деяке фізичне явище, наприклад, шум, який генерують напівпровідникові прилади, молодші біти оцифрованого звуку, інтервали між перериванням пристроїв або натисканням клавіш, температуру повітря і т. д. В сучасних потужних криптосистемах військового призначення використовують генератори випадкових чисел (ГВЧ), які є платами або зовнішніми пристроями, які підключаються до ЕОМ через порт вводу-виводу. Незважаючи на труднощі, які виникають при проектуванні генераторів псевдовипадкових чисел (ГПВЧ), вони широко використовуються в прикладних комп'ютерних програмах і легко компонуються з усіма типами комп'ютерних систем. Тому, на сьогоднішній день, більшість прикладних комп'ютерних програм використовують ГПВЧ для генерації потрібних випадкових даних.

Випадкові числа - фундаментальний елемент для надання обмеженого

доступу до інформації. Вони являють собою основний елемент криптографії, цифрового підпису, протоколів безпеки і іншого забезпечення надійності при зв'язку[11].

Генератор псевдовипадкових чисел - алгоритм, що породжує послідовність чисел, елементи якої майже незалежні один від одного і підкоряються заданому розподілу (зазвичай рівномірному).

Отже, після проведеного аналізу методів та засобів реалізації сервісу генерації відеоконтенту, на основі фільтрів, було вирішено використовувати псевдовипадковий метод випадкових чисел, так як саме цей метод дозволяє отримувати дані найшвидше з сервера, записувати данні та здійснювати інші операції з даними за допомогою API.

1.4 Постановка задач магістерської кваліфікаційної роботи

Після аналізу стану сервісів генерації, порівняння існуючих аналогів та аналізу методів і засобів реалізації програмного продукту було визначено наступні завдання, які необхідно виконати для розробки програмного продукту:

- Розробити метод генерації відеоконтенту;
- Розробити та реалізувати способи фільтрації кінофільмів;
- Розробити програмні модулі сервісу генерації відеоконтенту;
- Провести тестування програмного продукту.

1.5 Висновки

В цьому розділі розглянуто аналіз стану програмних засобів для генерації, що показав що створення власного ПЗ є актуальним. Було розглянуто такі аналоги як: Free-generator, Generator-online, Coolgenerator. Порівняння уже створених аналогів показало, що є необхідність розробити власний веб сайт. Також було проведено аналіз методів реалізації програмного продукту. Було обрано методи роботи з генерацією псевдовипадкових чисел. В результаті було розроблено основні завдання, які необхідно виконати при розробці

програмного продукту.

2 РОЗРОБКА МЕТОДУ ДЛЯ ГЕНЕРАЦІЇ ВІДЕОКОНТЕНТУ

2.1 Розробка структурної схеми програмного засобу сервісу генерації відеоконтенту

Структурна схема веб-застосунку відображає призначення та взаємодію головних структурних модулів програми між собою та з користувачем.

В веб-застосунку для генерації відеоконтенту є наступні модулі: база даних, модуль роботи з базою даних, модуль формування html сторінок, обробник веб-запитів клієнта, інтерфейс браузера.

Точкою входу в програму є головне меню, з якого починається навігація програмою по інших її складових.

Функціональна блок-схема в системній інженерії та розробці програмного забезпечення є блок-схемою. Вона описує функції та взаємозв’язки системи.

Блок-схема може використовувати додаткові схематичні символи для відображення певних властивостей.

Функціональні блок-схеми використовувалися в широкому діапазоні додатків, від системної інженерії до розробки програмного забезпечення. Вони стали необхідністю в проектуванні складних систем, щоб «досконало зрозуміти з зовнішнього дизайну роботу нинішньої системи та взаємозв’язок кожної з частин до цілого».

Виникло багато специфічних типів функціональних блок-схем. Наприклад, блок-схема функціонального потоку є комбінацією функціональної блок-схеми та блок-схеми. Багато методологій розробки програмного забезпечення побудовані з використанням певних методів функціональних блок-схем. Прикладом із галузі промислових обчислень є Function Block Diagram (FBD), графічна мова для проектування програмованих логічних контролерів[12].

Розроблена загальна структурна схема функціонування веб-застосунку Movie Generator зображено на рисунку 2.1. Ця схема відображає роботу застосунку на найвищому рівні абстракції.

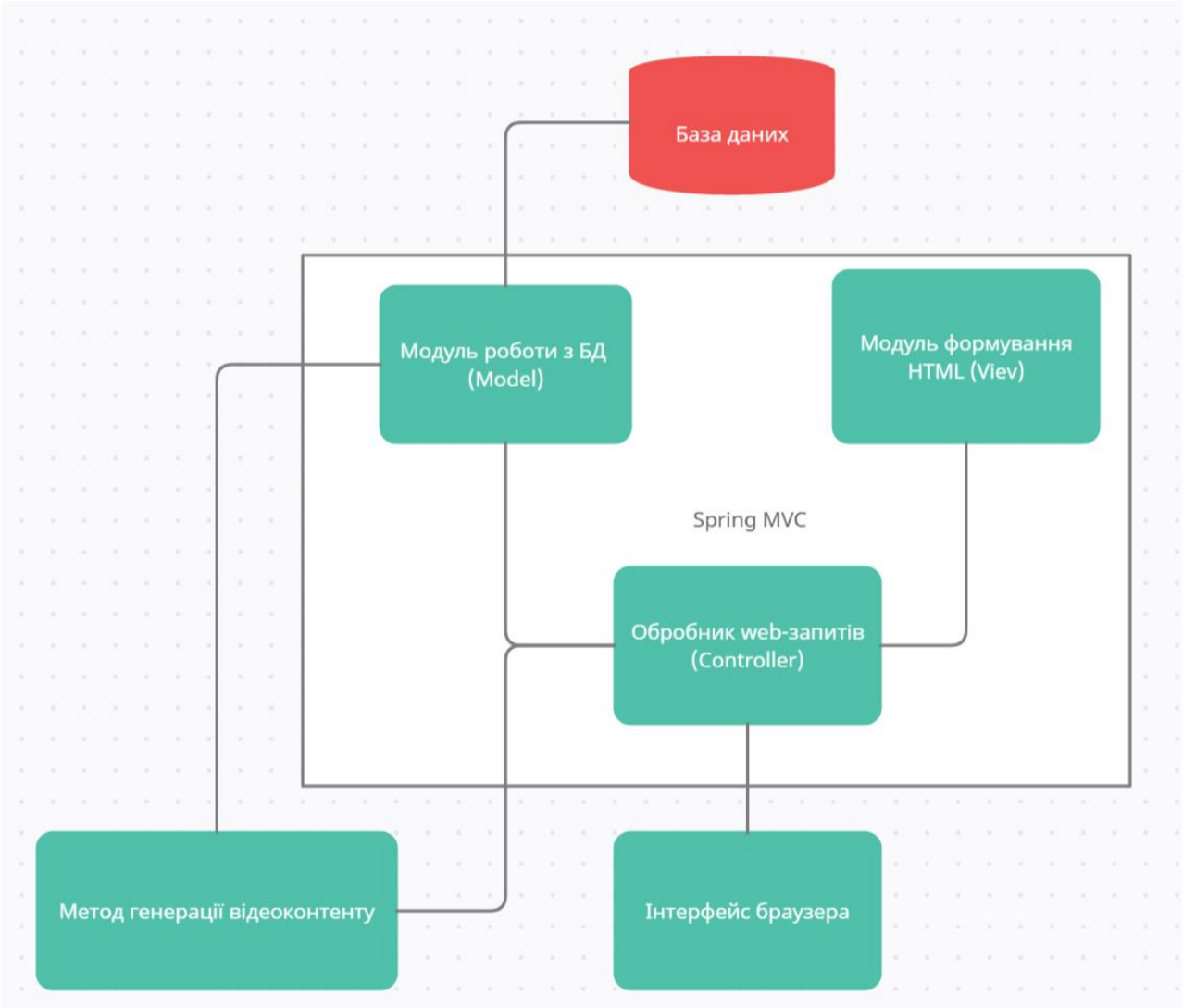

Рисунок 2.1 – Структурна схема функціонування веб-застосунку MovieGenerator

Користувач подає запит на генерацію та пошук, обробник web-запитів в свою чергу передаю його в модуль роботи з БД, який подає адаптує запит під БД, і в БД бере ту інформацію що йому потрібна, і потім повертаючись в обробник web-запитів, переходить в модуль формування HTML, який формує візуалізацію для користувача.

Для розробки великих та складних програм програмісту необхідно опанувати спеціальні прийоми отримання раціональної структури програми,

яка забезпечує майже дворазове скорочення обсягу програмування та багаторазове скорочення.

Підпорядкованість модулів програми відбивається у схемі ієрархії. Однак остання не відображає порядок їхнього виклику або функціонування програми. Вона зазвичай доповнюється розшифровкою функцій, що виконується модулями.

Перед складанням схеми ієрархії доцільно скласти зовнішні специфікації програми та скласти функціональні описи програми разом з описом змінних носіїв даних. Особливу увагу слід приділяти ієрархії типів структурованих даних та їхнього коментування.

Розчленування програми на підпрограми провадиться за принципом від загального до приватного, більш детального. Процес складання функціонального опису та складання схеми ієрархії є ітераційним, а вибір найкращого варіанта є багатокритеріальним. Розчленування повинне забезпечувати зручний порядок введення частин в експлуатацію.

Схеми ієрархії можна надати будь-який топологічний малюнок. Фрагменти з вертикальними викликами можуть бути перетворені на виклики одного рівня за допомогою введення додаткового модуля, який може не виконувати жодних корисних функцій з точки зору алгоритму програми. Функція нового модуля може бути лише моніторингу, тобто виклик інших модулів у порядку[13].

Структурна схема - це сукупність елементарних ланок об'єкта та зв'язків між ними, один із видів графічної моделі. Під елементарною ланкою мається на увазі частина об'єкта, системи керування тощо, яка реалізує елементарну функцію.

Будь-яка складна структура системи може бути представлена у вигляді комбінації попарно пов'язаних між собою ланок, причому існують лише три різновиди таких зв'язків: послідовний, паралельний і зворотний.

При послідовному зв'язку вихідна величина однієї ланки є вхідний для іншої, і, отже, її передавальна функція є твір двох ланок.

При паралельному зв'язку вхідна величина з'єднання є загальною для обох ланок, а вихідна утворюється в результаті підсумовування вихідних даних. Передавальна функція з'єднання дорівнює сумі передавальних функцій ланок.

За наявності зворотного зв'язку одна з ланок системи передає сигнал з виходу другої ланки назад на його вхід, де він підсумовується з вхідним впливом, або віднімається з нього. Канал, яким сигнал з виходу системи знову подається з її вхід, називається зворотним зв'язком, причому у разі зворотний зв'язок вважається позитивної, тоді як у другому — негативною[14].

Діаграма класів на більш детальному рівні абстракції описує взаємодію компонентів програмі між собою, їхню ієрархічну структуру та функції.

Основні типи класів:

1) Конфігуратори: MvcConfig, Application, WebSecurityConfig. Відповідають за загальне налаштування веб-застосунку. В них задаються основні параметри та конфігурації застосунку, такі як авторизація, локалізація, конфігурація шаблонізатора тощо.

2) Репозиторії: GenreRepo, MovieRepo, MovieRepoCustom, MovieRepoImpl, UserRepo. Відповідають за роботу з базою даних, з їх допомогою формуються sql запити до бази даних.

3) Сутності: Genre, Movie, Role, User. Відповідають за представлення даних у таблицях бази даних, зв’язок з програмним кодом та опис основного функціоналу екземплярів даних класів.

4) Контролери: GenreController, HomeController, MovieController, MovieSearchController, RegistrationController, UserController. Відповідають за обробку клієнтських веб-запитів, таких як створення, редагування, видалення, пошук даних в системі, тощо.

Діаграма класів - це статична діаграма. Вона представляє статичний вигляд програми. Діаграма класів використовується не тільки для візуалізації, опису та документування різних аспектів системи, але й для побудови виконуваного коду програмного додатка.

Діаграма класів описує атрибути та операції класу, а також обмеження, накладені на систему. Діаграми класів широко використовуються в моделюванні об'єктно-орієнтованих систем, оскільки вони є єдиними діаграмами UML, які можна відобразити безпосередньо з об'єктно-орієнтованими мовами.

Діаграма класів показує набір класів, інтерфейсів, асоціацій, співпраці та обмежень. Вона також відома як структурна діаграма.

Метою діаграми класів є моделювання статичного вигляду програми. Діаграми класів є єдиними діаграмами, які можна безпосередньо відобразити за допомогою об'єктно-орієнтованих мов і, таким чином, широко використовуватися під час побудови.

Діаграми UML, як-от діаграма активності, діаграма послідовності, можуть давати лише послідовність дій програми, однак діаграма класів дещо інша. Це найпопулярніша діаграма UML у спільноті програмістів.

Діаграма класів веб-застосунку зображена на рисунку 2.2.

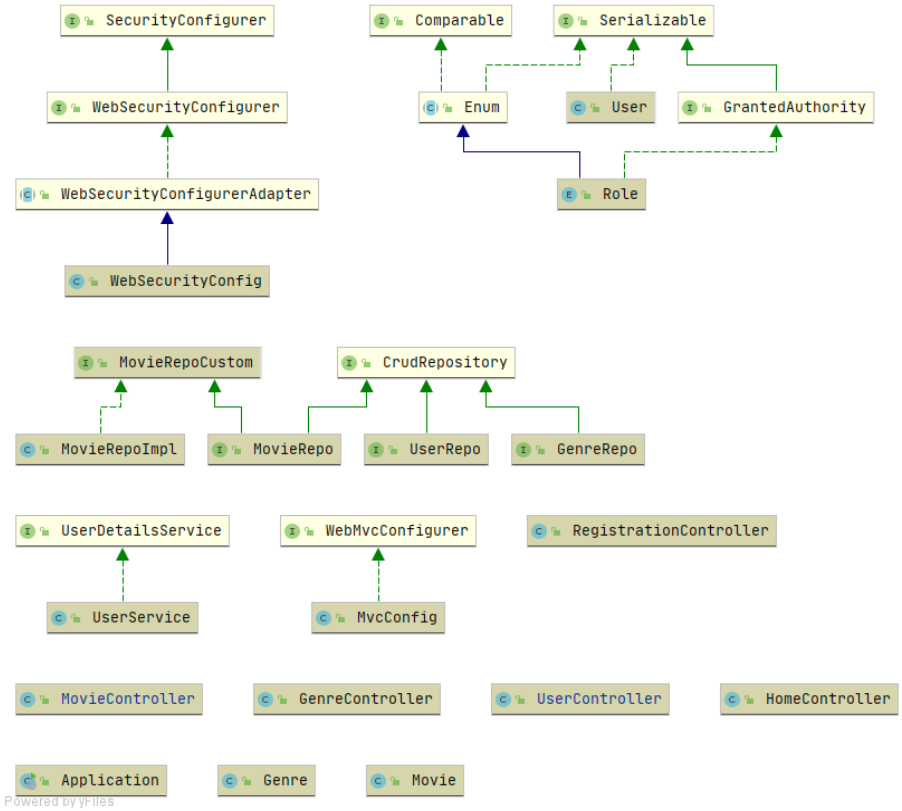

Рисунок 2.2 – Діаграма класів веб-застосунку MovieGenerator

2.2 Розробка методу генерації відеоконтенту

Метод пошуку — формальні правила, що визначають стратегію пошуку інформації, яка зберігається в певній структурі даних. Дані можуть бути представлені рядком, масивом, зв'язаним списком, деревом пошуку, хеш-таблицею, неструктурованим набором тощо.

Вхідними даними пошукових алгоритмів є пошуковий ключ (поле запису, використовуючи значення якого, відбувається пошук) та функція, яка визначає відповідність знайденого результату заданому ключу. Це може бути значення булевої функції «так» або «ні», функція обчислення еквівалентності, функції релевантності та пертинентності тощо. Постановку завдання пошуку здійснюють за допомогою графового або аналітичного опису, який базуються на елементах теорії графів, моделей, алгебри та алгебраїчних систем.

Ефективність алгоритму пошуку залежить від типу використовуваних даних та методів їх обробки. Її оцінюють на основі складності алгоритму або максимального часу виконання. Складність визначають максимальною кількістю операцій порівняння пошукового ключа для найбільш песимістичного випадку. Це значення позначають як On, де n – кількість виконаних порівнянь. Варіанти виконання пошукових алгоритмів — найкращий, середній, найгірший варіанти. Значення середнього варіанту поведінки пошукового алгоритму визначає корисність алгоритму.

Виокремлюють також інформаційний пошук залежно від завдання: семантичний інформаційний пошук, пошук у структурах, пошук оптимального розв'язання, пошук шляху, пошук закономірностей у даних, пошук подібності (розбіжностей) у мультимедійних даних — пошук обличь, розпізнавання образів тощо. Алгоритми пошуку часто використовують разом з алгоритмами сортування.

Алгоритми пошуку значення масиві даних можуть мати ніяких особливих вимог до вхідним даним, і такі алгоритми ми порівняємо між собою. Якщо ж

про вхідні дані заздалегідь дещо відомо (наприклад, що вони впорядковані за алфавітом), то для таких даних можуть бути застосовані інші більш швидкодіючі алгоритми.

За допомогою розширених фільтрів, дозволяється розширити функціонал пошуку для користувача. Ввівши достатню кількість фільтрів, ми прискорюємо та покращуємо пошук в 1,5 разів мінімум.

Фрагмент коду розробки генерації відеоконтенту за допомогою фільтрації, зображено на рисунку 2.3.

```
@GetMapping("")
public String main(Map<String, Object> model) {

    model.put("movies", movieRepo.findByFilterMap(new HashMap<>()));
    model.put("genres", genreRepo.findAll());
    model.put("filters", new HashMap<>());

    return "movies/movies-search";
}

@GetMapping("/search")
public String search(@AuthenticationPrincipal User user,
                     @RequestParam(required = false) Map<String, String> form,
                     Map<String, Object> model) {
    List<Movie> movies;

//        movies = movieRepo.findAll();
    movies = movieRepo.findByFilterMap(form);

    model.put("movies", movies);
    model.put("filters", form);
    model.put("genres", genreRepo.findAll());


    return "movies/movies-search";
}
```

Рисунок 2.3 – Фрагмент коду реалізації методу генерації відеоконтенту

2.3 Розробка алгоритму програмного продукту

Важливим модулем даного веб застосунку є випадкова генерація нових рекомендацій фільмів для перегляду. На рисунку 2.4 зображена блок схема роботи алгоритму генерації псевдовипадкових чисел.

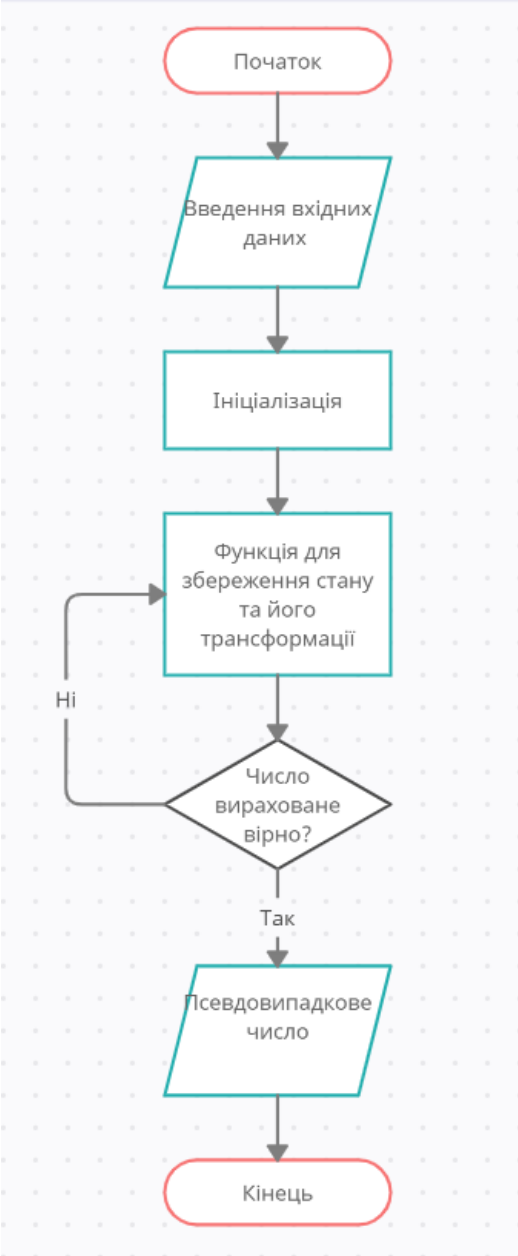

Рисунок 2.4 – Блок-схема алгоритму генерації псевдовипадкових чисел

Метод діаграми пріоритету (PDM) — це техніка візуального представлення, яка зображує дії, залучені в проект. Це метод побудови мережевої діаграми розкладу проекту, який використовує блоки/вузли для представлення діяльності та з’єднує їх за допомогою стрілок, які показують залежності.

Метод діаграми пріоритету (PDM) є інструментом для планування діяльності в плані проекту. Це метод побудови мережевої діаграми розкладу

проекту, який використовує блоки, які називаються вузлами, для представлення діяльності та з’єднує їх за допомогою стрілок, які показують залежності. Його також називають методом активності на вузлі (AON). На рисунку 2.5 зображено блок-схему авторизації користувача в системі.

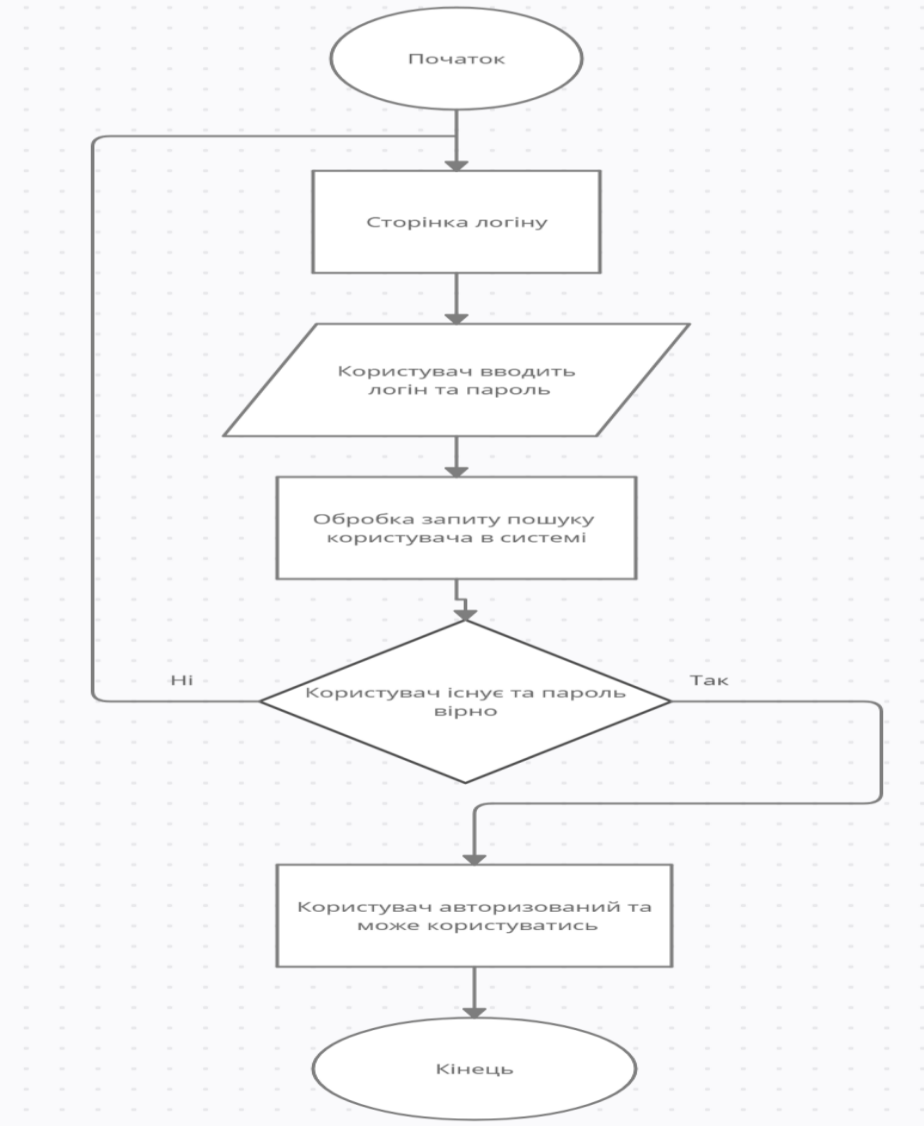

Рисунок 2.5 – Блок-схема авторизації користувача в системі

Діаграма прецедентів візуально зображає різноманітні сценарії взаємодії між акторами (користувачами) і прецедентами (випадками використання); описує функціональні аспекти системи (бізнес логіку).

Діаграми прецедентів відіграють важливу роль не тільки у комунікації між збирачами вимог до проекту і потенційними користувачами.

Діаграми Прецедентів дописані бізнес логікою і детальними специфікаціями прецедентів, як джерельна інформація, успішно використовують учасники розробки проекту на всіх його фазах (зародження, дизайн, програмування, тестування, документування).

Діаграма прецедентів, що зображена на рисунку 2.6 , показує взаємодію користувача з за стосунком, його можливі дії та реакції системи на них.

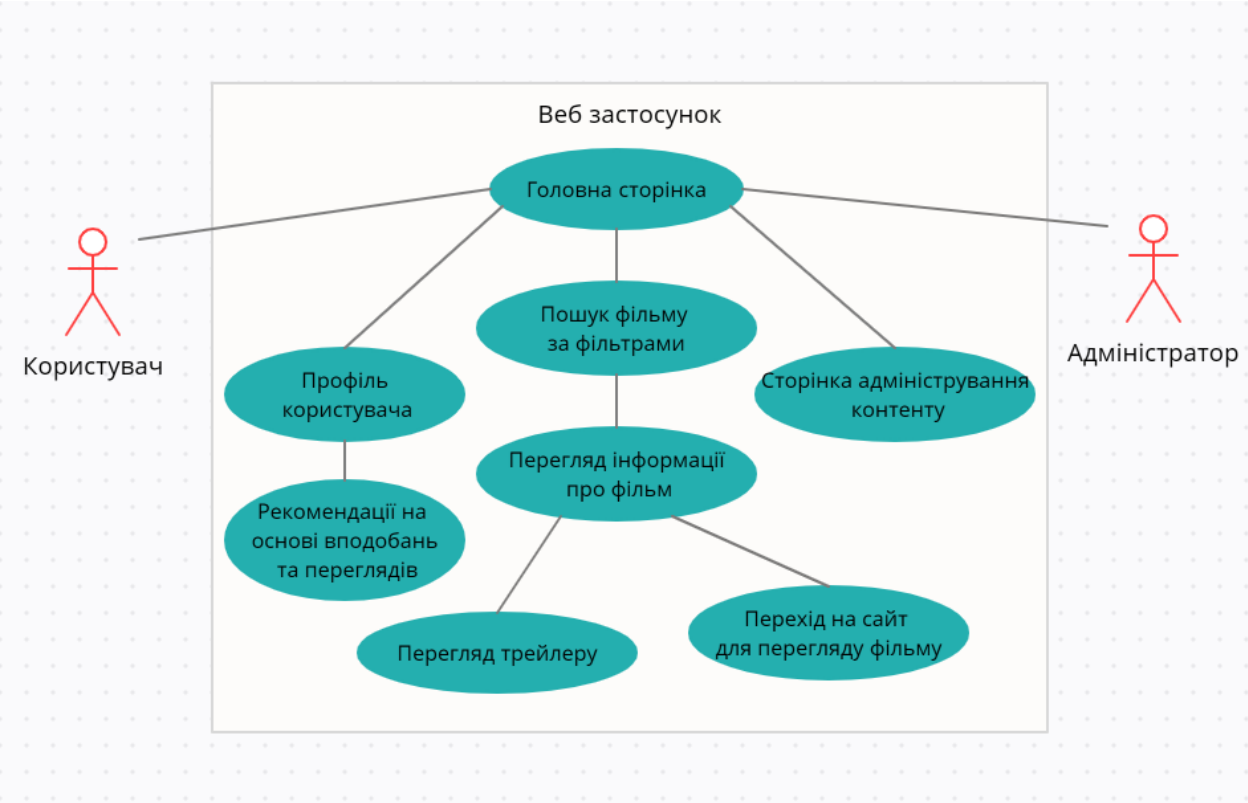

Рисунок 2.6 – Діаграма прецедентів веб-застосунку Movie Generator

Загальна UML діаграма класів веб застосунку зображена на рисунку 2.7. В ній зображені всі класи, що містять поля та методи

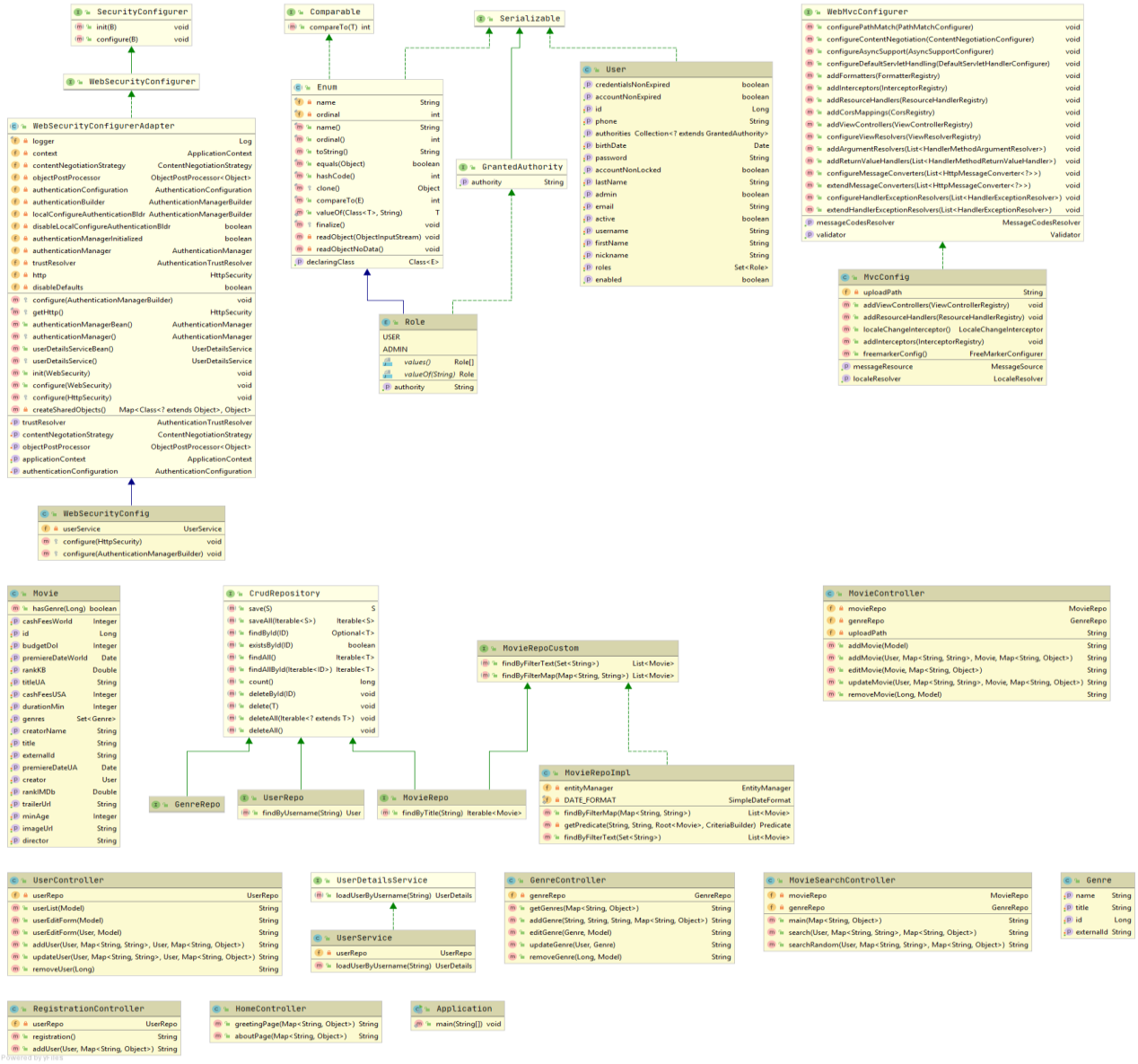

Рисунок 2.7 – Загальна UML діаграма класів веб застосунку

Таким чином дослідивши та удосконаливши метод генерації відеоконтенту, який за допомогою розширених фільтрів, дозволяє розширити функціонал пошуку для користувача. Ввівши достатню кількість фільтрів, ми зменшуємо час вибору та покращуємо пошук в 1,5 разів мінімум. По опитуваннях 28 людей, які використовували такий пошук, підтвердили це, та були зацікавленні.

2.4 Висновки

В даному розділі були реалізовані особливості та основні можливості веб застосунку. Розроблено структурну схему функціонування, діаграму класів, діаграму прецедентів та блок-схему алгоритму генерації псевдовипадкових чисел.

3 РОЗРОБКА СЕРВІСУ ГЕНЕРАЦІЇ ВІДЕКОНТЕНТУ НА ОСНОВІ ФІЛЬТРІВ

3.1 Обгрунтування вибору мови розробки

Перед початком створення будь-якого програмного засобу - необхідно визначитись з ресурсами та технологіями, які будуть використовуватись для розробки програмного продукту.

В принципі, написати програму можна на будь-якій мові. Питання в тому, чи буде вона працювати ефективно і без збоїв? Ось чому для вирішення різних завдань слід вибирати найбільш відповідні мови програмування.

Java — це мова програмування, створена в традиціях C і C++, але з іншою надією: «писати один раз, запускати всюди». Це мова, яка не залежить від платформи, тому програми можна запускати на багатьох різних типах комп’ютерів. Ще одна особливість, яка виділяє Java, — це її API, який є бібліотекою тисяч класів. Мова Java є спрощеною версією C++, але величезний API іноді ускладнює вивчення мови. Тим не менш, це добре відома мова для розробників програмного забезпечення та IT-фахівців.

Java є багатопоточною , що означає, що кілька завдань можна виконувати одночасно, а користувачі можуть створювати інтерактивні програми, які працюють безперебійно. Завдяки його безпечним функціям можна розробляти системи без вірусів і несанкціонованого доступу. Методи аутентифікації засновані на шифруванні з відкритим ключем.

Java - це об'єктно-орієнтована мова програмування і має платформу віртуальної машини, яка дозволяє створювати скомпільовані програми, які працюють майже на кожній платформі. Java пообіцяла: «Напишіть один раз, запустіть будь-де».

Незалежно від платформи, компілятор перетворює вихідний код у байт-код, а потім JVM виконує байт-код, згенерований компілятором. Цей байт-код може працювати на будь-якій платформі.

Організація програми в термінах колекції об’єктів – це спосіб об’єктно-орієнтованого програмування, кожен з яких представляє екземпляр класу. Існує 4 стовпи концепції ООП:

- Абстракція;
- інкапсуляція;
- спадкування;
- поліморфізм.

Java є однією з простих мов, оскільки вона не має складних функцій, таких як покажчики, перевантаження операторів, множинне спадкування, явне виділення пам’яті.

Вона розроблена таким чином, що докладає багато зусиль для перевірки помилок якомога раніше, тому компілятор java здатний виявити навіть ті помилки, які нелегко виявити іншою мовою програмування.

У java у нас немає покажчиків, тому ми не можемо отримати доступ до масивів поза межами, тобто він показує виключення ArrayIndexOutOfBound, якщо ми спробуємо це зробити.

Можна створювати розподілені програми, використовуючи мову програмування Java. Виклик віддаленого методу та Enterprise Java Beans використовуються для створення розподілених програм на java.

Java підтримує багатопотоковість. Це функція Java, яка дозволяє одночасно виконувати дві або більше частин програми для максимального використання ЦП.

JavaScript — це легка мова програмування («мова сценаріїв»), яка використовується для того, щоб зробити веб-сторінки інтерактивними. Він може вставляти динамічний текст у HTML. JavaScript також відомий як мова браузера. JavaScript(JS) не схожий або не пов’язаний з Java. Обидві мови мають синтаксис, подібний до C, і широко використовуються у клієнтських і серверних веб-додатках, але є лише кілька подібностей.

JavaScript був створений в першу чергу для маніпуляцій DOM. Раніше веб-сайти були переважно статичними, після створення JS були створені

динамічні веб-сайти. Функції в JS є об'єктами. Вони можуть мати властивості та методи, як і інший об’єкт. Їх можна передавати як аргументи в інших функціях. Може обробляти дату та час. Виконує перевірку форми, хоча форми створені за допомогою HTML. Не потрібен компілятор.

Подібно до серверних мов сценаріїв, таких як PHP і ASP, код JavaScript часто вставляється будь-де в HTML веб-сторінки. Вихідні дані на стороні сервера відображаються в HTML, але код JavaScript залишається видимим у вихідному коді веб-сторінки. Файл може бути окремим файлом «.js», який можна відобразити у браузері.

Незалежно від того, де ви розміщуєте JavaScript, він завжди виконується в клієнтському середовищі, щоб заощадити велику пропускну здатність і зробити процес виконання швидким.

У JavaScript XMLHttpRequest є важливим об’єктом, розробленим Microsoft. Виклик об’єкта, зроблений XMLHttpRequest як асинхронний HTTP-запит до сервера для передачі даних на обидві сторони без перезавантаження сторінки Найбільша перевага JavaScript у тому, що він підтримує всі сучасні браузери та забезпечує еквівалентний результат.

Глобальні компанії підтримують розвиток спільноти, створюючи важливі проекти. Прикладом може служити Google (створений фреймворк Angular) або Facebook (створен фреймворк React.js). JavaScript використовується всюди в Інтернеті.тJavaScript добре працює з іншими мовами і може бути використаний у величезних програмах. Існує багато проектів із відкритим кодом, які надають корисну допомогу при додаванні розробника JavaScript. Існує багато доступних курсів у галузі JavaScript, завдяки яким ви швидко та просто розширите свої знання цієї мови програмування.

Почати працювати в JavaScript не складно. З цієї причини багато хто з нас воліють починати свою пригоду в IT-секторі з вивчення цієї мови. Це дає можливість створювати багаті інтерфейси. Існує кілька способів використання JavaScript через сервери Node.js. Можна розробити цілу програму JavaScript від початку до кінця, використовуючи лише JavaScript. Недоліками JavaScript є.

Це може бути складно для розробки великих програм, хоча ви також будете використовувати накладення TypeScript.

Це стосується великих інтерфейсних проектів. Конфігурація часто буває виснажливим завданням щодо кількості інструментів, які потрібно об’єднати, щоб створити середовище для такого проекту. Це часто безпосередньо пов’язано з роботою бібліотеки. Основною проблемою або недоліком JavaScript є те, що код завжди видимий для всіх, хто може переглянути код JavaScript.

Незалежно від того, яку пропорцію швидко інтерпретувати JavaScript, JavaScript DOM (об’єктна модель документа) повільний і може ніколи не відображатися швидко з HTML. Якщо помилка виникає в JavaScript, він може припинити відтворення всього веб-сайту. Браузери надзвичайно толерантні до помилок JavaScript.

Зазвичай JavaScript інтерпретується різними браузерами по-різному. Це дещо ускладнює читання та запис міжбраузерного кодуХоча деякі редактори HTML підтримують налагодження, вони не настільки ефективні, як інші редактори, наприклад C/C++. Тому розробнику важко виявити проблему.

Ці безперервні перетворення займають більше часу для перетворення числа в ціле число. Це збільшує час, необхідний для виконання сценарію, і зменшує його швидкість.

Ruby — це чиста об’єктно-орієнтована мова, розроблена Юкіхіро Мацумото (також відомим як Matz у спільноті Ruby) у середині 1990-х років у Японії. Усе в Ruby є об’єктом, за винятком блоків, але для нього також є заміни, тобто procs та lambda. Мета розробки Ruby полягала в тому, щоб змусити його діяти як розумний буфер між людьми-програмістами та базовою обчислювальною технікою. Ruby має синтаксис, подібний до синтаксису багатьох мов програмування, таких як C і Java, тому програмістам Java та C його легко вивчити. Він підтримує в основному всі платформи, такі як Windows, Mac, Linux.

Ruby заснований на багатьох інших мовах, таких як Perl, Lisp, Smalltalk, Eiffel і Ada. Це інтерпретована мова сценаріїв, що означає, що більшість його

реалізацій виконують інструкції безпосередньо і вільно, без попередньої компіляції програми в інструкції машинною мовою. Програмісти Ruby також мають доступ до потужних RubyGems (RubyGems надає стандартний формат для програм і бібліотек Ruby).

Програмування на Ruby легко навчитися через його синтаксис, схожий на вже широко використовувані мови.

Код, написаний на Ruby, невеликий, елегантний і потужний, оскільки має меншу кількість рядків коду.

Ruby дозволяє просто і швидко створювати веб-додатки, що призводить до менш важкої роботи. Оскільки Ruby є безкоштовним, тобто Ruby може вільно копіювати, використовувати, змінювати, він дозволяє програмістам вносити необхідні зміни в разі потреби.

Ruby — це динамічна мова програмування, завдяки якій немає жорстких правил щодо вбудовування функцій і вона дуже близька до розмовних мов.

Ruby є досить новим і має власну унікальну мову кодування, що ускладнює для програмістів кодування на ньому відразу, але після певної практики його легко використовувати. Багато програмістів вважають за краще дотримуватися того, що вони вже знають і можуть розробити.

Код, написаний на Ruby, важче налагоджувати, оскільки більшу частину часу він генерується під час виконання, тому його стає важко читати під час налагодження.

Ruby не має великої кількості інформаційних ресурсів у порівнянні з іншими мовами програмування.

Ruby — це інтерпретована мова сценаріїв, мови сценаріїв зазвичай повільніші, ніж скомпільовані, тому Ruby повільніша, ніж багато інших мов. Ruby і Java є об'єктно-орієнтованими мовами, а також вони сильно типізовані. Java типізується статично, тоді як Ruby — динамічно.

Обидві мови мають різні методи виконання коду. Java спочатку перетворює код на машинну мову, щоб вона могла його зрозуміти, і через це код Java працює швидше, ніж код Ruby. І Java, і Ruby забезпечують

успадкування, і обидва мають публічні, приватні та захищені методи. Функції в Ruby займають менше рядків коду, ніж Java, через що розробники та програмісти віддають перевагу Ruby.

Тобто він може використовуватись і використовується для вирішення різних завдань у різних галузях. Тим не менш, у Ruby є ніша, в якій він використовується найчастіше. Це веб-розробка.

Ruby стала популярною мовою для створення веб-додатків завдяки Ruby on Rails. Він вплинув на веб-розробку загалом та інші фреймворки, зокрема, про що нижче розповідають експерти.

У «рейках» реалізовані інноваційні можливості, включаючи безшовну інтеграцію з базою даних, міграцію, створення уявлень для прискорення розробки. Ці можливості пізніше були впроваджені в інших фреймворках, включаючи Django, Laravel та Phoenix.

У Ruby on Rails застосовується архітектура MVC, а також відомі інженерні патерни, включаючи DRY, ActiveRecord, convention over configuration (угода конфігурації). Принцип угоди про конфігурацію продовжує принцип найменшого подиву, який використовував Юкіхіро Матцумото при розробці мови Ruby. Convention over configuration означає, що конфігурація необхідна тільки там, де будь-який аспект виходить за межі специфікації.

Ruby застосовується не лише у веб-розробці. На Ruby написані утиліта командного рядка Homebrew, програмне забезпечення для інформаційної безпеки Metasploit, програмне забезпечення для створення віртуального середовища розробки Vagrant та інші відомі програми.

Нижче наведено списки точок, що описують ключові відмінності Java від Ruby.

– java має бути скомпільована перед запуском програми, тоді як у Ruby немає необхідності компілювати код;

– тільки класи є об'єктами, тоді як у Ruby є Object;

– змінні Java статично типізовані, тоді як у Ruby змінні динамічно типізовані;

– змінні-члени мають ідентифікатори доступу (Private, Public і Protected) у Java, тоді як у Ruby за замовчуванням усі змінні-члени закриті;

– оголошення нульового значення відрізняється як і Java, і у Ruby, оскільки воно оголошується різними ключовими словами, т. е. нульове значення оголошується з «null» у Java, а Ruby воно оголошується з ключовим словом «nil»;

Приведення в Java, об'єкти можуть бути перетворені на інші об'єкти, якщо об'єкти, які будуть перетворені, відносяться до типу об'єктів, до яких наводяться. Але Ruby приведення не використовується, оскільки змінні динамічно типізовані, а також присвоюються будь-якому іншому типу. Порівняння мов розробки відображено в таблиці 3.1.

Таблиця 3. – Порівняльний аналіз мов розробки

Критерії	Java	Ruby	JavaScript
JVM	Будь яка або в браузері	Багатоплатормова	Браузер або Node.js
Об’єктно-орієнтованість	Об’єктно-орієнтовна	Об’єктно-орієнтовна	Мова сценаріїв на основі об’єктів
Багатопотоковість	Підтримує	Підтримує	Не підтримує
Типізація	Сильно типізована	Динамічна	Слабо типізована
Паралельність	Має потоковий підхід до паралельності	Має підхід до паралельності на основі подій	Має підхід до паралельності на основі подій

З огляду на порівняльний аналіз мов програмування, зрозуміло, що Java є найкращим вибором для розробки веб додатку. Окрім ряду переваг над аналогами, мова програмування що використовується для розробки клієнтської та серверної частин є однаковою, що значно спрощує процес розробки.

3.2 Обгрунтування вибору середовища розробки

Після вибору мови програмування, за допомогою якої буде виконуватись розробка програмного засобу, необхідно вибрати IDE.

Середовище розробки програмного забезпечення (SDE) - це середовище, яке автоматизує або розширює підпрограми, що беруть участь у циклі розробки програмного забезпечення. Це включає в себе програмування в багатьох завданнях, таких як управління командою та проектами, а також великі завдання програмування, такі як управління конфігурацією. SDE також підтримує масштабне та довготривале обслуговування програмного забезпечення.

Основними функціями IDE є: IDE повинна мати можливість заповнення коду для ідентифікації функцій мови та ключового слова Java. Вона повинна мати потужне управління ресурсами, яке допомагає виявити відсутні ресурси, заголовки, бібліотеки тощо. Хороший інструмент налагодження для повного тестування розробленого додатка. Компілювати та будувати функції.

IDE займає дуже мало часу та зусиль, оскільки вся концепція IDE полягає в тому, щоб спростити та прискорити розробку.

Це відповідає певним стандартам компанії, отже, принцип роботи буде незмінним протягом усього і допомагає кодерам.

Вона постачається з хорошими інструментами управління проектами та документами для автоматизації багатьох речей.

Корисно для спрощення розробки додатків баз даних.

Вона має функції для розробки хорошого інтерфейсу користувача з текстовими полями, кнопками тощо[15].

Прийняття рішення про те, яка IDE чи редактор відповідає нашим потребам, залежить від різних факторів, включаючи характер проектів або додатків, що розробляються, процес, що використовується командою розробників, рівень особистості та навички програміста, а також роль в

організації. Особисті уподобання та стандартизація інструментів також відіграють важливу роль у виборі IDE або редактора.

Головною перевагою використання IDE для розробки є те, що коли компілятор інтегрований з IDE, ми отримуємо весь пакет в одному місці, щоб ми могли заповнити код, скомпілювати, налагодити та виконати програму в тому самому програмному забезпеченні.

IDE мають привабливий користувальницький інтерфейс і комплектуються всіма елементами розробки програмного забезпечення, які ми можемо використовувати для розробки програмних додатків[15].

IntelliJ IDEA - це середовище розробки для розробки програмних додатків з використанням Java. IntelliJ IDEA була розроблена компанією JetBrains. Він доступний як ліцензована версія спільноти Apache 2 та у власній комерційній версії. Обидва видання можуть бути використані для комерційного розвитку. На рисунку 3.3 зображено вікно IntelliJ IDEA.

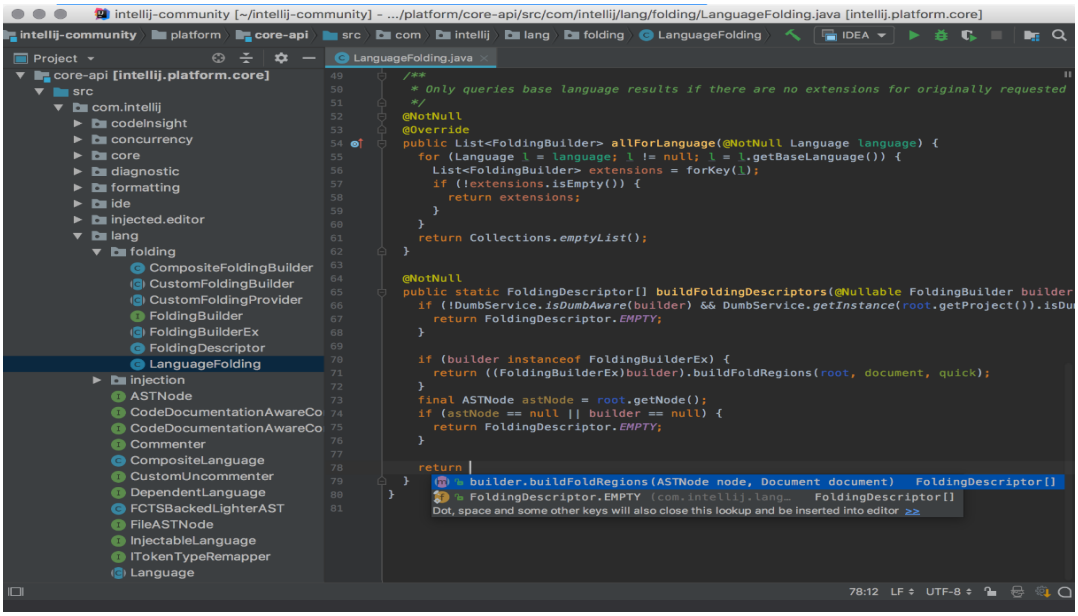

Рисунок 3.3 – Вікно IntelliJ IDEA

Він дає поради щодо заповнення коду, аналізу коду та надійних інструментів рефакторинге. Він має критично важливі інструменти, такі як система контролю версій, підтримка багатьох мов та фреймворків. Він

здатний слідувати контексту розробника та автоматично запускає відповідні інструменти.

У ньому наведено перелік найбільш відповідних символів, що застосовуються до поточного контексту. Він постійно переміщує найновіші класи, методи тощо тощо у верхній частині списку пропозицій. Таким чином, заповнення коду відбувається швидше. IntelliJ має можливість аналізувати потік даних і вгадувати можливий символ під час виконання.

Ви можете легко включити до коду Java фрагменти іншої мови, наприклад - SQL.

IntelliJ пропонує ретельний та ефективний рефакторинг, оскільки він знає все про використання символів. IntelliJ Idea постачається з широким розмаїттям вбудованих інструментів, таких як GIT, контроль версій, декомпілятор, покриття, база даних SQL тощо.

Має потужний компілятор, який здатний виявляти дублікати, запахи коду тощо. Він має сильну інтеграцію з серверами додатків.

Плюси:

– intelliJ Idea добре знаходить повторювані блоки коду та показує помилки перед компіляцією;

– він має потужну функцію налаштування для зміни структури проекту відповідно до потреб користувача;

– хороший інтерфейс з великою кількістю варіантів тем.

Мінуси:

– документація щодо інструментів потребує вдосконалення;

– висока ціна на корпоративну версію, а іноді IDE виходить з ладу, якщо це величезний додаток[15].

Eclipse - це повнофункціональна потужна IDE із відкритим кодом, яка широко використовується для розробки додатків Java. Eclipse оснащений базовим робочим середовищем та розширюваною системою плагінів, за допомогою якої ми можемо налаштувати

середовище. Пишеться здебільшого на Java. На рисунку 3.4 зображено вікно Eclipse.

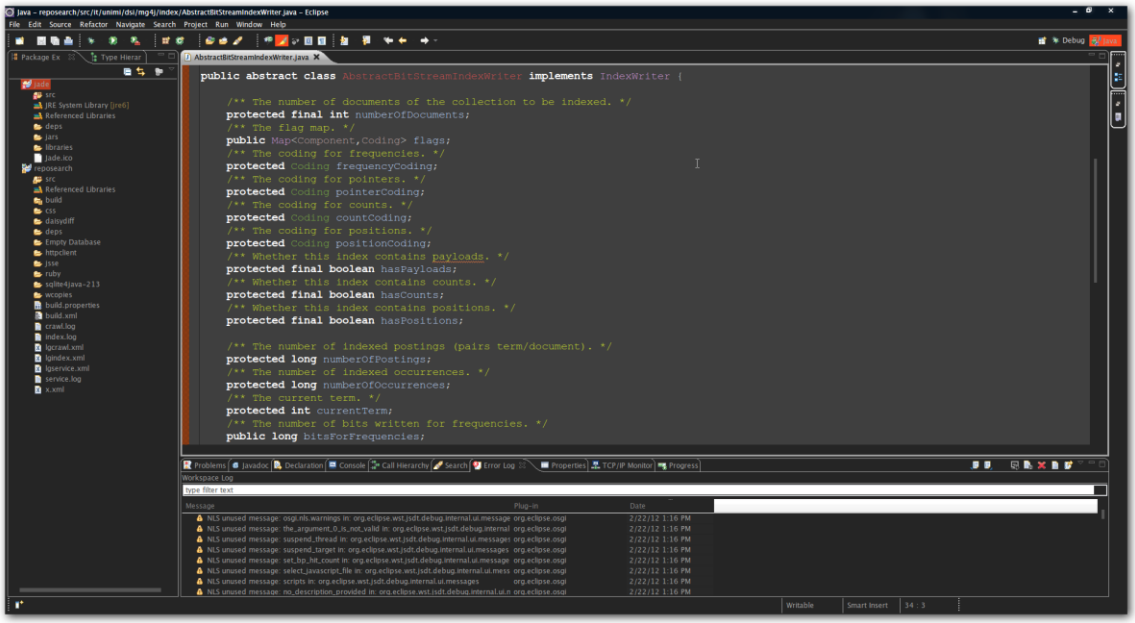

Рисунок 3.4 – Вікно Eclipse

Оскільки він відкритий, він допомагає розробникам налаштувати рішення та зробити додаток більш надійним. Він заснований на базовій основі Java, і, отже, робить себе дуже розширюваним, гнучким і сумісним з багатьма мовами, такими як C ++, Groovy, Python, Perl, C # тощо. Це робить його найкращим вибором для розробників.

Eclipse є крос-платформним і працює на Linux, Mac OS та Windows. Підтримка розширених інструментів. Редагування, перегляд, рефакторинг та налагодження: Eclipse надає всі ці функції та полегшує програмістам розробку додатків.

Eclipse підтримує налагодження як локально, так і віддалено, припускаючи, що ви використовуєте JVM, що підтримує віддалену налагодження. Eclipse має велику допомогу та документацію.

Eclipse має власний ринок, що дозволяє користувачеві завантажувати клієнтські рішення. Він має хороший робочий простір, що дозволяє

розробникам легко ідентифікувати проекти, папки та файли. Він має настійну рекомендацію та функцію налагодження для помилок.

Це дозволяє інтегрувати з сервером Apache Maven та контролем версій Git. Це стандартний віджет з підтримкою Gradle. Eclipse має хорошу можливість інтеграції для створення таких інструментів, як ANT та Maven.

Користувачі можуть розробляти різні додатки на одній платформі, такі як веб- та автономні програми, веб-служби тощо. Надійні рекомендації щодо коду та налагоджувачі вбудовані в Eclipse.

Мінуси:

– eclipse постачається з великою кількістю перевірок для файлів JSP та HTML;

– початкове налаштування часом стає важким без належних вказівок та документації[15].

NetBeans - це безкоштовне інтегроване середовище розробки з відкритим кодом, яким керує Apache Software Foundation. Корисно розробляти веб-додатки, настільні, мобільні, C ++, HTML 5 тощо. NetBeans дозволяє розробляти додатки із набору модульних програмних компонентів, які називаються модулями. NetBeans працює на Windows, Mac OS, Linux та Solaris.

Він поставляється разом з гарною архітектурою та вбудованими інструментами, які додають цінності до повної SDLC від вимог до проекту до розгортання. Він має активну спільноту користувачів та розробників у всьому світі. Він містить різні модулі, за допомогою яких функції виконуються добре. Він пропонує плавне і швидке редагування коду.

NetBeans - це мовний редактор, тобто він виявляє помилки, коли програміст час від часу набирає та допомагає в спливаючих вікнах документації та інтелектуальному заповненні коду. Інструмент

рефакторингу NetBeans дозволяє програмісту перебудувати код, не порушуючи його.

NetBeans також виконує аналіз вихідного коду та надає широкий набір підказок для вдосконалення коду або швидкого його виправлення. Він включає інструмент проектування графічних інтерфейсів Swing, раніше відомий як “Project Matisse”. На рисунку 3.5 зображено вікно NetBeans.

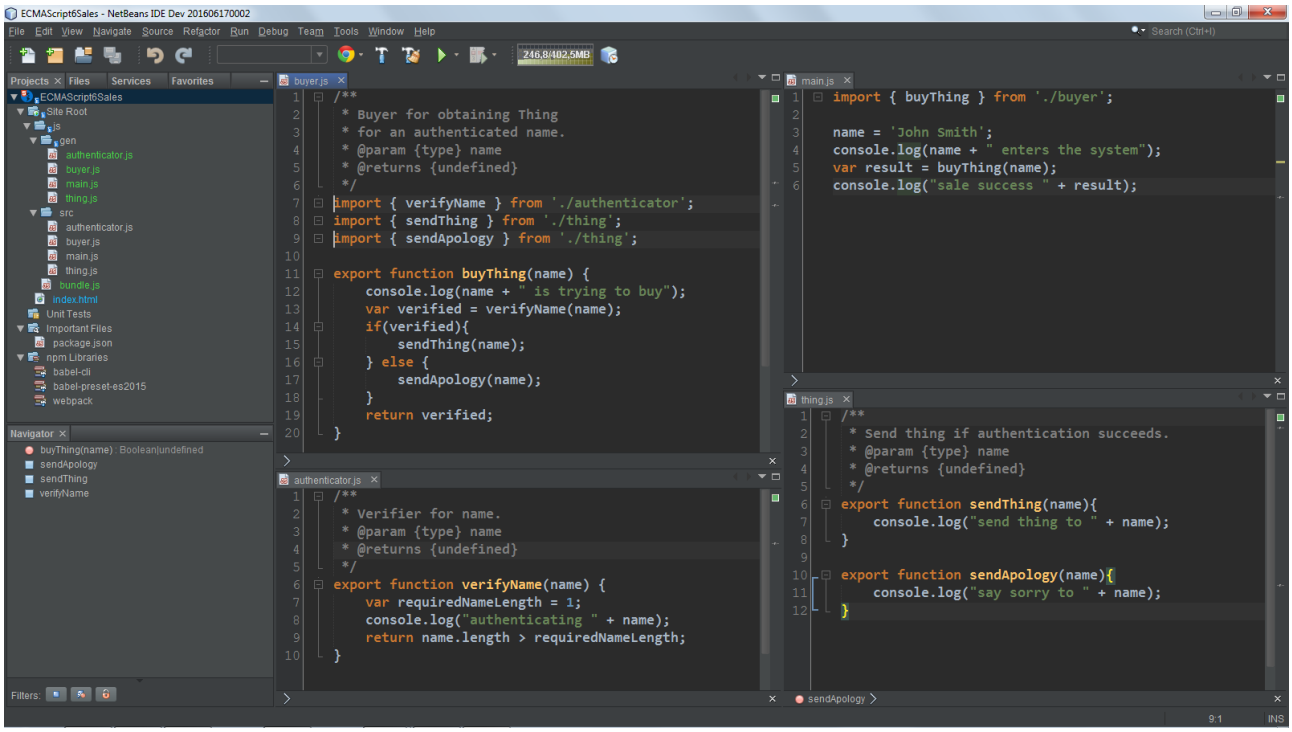

Рисунок 3.5 – Вікно NetBeans

Він також має вбудовану підтримку Maven та Ant, а також плагін для Gradle. NetBeans пропонує хорошу підтримку між платформами та багатомовними мовами. Він має багатий набір спільноти, яка надає плагіни. Він має дуже просту та легку функцію управління проектами, тому розробники використовують її в повній мірі.

Його консоль пропонує дуже швидке та розумне редагування коду в середовищі розробки. Він також постачається із засобом статичного аналізу та перетворювачами коду.

Плюси:

– netbeans дозволяє розробникам розгортати код із власного середовища;

– користувачі можуть форматувати та визначати правила для всіх мов;

– він також має паралельну функцію порівняння коду, завдяки якій подібні сторінки можна писати одночасно.

Мінуси:

– через великі розміри інструменту іноді він стає повільним в обробці. Тож бажано мати полегшену версію;

– плагіни, надані NetBeans для розробки IOS та Android, можна вдосконалити[15].

Таким чином порівнявши середовища розробки, було обрано найкраще для виконання поставленої задачі, це IntelliJ IDEA, воно є найкращим для програмування.

3.3 Програмна реалізація засобу сервісу генерації відеоконтенту, із використанням розширених фільтрів

Опис серверної частини мовою Java.

Виходячи з аналізу предметної області, для роботи веб застосунку було виділено наступні сутності для зберігання інформації у базі даних:

1) Movie – відповідає за збереження даних про фільм, його оригінальну назву, українську, рік випуску, тривалість, тощо. Програмну реалізацію наведено на рисунку 3.6.

```java
@Entity
public class Movie {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    private String externalId;

    private String title;

    private String titleUA;

    private Integer durationMin;

    private Integer minAge;

    private Integer budgetDol;

    private Integer cashFeesWorld;

    private Integer cashFeesUSA;

    @DateTimeFormat(pattern = "yyyy-MM-dd")
    private Date premiereDateWorld;
    @DateTimeFormat(pattern = "yyyy-MM-dd")
    private Date premiereDateUA;

    private Double rankKB;

    private Double rankIMDb;

    private String imageUrl;

    private String trailerUrl;

    private String director;

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "user_id")
    private User creator;
```

Рисунок 3.6 – Програмна реалізація сутності Movie

2) User – відповідає за збереження інформації про користувача, його логін, пароль, ім’я, прізвище, пошту, тощо. Програмну реалізацію наведено на рисунку 3.7.

```java
@Entity
@Table(name = "usr")
public class User implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    private String username;

    private String password;

    private boolean active;

    private String nickname;

    private String email;

    private String firstName;

    private String lastName;

    private String phone;

    @DateTimeFormat(pattern = "yyyy-MM-dd")
    private Date birthDate;

    @ElementCollection(targetClass = Role.class, fetch = FetchType.EAGER)
    @CollectionTable(name = "user_role", joinColumns = @JoinColumn(name = "user_id"))
    @Enumerated(EnumType.STRING)
    private Set<Role> roles = new HashSet<>();
```

Рисунок 3.6 – Програмна реалізація сутності User

3) Genre – відповідає за збереження інформації про жанр, його назву та інші атрибути. . Програмну реалізацію наведено на рисунку 3.8.

```java
@Entity
public class Genre {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    private String name;

    private String title;

    private String externalId;

    public Genre() {
    }
```

Рисунок 3.8 – Програмна реалізація сутності Genre

4) Role – відповідає за збереження інформації про роль користувача в системі. Програмну реалізацію наведено на рисунку 3.9.

```java
public enum Role implements GrantedAuthority {
    USER,
    ADMIN;

    @Override
    public String getAuthority() {
        return name();
    }
}
```

Рисунок 3.9 – Програмна реалізація сутності Role

Розроблені сутності дозволяють зручно працювати з предметною областю та виконують усі поставлені функції. Також при модифікації застосунку та з розширенням функціоналу їх буде легко масштабувати та доповнити необхідним функціоналом.

Для обробки клієнтських веб-запитів було розроблено наступні основні класи програми:

1) MovieController – відповідає за обробку всіх запитів, які пов'язані з обробкою даних про фільми, такі як отримання, створення, редагування та видалення. Його програмний інтерфейс наведений на рисунку 3.10.

```
@Controller
@PreAuthorize("hasAuthority('ADMIN')")
@RequestMapping("/movies")
public class MovieController {

    @Autowired
    private MovieRepo movieRepo;

    @Autowired
    private GenreRepo genreRepo;

    @Value("${upload.path}")
    private String uploadPath;

    @GetMapping("/add")
    public String addMovie(Model model) {...}

    @PostMapping(path = "/add", consumes = {MediaType.APPLICATION_FORM_URLENCODED_VALUE})
    public String addMovie(@AuthenticationPrincipal User user,
                           @RequestParam(required = false) Map<String, String> form,
                           Movie newMovie,
                           Map<String, Object> model) throws IOException {...}

    @GetMapping("/edit/{movie}")
    public String editMovie(@PathVariable Movie movie, Map<String, Object> model) {...}

    @PostMapping(path = "/update", consumes = {MediaType.APPLICATION_FORM_URLENCODED_VALUE})
    public String updateMovie(@AuthenticationPrincipal User user,
                              @RequestParam(required = false) Map<String, String> form,
                              Movie movie,
                              Map<String, Object> model) throws IOException {...}

    @GetMapping("/remove/{movieId}")
    public String removeMovie(@PathVariable Long movieId, Model model) {...}

}
```

Рисунок 3.10 – Програмна реалізація інтерфейсу класу MovieController

2) UserController – відповідає за обробку клієнтських веб-запитів, які пов'язані з обробкою даних про користувача, такі як отримання, редагування,

видалення та створення. Його програмний інтерфейс наведений на рисунку 3.11.

```
@Controller
@RequestMapping("/users")
@PreAuthorize("hasAuthority('ADMIN')")
public class UserController {
    @Autowired
    private UserRepo userRepo;

    @GetMapping
    public String userList(Model model) {...}

    @GetMapping("/add")
    public String userEditForm(Model model) {...}

    @GetMapping("/edit/{user}")
    public String userEditForm(@PathVariable User user, Model model) {...}

    @PostMapping(path = "/add", consumes = {MediaType.APPLICATION_FORM_URLENCODED_VALUE})
    public String addUser(@AuthenticationPrincipal User u,
                          @RequestParam(required = false) Map<String, String> form,
                          User user,
                          Map<String, Object> model) {...}

    @PostMapping(path = "/update", consumes = {MediaType.APPLICATION_FORM_URLENCODED_VALUE})
    public String updateUser(@AuthenticationPrincipal User u,
                             @RequestParam(required = false) Map<String, String> form,
                             User user,
                             Map<String, Object> model) {...}

    @GetMapping("/remove/{userId}")
    public String removeUser(@PathVariable Long userId) {...}

}
```

Рисунок 3.11 – Програмна реалізація інтерфейсу класу UserController

3) GenreController – відповідає за обробку клієнтських веб-запитів, які пов’язані з обробкою даних про жанри фільмів, такі як отримання, редагування, видалення та створення. Його програмний інтерфейс наведений на рисунку 3.12.

```
@Controller
@RequestMapping("/genres")
@PreAuthorize("hasAuthority('ADMIN')")
public class GenreController {

    @Autowired
    private GenreRepo genreRepo;

    @GetMapping("")
    public String getGenres(Map<String, Object> model) {...}

    @PostMapping("/add")
    public String addGenre(@RequestParam String name,
                           @RequestParam String title,
                           @RequestParam(required = false) String externalId,
                           Map<String, Object> model) throws IOException {...}

    @GetMapping("/edit/{genre}")
    public String editGenre(@PathVariable Genre genre, Model model) {...}

    @PostMapping("/update")
    public String updateGenre(@AuthenticationPrincipal User user,
                              Genre genre) {...}

    @GetMapping("/remove/{movieId}")
    public String removeGenre(@PathVariable Long movieId, Model model) {...}
}
```

Рисунок 3.12 – Програмна реалізація інтерфейсу класу GenreController

4) RegistrationController –відповідає за обробку клієнтських веб-запитів, які пов’язані з реєстрацією нового користувача в системі. Його програмний інтерфейс наведений на рисунку 3.13.

```
@Controller
public class RegistrationController {

    @Autowired
    private UserRepo userRepo;

    @GetMapping("/registration")
    public String registration() { return "users/registration"; }

    @PostMapping("/registration")
    public String addUser(User user, Map<String, Object> model) {...}
}
```

Рисунок 3.13 – Програмна реалізація інтерфейсу класу RegistrationController

5) MovieSearchController – відповідає за обробку клієнтських веб-запитів, які пов’язані з пошуком фільмів за певними фільтрами. Його програмний інтерфейс наведений на рисунку 3.14.

```
@Controller
@RequestMapping("/movies")
public class MovieSearchController {

    @Autowired
    private MovieRepo movieRepo;

    @Autowired
    private GenreRepo genreRepo;

    @GetMapping("")
    public String main(@RequestParam(required = false, defaultValue = "") String filter,
                       Map<String, Object> model) {...}

    @GetMapping("/search")
    public String main(@AuthenticationPrincipal User user,
                       @RequestParam(required = false) Map<String, String> form,
                       Map<String, Object> model) {...}
}
```

Рисунок 3.14 – Програмна реалізація інтерфейсу класу MovieSearchController

6) HomeController – відповідає за обробку клієнтських веб-запитів, які пов’язані зі сторінками «Головна» та «Про сайт». Його програмний інтерфейс наведений на рисунку 3.15.

```
@Controller
public class HomeController {

    @GetMapping("/")
    public String greetingPage(Map<String, Object> model) { return "greeting"; }

    @GetMapping("/about")
    public String aboutPage(Map<String, Object> model) { return "about"; }
}
```

Рисунок 3.15 – Програмна реалізація інтерфейсу класу HomeController

Класи GenreRepo, MovieRepo, MovieRepoCustom, MovieRepoImpl, UserRepo відповідають за роботу з базою даних, з їх допомогою формуються sql запити до бази даних. Програмний код даних класів наведений у додатку.

Класи WebSecurityConfig, MvcConfig, Application відповідають за загальне налаштування веб-застосунку. В них задаються основні параметри та конфігурації застосунку, такі як авторизація, локалізація, конфігурація шаблонізатора тощо. Їх програмний інтерфейс наведено на рисунку 3.16.

```java
@Configuration
public class MvcConfig implements WebMvcConfigurer {

    @Value("${upload.path}")
    private String uploadPath;

    public void addViewControllers(ViewControllerRegistry registry) {...}

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {...}

    @Bean(name = "localeResolver")
    public LocaleResolver getLocaleResolver() {...}

    @Bean
    public LocaleChangeInterceptor localeChangeInterceptor() {...}

    @Override
    public void addInterceptors(InterceptorRegistry registry) { registry.addInterceptor(localeChangeInterceptor()); }

    @Bean(name = "messageSource")
    public MessageSource getMessageResource() {...}

    @Bean(name = "freemarkerConfig")
    public FreeMarkerConfigurer freemarkerConfig() {...}

}
```

```java
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private UserService userService;

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

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {...}
}
```

```java
@SpringBootApplication
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

Рисунок 3.16 – Програмна реалізація інтерфейсу класів WebSecurityConfig, MvcConfig, Application

Для розробки клієнтського інтерфейсу використовувався фреймворк Freemarker. Це зручний шаблонізатор, який дозволяє використовувати Java об’єкти при динамічній побудові html сторінки.

Для придання html сторінкам приємного та зручного вигляду використовувався фреймворк для роботи зі стилями Bootstrap. Він дозволяє швидко налаштувати стилі для головних компонентів сторінки використовуючи зручний інтерфейс для роботи.

На рисунку 3.17 зображено приклад файлу шаблону.

```
<#macro login path isRegisterForm>
    <form action="${path}" method="post">
        <div class="form-group-row mb-3 d-flex justify-content-center">
            <div class="col-sm-1">
                <label class="col-form-label"><@s.message "msg.login.username.label"/>:</label>
            </div>
            <div class="col-sm-3">
                <input type="text" name="username" class="form-control"/>
            </div>
        </div>
        <div class="form-group-row mb-3 d-flex justify-content-center">
            <div class="col-sm-1">
                <label class="col-form-label"><@s.message "msg.login.password.label"/>:</label>
            </div>
            <div class="col-sm-3">
                <input type="password" name="password" class="form-control"/>
            </div>
        </div>
        <div class="form-group-row mb-3 d-flex justify-content-center">
            <input type="hidden" name="_csrf" value="${_csrf.token}"/>
            <div class="col-sm-1">
                <button type="submit" class="btn btn-primary"><#if isRegisterForm><@s.message "msg.login.btn.register"/><#else>
            </div>
            <div class="col-sm-1">
                <#if !isRegisterForm><a href="/registration" class=""><@s.message "msg.login.btn.registration"/></a></#if>
            </div>
        </div>
    </form>
</#macro>

<#macro logout>
    <form action="/logout" method="post">
        <input type="hidden" name="_csrf" value="${_csrf.token}"/>
        <button type="submit" class="btn btn-primary"><@s.message "msg.login.btn.logout"/></button>
    </form>
</#macro>
```

Рисунок 3.17 – Лістинг файлу шаблону

Для всіх веб-сторінок та підкомпонетів веб-застосунку було відповідно створені шаблони відображення, вони зручно поділені у зрозумілій файловій структурі, що зображена на рисунку 3.18.

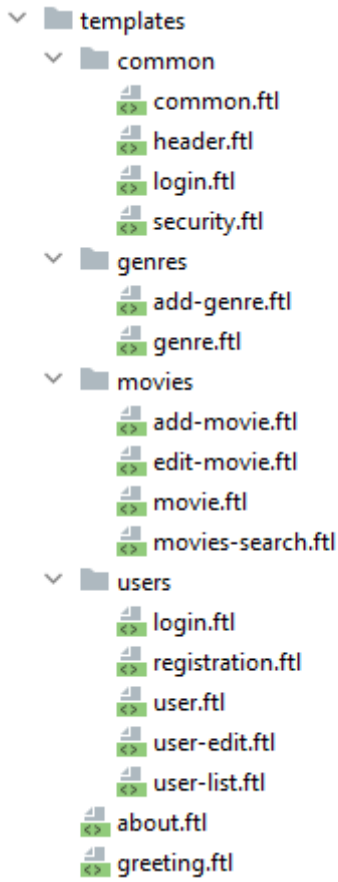

Рисунок 3.18 – Файлова структура компонентів

Для локалізації веб-застосунку та роботі на різних мовах, всі статичні рядкові значення в html сторінках були винесені у файли messages.properties. З їх допомогою можна з легкістю налаштувати можливість динамічного переключення мови інтерфейсу додатку. Фрагмент з такого файлу наведено на рисунку 3.19.

```
msg.user.btn.add=Додати

msg.movie.title.list=Фільми

msg.movie.title.title=Назва в оригіналі
msg.movie.title.titleUA=Назва українською
msg.movie.title.director=Режисер
msg.movie.title.durationMin=Тривалість (хв.)
msg.movie.title.minAge=Вікові обмеження
msg.movie.title.minAge.plus=+
msg.movie.title.budgetDol=Бюджет
msg.movie.title.cashFeesWorld=Касові збори в світі
msg.movie.title.cashFeesUSA=Касові збори в США
msg.movie.title.premiereDateWorld=Прем'єра в світі
msg.movie.title.premiereDateUA=Прем'єра в Україні
msg.movie.title.externalId=Зовнішній ІД
msg.movie.title.rankKB=Рейтинг КіноБаза
msg.movie.title.rankIMDb=Рейтинг IMDb
msg.movie.title.imageUrl=Обкладинка (посилання)
msg.movie.title.trailerUrl=Трейлер (посилання)
msg.movie.title.genres=Жанри

msg.movie.btn.add=Додати фільм
msg.movie.btn.save=Зберегти фільм
msg.movie.btn.trailer=Трейлер

msg.genre.title.list=Жанри

msg.genre.title.name=Назва англійською
msg.genre.title.title=Назва українською
msg.genre.title.externalId=Зовнішній ІД

msg.genre.btn.add=Додати жанр
msg.genre.btn.save=Зберегти жанр
```

Рисунок 3.19 – Фрагмент коду динамічного переключення мови інтерфейсу

Деякі компоненти на стороні клієнта були реалізовані на мові Java Script. Наприклад для перегляду трейлеру фільму прямо на сайті. Фрагмент програмної реалізації даного функціоналу наведений на рисунку 3.20.

```
function autoPlayYouTubeModal() {
    var triggerOpen = $("body").find('[data-tagVideo]');
    triggerOpen.click(function() {
        var theModal = $(this).data("bs-target"),
            videoSRC = $(this).attr("data-tagVideo"),
            videoSRCauto = videoSRC + "?autoplay=1";
        $(theModal + ' iframe').attr('src', videoSRCauto);
        $(theModal + ' button.btn-close').click(function() {
            $(theModal + ' iframe').attr('src', '');
        });
    });
}
```

Рисунок 3.20 – Фрагмент програмної реалізації перегляду трейлеру

Одним з головних частин коду є (рис 3.21) – за допомогою цього коду, відбувається випадковий пошук фільмів, без вказівки додаткових фільтрів та виводиться користувачу.

```
    @GetMapping("/search-random")
    public String searchRandom(@AuthenticationPrincipal User user,
                               @RequestParam(required = false) Map<String, String> form,
                               Map<String, Object> model) {
        List<Movie> movies;

//          movies = movieRepo.findAll();
        movies = movieRepo.findByFilterMap(form);

        if(movies != null && !movies.isEmpty()) {
            Random rnd = new Random();
            int upperBound = movies.size();
            int movieIndex = rnd.nextInt(upperBound);
            Movie movie = movies.get(movieIndex);
            movies = Collections.singletonList(movie);
        }
        model.put("movies", movies);
        model.put("filters", form);
        model.put("genres", genreRepo.findAll());

        return "movies/movies-search";
    }

}
```

Рисунок 3.21 – Фрагмент програмної реалізації пошуку відеоконтенту

За допомогою фільтрів формується запит, знаходить декілька фільмів, а потім з них випадковим чином видає один. Фрагмент коду зображено на рисунку 3.22.

```
@GetMapping("")
public String main(Map<String, Object> model) {

    model.put("movies", movieRepo.findByFilterMap(new HashMap<>()));
    model.put("genres", genreRepo.findAll());
    model.put("filters", new HashMap<>());

    return "movies/movies-search";
}

@GetMapping("/search")
public String search(@AuthenticationPrincipal User user,
                     @RequestParam(required = false) Map<String, String> form,
                     Map<String, Object> model) {
    List<Movie> movies;

//        movies = movieRepo.findAll();
    movies = movieRepo.findByFilterMap(form);

    model.put("movies", movies);
    model.put("filters", form);
    model.put("genres", genreRepo.findAll());


    return "movies/movies-search";
}
```

Рисунок 3.22 – Фрагмент коду пошуку за фільтрами

3.4 Висновки

У третьому розділі магістерської кваліфікаційної роботи проведено огляд та аналіз технологій і засобів розробки програмних засобів для реалізації методів генерації випадкових чисел, для послідовності номерів. Для реалізації програмного засобу було вибрано мову програмування Java та середовище IntelliJ IDEA. Для розробки серверної частини було вибрано фреймворк Spring, а для клієнтської частини Freemarker та Bootstrap. Реалізовано метод генерації.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАСОБУ

4.1 Методи тестування

Тестування якого програмного забезпечення — загальне поняття, що включає планування, проектування та, власне, виконання тестів.

Тестування сайту – це процес, під час якого визначається, наскільки сайт зручний і привабливий для відвідувача, наскільки швидко і легко на ньому можна відшукати потрібну інформацію, чи відповідає дизайн сайту проекту, якому він присвячений і найважливіше, чи вирішує ті бізнес завдання, для яких він був створений. Тестування сайтів – це трудомісткий процес, який відбувається вже після закінчення робіт з програмування Інтернет-ресурсу в цілому або його модулів. Людина, яка відповідає за якість продукту, тобто тестувальник, оцінює сайт на якість дотримуючись спеціальної методики, а сам процес тестування WEB ділиться на декілька обов'язкових етапів.

Веб-тестування або тестування веб-сайту – це перевірка веб-програми чи веб-сайту на наявність потенційних помилок перед тим, як вони опубліковані та доступні для широкого загалу. Веб-тестування перевіряє функціональність, зручність використання, безпеку, сумісність, продуктивність веб-програми або веб-сайту.

На цьому етапі перевіряються такі питання, як безпека веб-додатків, функціонування сайту, його доступ для неповносправних, а також звичайних користувачів, а також його здатність обробляти трафік.

У програмній інженерії можна виконувати наведені нижче типи/методи тестування залежно від ваших вимог до веб-тестування.

1. Тестування функціональності веб-сайту

Тестування функціональності веб-сайту – це процес, який включає в себе кілька параметрів тестування, таких як користувальницький інтерфейс, API, тестування бази даних, тестування безпеки, клієнтське та серверне тестування та основні функції веб-сайту. Функціональне тестування дуже зручне і дозволяє

користувачам виконувати як ручне, так і автоматичне тестування. Це виконується для перевірки функціональності кожної функції на веб-сайті.

Веб-тестування включає в себе чи всі посилання на ваших веб-сторінках працюють правильно, і переконайтеся, що немає непрацюючих посилань. Посилання, які потрібно перевірити, включатимуть:

– вихідні посилання;

– внутрішні посилання;

– якірні посилання;

– посилання MailTo.

Перевірки скриптів у формі працюють належним чином. Наприклад, якщо користувач не заповнює обов’язкове поле у формі, з’являється повідомлення про помилку. Перевірте, чи заповнюються значення за замовчуванням

Після надсилання дані у формах надсилаються в реальну базу даних або зв’язуються з робочою адресою електронної пошти

Тестові файли cookie працюють належним чином. Файли cookie – це невеликі файли, які використовуються веб-сайтами для запам’ятовування активних сеансів користувача, тому вам не потрібно входити в систему щоразу, коли ви відвідуєте веб-сайт. Тестування файлів cookie включатиме

Тестові файли cookie (сеанси) видаляються або після очищення кешу, або після закінчення терміну їх дії.

Видаліть файли cookie (сесії) і перевірте, чи запитуються облікові дані для входу, коли ви наступного разу відвідуєте сайт.

Тестування робочого процесу включатиме в себе тестування вашого робочого процесу/бізнес-сценарії від кінця до кінця, яке веде користувача через серію веб-сторінок.

Також тестуйте негативні сценарії, щоб, коли користувач виконує несподіваний крок, у вашій веб-програмі відображалося відповідне повідомлення про помилку або довідка.

2. Тестування юзабіліті

Тестування юзабіліті тепер стало важливою частиною будь-якого веб-проекту. Перевірка навігації по сайту: меню, кнопки або посилання на різні сторінки вашого сайту мають бути легко видимими й узгодженими на всіх веб-сторінках

Вміст має бути розбірливим без орфографічних чи граматичних помилок. Зображення, якщо вони присутні, повинні містити «альтернативний» текст.

3. Тестування інтерфейсу

Тут потрібно перевірити три області – додатки, веб і сервер баз даних

Застосування, тестові запити правильно надсилаються в базу даних, а вихідні дані на стороні клієнта відображаються правильно. Помилки, якщо такі є, повинні бути перехоплені програмою та повинні бути показані лише адміністратору, а не кінцевому користувачеві.

Тестовий веб-сервер обробляє всі запити додатків без відмови в обслуговуванні.

Переконайтеся, що запити, надіслані до бази даних, дають очікувані результати.

Перевірте відповідь системи, коли не вдається встановити з’єднання між трьома рівнями (додаток, веб і база даних), і кінцевому користувачеві відображається відповідне повідомлення.

4. Тестування бази даних

База даних є одним із найважливіших компонентів вашого веб-додатка, і необхідно приділити особливу увагу, щоб ретельно перевірити його. Перевірте, чи не відображаються помилки під час виконання запитів. Цілісність даних підтримується під час створення, оновлення або видалення даних у базі даних. Перевірте час відповіді на запити та налаштуйте його, якщо необхідно. Тестові дані, отримані з вашої бази даних, точно відображаються у вашому веб-додатку.

5. Тестування на сумісність.

Тест на сумісність гарантує, що ваш веб-додаток правильно відображається на різних пристроях. Це включатиме - тест на сумісність браузера: один і той самий веб-сайт у різних браузерах відображатиметься по-різному. Вам потрібно перевірити, чи правильно відображається ваш веб-додаток у браузерах, JavaScript, AJAX та аутентифікація працюють нормально. Ви також можете перевірити сумісність мобільного браузера.

Відображення веб-елементів, таких як кнопки, текстові поля тощо, змінюється зі зміною операційної системи. Переконайтеся, що ваш веб-сайт добре працює для різних комбінацій операційних систем, таких як Windows, Linux, Mac і браузерів, таких як Firefox, Internet Explorer, Safari тощо.

6. Тестування продуктивності

Це забезпечить роботу вашого сайту при будь-яких навантаженнях. Діяльність з тестування програмного забезпечення включатиме, але не обмежуючись. Час відповіді веб-сайту на різних швидкостях підключення.

Перевірте навантаження веб-програми, щоб визначити її поведінку при звичайних і пікових навантаженнях. Стрес-тестування вашого веб-сайту, щоб визначити його точку зупинки, коли він перевищив нормальне навантаження в час пік. Перевірте, чи відбувається збій через пікове навантаження, як сайт відновлюється після такої події. Переконайтеся, що такі методи оптимізації, як стиснення zip, кеш браузера та сервера, увімкнено, щоб скоротити час завантаження.

7. Тестування безпеки

Тестування безпеки є життєво важливим для веб-сайтів електронної комерції, які зберігають конфіденційну інформацію про клієнтів. Не слід дозволяти тестовий несанкціонований доступ до захищених сторінок. Обмежені файли не можна завантажувати без відповідного доступу. Сеанси перевірки автоматично припиняються після тривалої бездіяльності користувача. При використанні сертифікатів SSL веб-сайт має перенаправляти на зашифровані сторінки SSL[16].

4.2 Тестування програмного засобу

Тестування додатку було проведено для вибору фільму.

Робота веб сайту починається з запуску його в браузері та відображення інтерфейсу головної сторінки на екрані користувача (рисунок 4.1). Зображено вітання до користувача та описано суть сервісу.

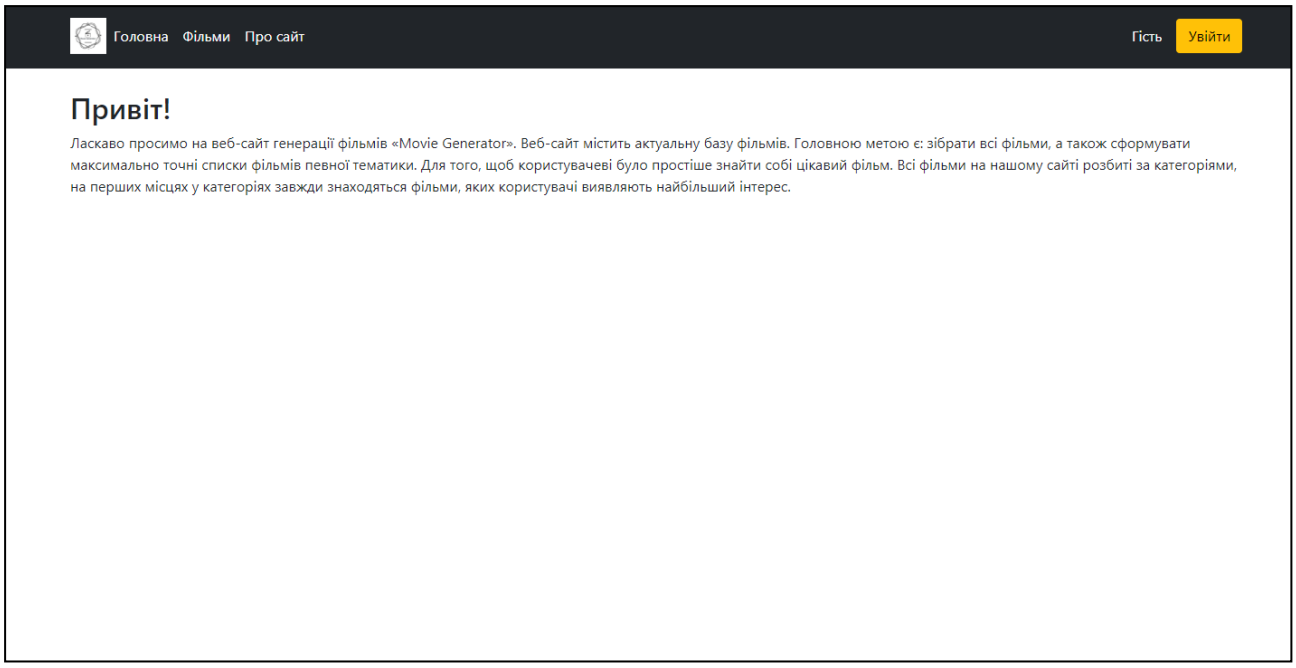

Рисунок 4.1 - Скріншот інтерфейсу головної сторінки

Відкривши «Фільми», отримуємо ось таку сторінку(рис. 4.2), це основна сторінка для пошуку фільмів. Маємо ось такі фільтра: «Назва», «Рік», «Рейтинг».

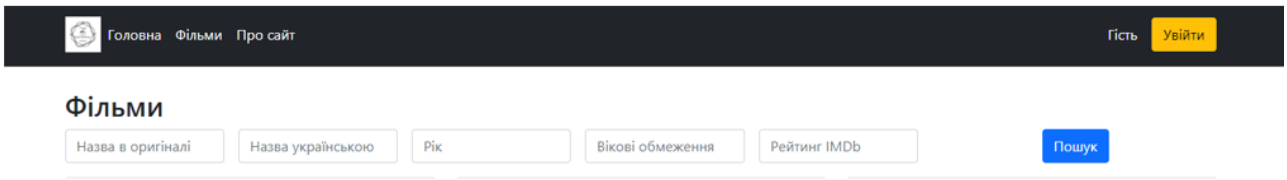

Рисунок 4.2 - Скріншот функціоналу сторінки «Фільми»

Заповнивши поля, отримуємо декілька відповідних фільмів(рис. 4.3), які виводяться на цій же сторінці, з короткою але зрозумілою інформацією.

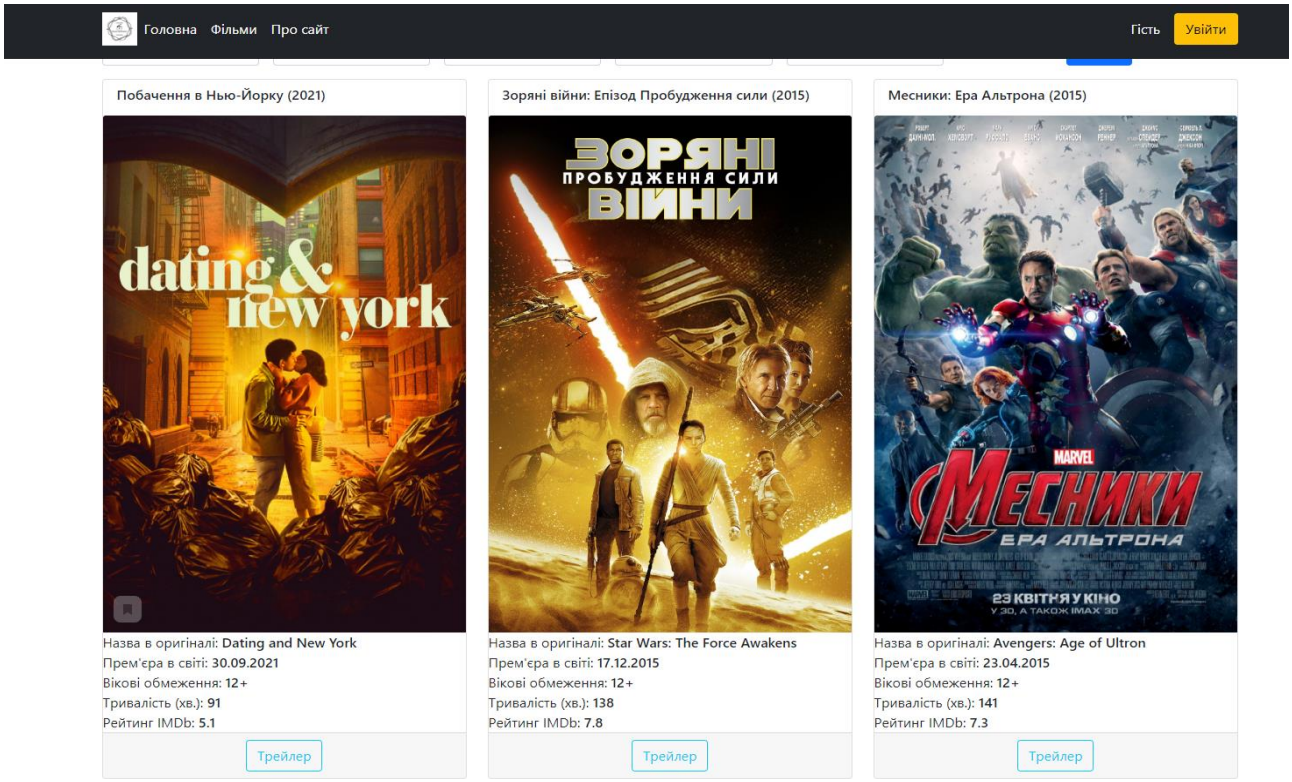

Рисунок 4.3 - Скріншот результату пошуку

Про кожний фільм, є коротка інформація, є афіша, та є можливість подивитись трейлер до фільму(рис 4.4).

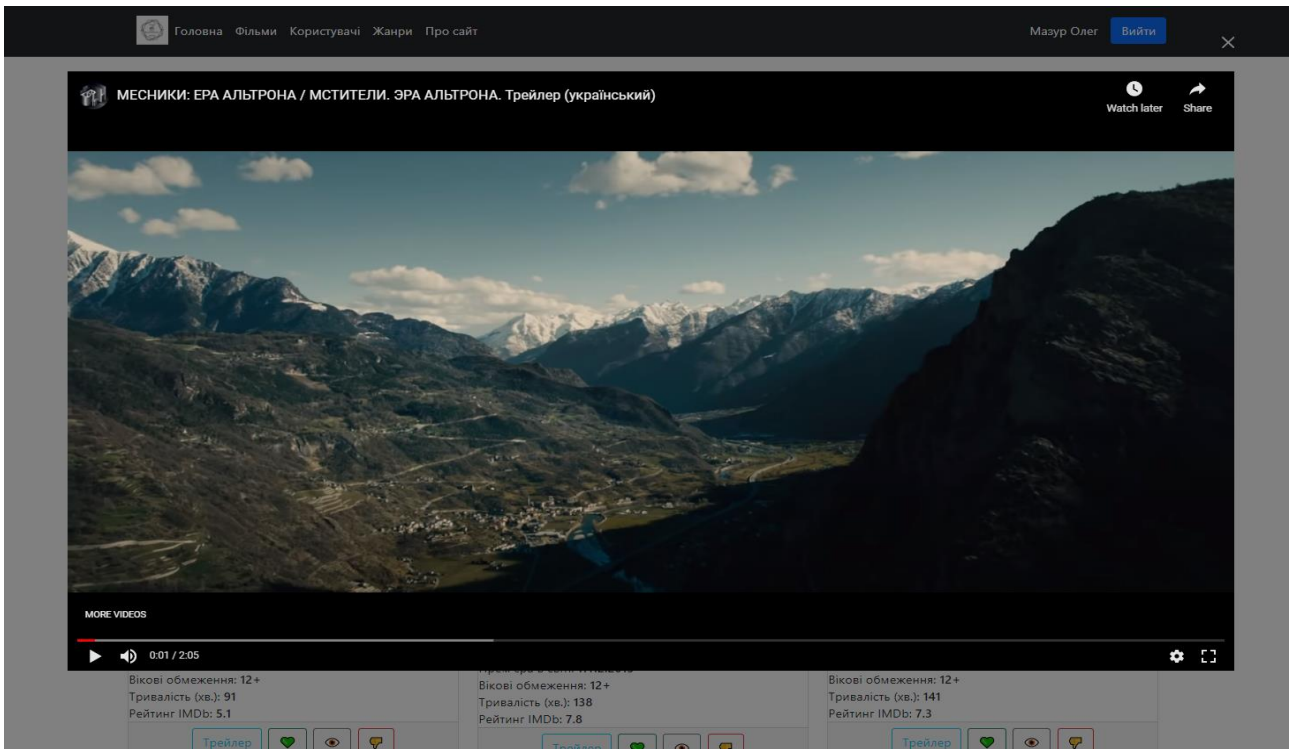

Рисунок 4.4 - Скріншот виводу трейлеру фільму

Також є кнопка «Про сайт», де знаходиться інформація про веб сайт, також правила користування та описана суть реєстрації(рис. 4.5).

Рисунок 4.5 - Скріншот сторінки «Про сайт»

Справа зверху є кнопка «Увійти», тобто є функція реєстрації. Натиснувши її, отримуємо сторінку зображену на рисунку 4.6. Ми бачимо що можна увести логін та пароль, або зареєструватись.

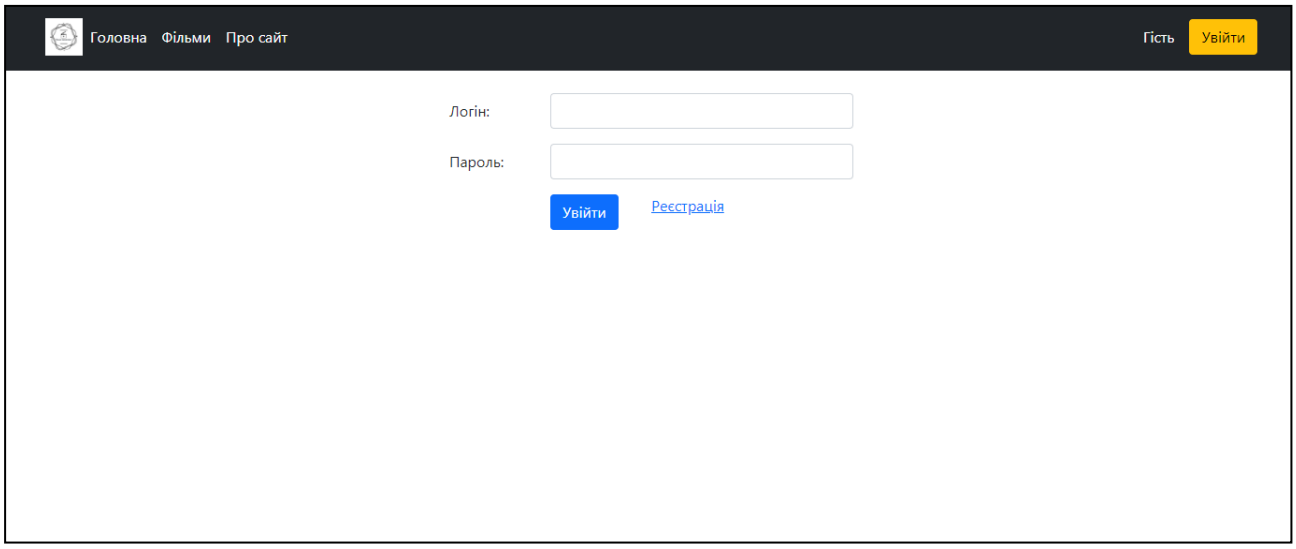

Рисунок 4.6 - Скріншот сторінки входу

На цій сторінці може увійти як користувач – зі своїм логіном та паролем, так і адміністратор – зі своїм логіном та паролем. В користувача з’являється можливість «лайкати» або «дизлайкати» фільми, також відмічати вже переглянуті фільми(рис 4.7).

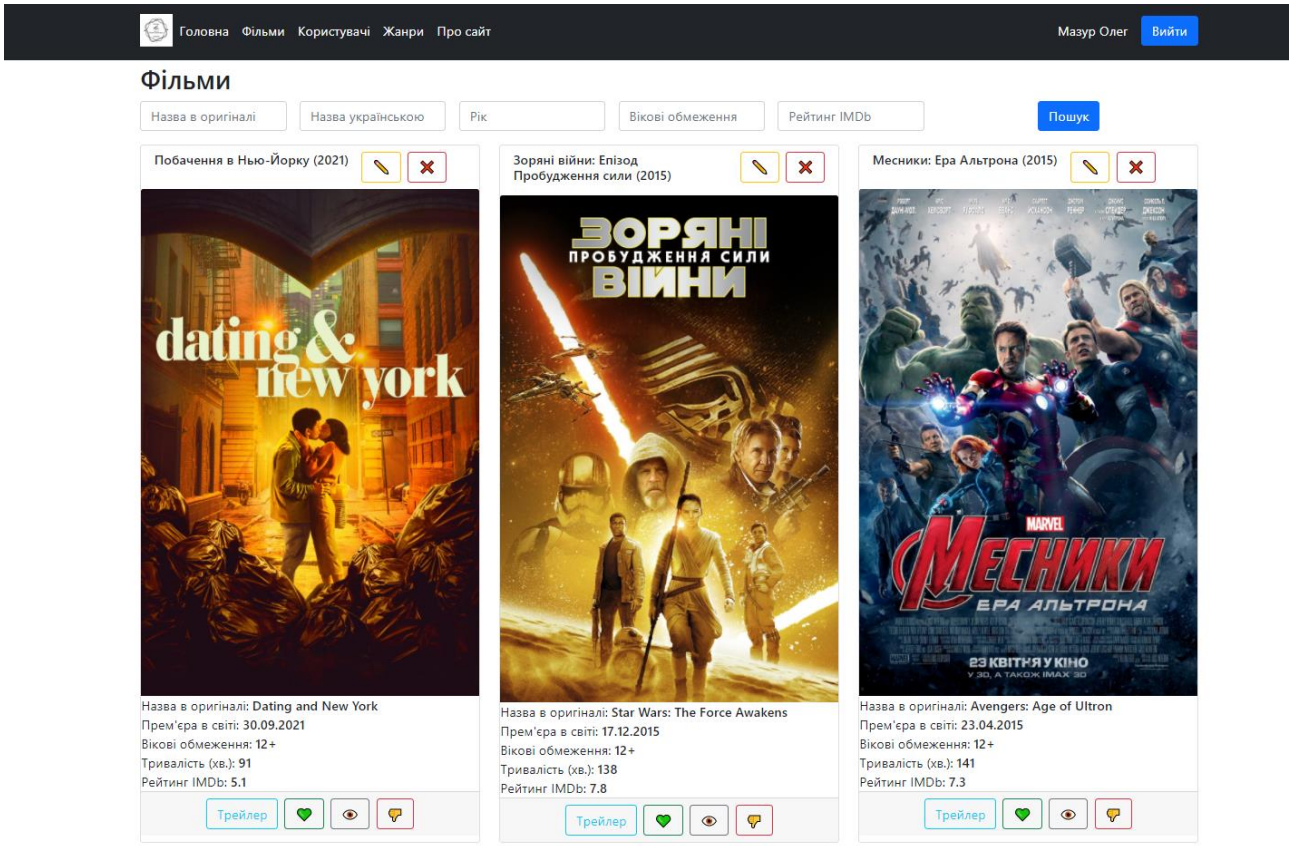

Рисунок 4.7 – Взаємодія користувача з фільмами

В користувача є можливість введення додаткових даних про себе, в такому випадку, в майбутньому зібравши цю інформацію та запустивши нейронну мережу, можна буде більш якісніше підбирати кінофільми для користувачів, відносно їх вподобань та біографії(рис. 4.8).

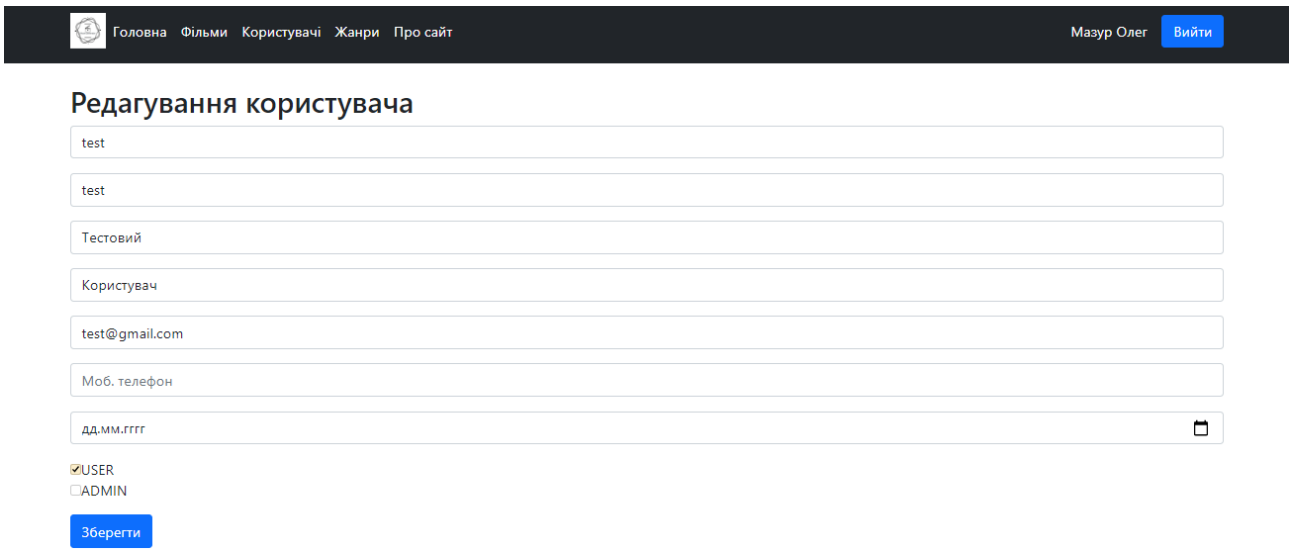

Рисунок 4.8 – Можливість користувача редагування даних

А в адміністратора в свою чергу є можливість перегляду інформації про користувачів, щоб контролювати порядок та ситуацію(рис. 4.9).

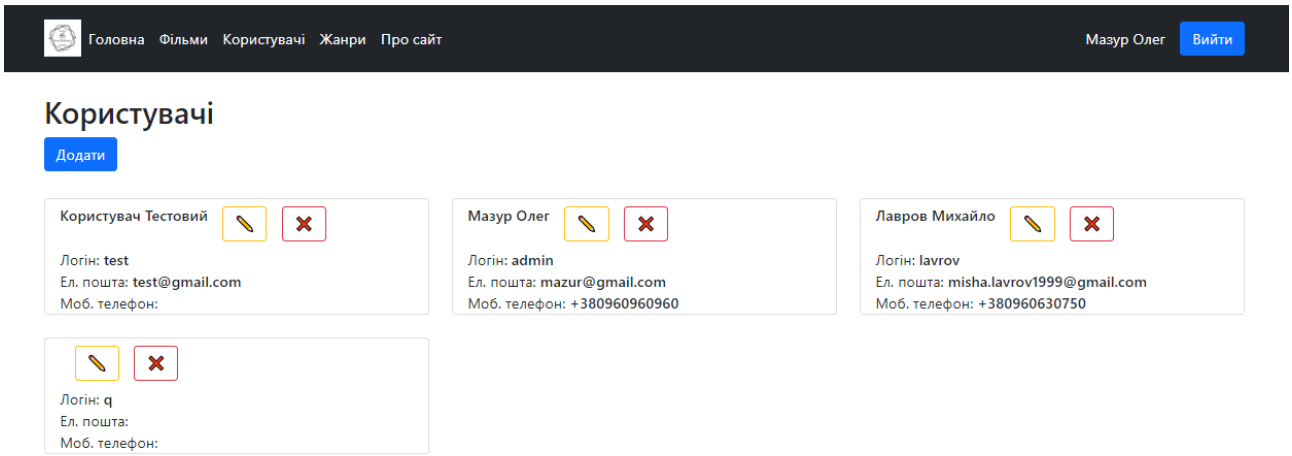

Рисунок 4.9 – Перегляд користувачів адміністратором

Також в адміністратора є можливість моніторингу та редагування інформації про кінофільми. Цю сторінку зображено на рисунку 4.10.

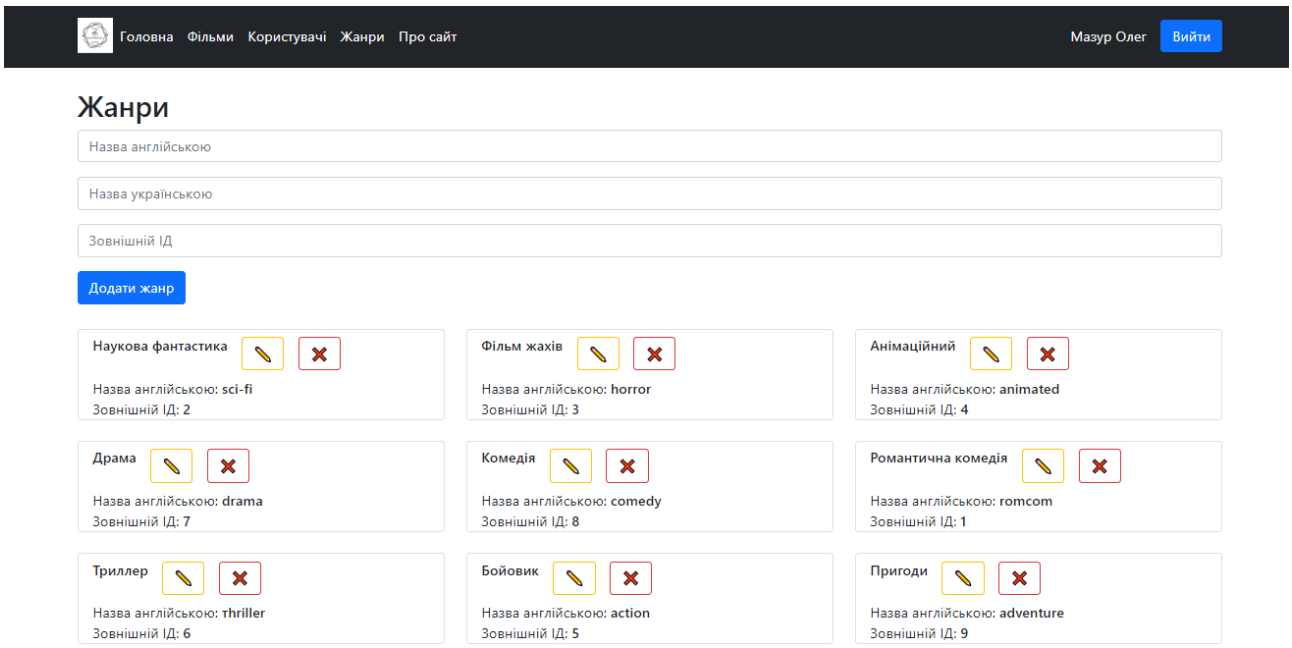

Рисунок 4.10 – Моніторинг та редагування адміністратором кінофільмів

4.3 Висновки

У четвертому розділі магістерської кваліфікаційної роботи було розглянуто та проаналізовано методи тестування. У результаті тестування, було підтверджено правильність роботи методів генерації, та коректність роботи веб сайту для використання. Тестування програми показало повну працездатність і відповідність технічному завданню, що було поставлено.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення комерційного та технологічного аудиту є оцінювання комерційного потенціалу впровадження методів та засобів сервісу генерації відео контенту на основі фільтрів, які дають можливість прискорити процес вибору кінофільмів користувачами, шляхом генерації відеоконтенту за допомогою різноманітних фільтрів.

Для проведення технологічного аудиту було залучено 3-х незалежних експертів: Крилик Л. В., Черноволик Г. О. з Вінницького національного технічного університету та Лаврова М. В. з SoftXpansion, які надали свої оцінки комерційній розробці за допомогою таблиці 5.1 [17].

Таблиця 5.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів

Продовження табл. 5.1

Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки пові-домлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Таблиця 5.2 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

В таблиці 5.3 наведено результати оцінювання експертами комерційного потенціалу розробки.

Таблиця 5.3 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	Крилик Л. В.	Черноволик Г. О.	Лавров М. В.
	Бали, виставлені експертами:		
1	3	3	3
2	2	2	2
3	3	4	3
4	3	4	4
5	3	3	4
6	2	3	2
7	3	3	3
8	3	3	3
9	1	1	1
10	4	4	4
11	4	4	4
12	4	4	4
Сума балів	$СБ_1$=35	$СБ_2$=38	$СБ_3$=37
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{35+38+37}{3} = 37$		

Середньоарифметична сума балів, розрахована на основі висновків експертів склала 37 балів, що згідно таблиці 5.2 вважається, що рівень комерційного потенціалу проведених досліджень є вище середнього.

Методи та засоби сервісу генерації відео контенту на основі фільтрів буде реалізовано в Intellij IDEA як сайт, з базою даних, якою можуть користуватись будь-хто, хто бажає подивитись фільм.

Порівняємо нову розробку з аналогом, який є на ринку, а саме з generator-online.com/uk/movies/.

Основними недоліками аналога є – застарілий дизайн інтерфейсу, також до недоліків можна віднести, що доволі мало фільтрів для відбору фільму, також можуть потрапляти такі ж самі фільми що вже були.

У розробці дана проблема вирішується шляхом збільшується кількості фільтрів, та в кожного користувача є акаунт – за допомогою якого фіксується вся інформація: які фільми вже були переглянуті, які більше подобаються.

Також нова система випереджає аналог за такими параметрами як – швидкодія багатофункціонального генерування, та змогою передивитись трейлер фільму.

В таблиці 5.4 наведені основні техніко-економічні показники аналога і нової розробки.

Таблиця 5.4 - Основні технічні показники аналога і нової розробки

Показники	Аналог	Нова розробка	Відношення параметрів нової розробки до параметрів аналога
Внесення та збереження даних користувача	-	+	1
Безкоштовність (вільний доступ)	+	+	0,5
Фільтра	-	+	1
Сучасний інтерфейс	-	+	1
Українська мова	-	+	1

5.2 Прогнозування витрат на виконання науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи групуються за такими статтями: витрати на оплату праці, витрати на соціальні заходи, матеріали, паливо та енергія для науково-виробничих цілей, витрати на

службові відрядження, програмне забезпечення для наукових робіт, інші витрати, накладні витрати.

1. Основна заробітна плата кожного із дослідників $З_О$, якщо вони працюють в наукових установах бюджетної сфери визначається за формулою:

$$З_О = \frac{М}{Т_р} * t \text{ (грн)} \quad (5.1)$$

де $М$ – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

$Т_Р$ – число робочих днів в місяці; приблизно $Т_Р \approx 21...23$ дні;

t – число робочих днів роботи дослідника.

Для розробки методів та засобів сервісу генерації відео контенту на основі фільтрів необхідно залучити програміста з посадовим окладом 9500 грн. Кількість робочих днів у місяці складає 22, а кількість робочих днів програміста складає 30. Зведемо сумарні розрахунки до таблиця 5.5.

Таблиця 5.5 – Заробітна плата дослідника в науковій установі бюджетної сфери

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату грн.
Керівник	15000	681,8	5	3409
Програміст	9500	431,8	30	12955
Всього				16364

2. Розрахунок додаткової заробітної плати робітників

Додаткова заробітна плата $З_д$ всіх розробників та робітників, які приймали устають в розробці нового технічного рішення розраховується як 10 - 12 % від основної заробітної плати робітників.

На даному підприємстві додаткова заробітна плата начисляється в розмірі 10% від основної заробітної плати.

$$З_д = (З_о + З_р) * \frac{Н_{дод}}{100\%} \quad (5.2)$$

$$З_д = 0{,}11 * 16364 = 1800 \text{ (грн)}$$

3. Нарахування на заробітну плату $Н_{ЗП}$ дослідників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою (5.3):

$$Н_{ЗП} = (З_о + З_д) * \frac{\beta}{100} \text{ (грн)} \quad (5.3)$$

де $З_о$ – основна заробітна плата розробників, грн.;

$З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.;

β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % .

Дана діяльність відноситься до бюджетної сфери, тому ставка єдиного внеску на загальнообов'язкове державне соціальне страхування буде складати 22%, тоді**:**

$$Н_{ЗП} = (16364 + 1800) * \frac{22}{100} = 3996 \text{ (грн)}$$

4. Витрати на матеріали М та комлектуючі К, що були використані під час виконання даного етапу роботи, розраховуються по кожному виду матеріалів за формулою:

$$М = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i - \sum_{1}^{n} В_i \cdot Ц_в \text{ грн.,} \quad (5.4)$$

де $Н_i$ – витрати матеріалу *і*-го найменування, кг;

$Ц_i$ – вартість матеріалу *і*-го найменування, грн./кг.;

$К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15);

$В_i$ – маса відходів матеріалу *і*-го найменування, кг;

$Ц_в$ – ціна відходів матеріалу *і*-го найменування, грн/кг;

n – кількість видів матеріалів.

Таблиця 5.6 – Матеріали, що використані на розробку

Найменування матеріалу	Ціна за одиницю, грн.	Витрачено	Вартість витраченого матеріалу, грн.
Папір	125	1	125
Ручка	13	1	13
CD-диск	15	1	15
Флешка	200	1	200
Всього			353
З врахуванням коефіцієнта транспортування			388,3

5. Програмне забезпечення для наукової роботи включає витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення необхідного для проведення дослідження.

Для розробки використовується безкоштовне програмне забезпечення.

6. Амортизація обладнання, комп'ютерів та приміщень, які використовувались під час виконання даного етапу роботи

Дані відрахування розраховують по кожному виду обладнання, приміщенням тощо.

$$A = \frac{Ц \cdot T}{Ткор \cdot 12} \quad [грн], \tag{5.5}$$

де Ц – балансова вартість даного виду обладнання (приміщень), грн.;

$Т_{кор}$ – час користування;

Т – термін використання обладнання (приміщень), цілі місяці.

Згідно пункта 137.3.3 Податкового кодекса амортизація нараховується на основні засоби вартістю понад 2500 грн. В нашому випадку для написання магістерської роботи використовувався персональний комп'ютер вартістю 21000 грн.

$$\text{A} = \frac{21000 \cdot 1}{2 \cdot 12} = 875$$

7. До статті «Паливо та енергія для науково-виробничих цілей» відносяться витрати на всі види палива й енергії, що безпосередньо використовуються з технологічною метою на проведення досліджень.

$$В_е = \sum_{i=1}^{n} \frac{W_{yt} \cdot t_i \cdot Ц_е \cdot К_{впі}}{\eta_і} \quad (5.6)$$

де W_{yt} – встановлена потужність обладнання на певному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн;

$К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі} < 1$;

$\eta_і$ – коефіцієнт корисної дії обладнання, $\eta_і < 1$.

Для написання магістерської роботи використовується персональний комп'ютер для якого розрахуємо витрати на електроенергію.

$$В_е = \frac{0{,}3 \cdot 230 \cdot 4{,}1 \cdot 0{,}5}{0{,}8} = 176{,}81$$

Витрати на службові відрядження, витрати на роботи, які виконують сторонні підприємства, установи, організації та інші витрати в нашому досліджені не враховуються оскільки їх не було.

Накладні (загальновиробничі) витрати Внзв охоплюють: витрати на управління організацією, оплата службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо. Накладні (загальновиробничі) витрати Внзв можна прийняти як (100…150)% від суми основної заробітної плати розробників та робітників, які виконували дану МКНР, тобто:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.7)$$

де $Н_{нзв}$ – норма нарахування за статтею «Інші витрати».

$$В_{нзв} = 16364 \cdot \frac{100}{100\%} = 16364 \text{ грн}$$

Сума всіх попередніх статей витрат дає витрати, які безпосередньо стосуються даного розділу МКНР

В = 16364+1800+3996+388,3+875+176,81+16364=39963,4

Прогнозування загальних втрат ЗВ на виконання та впровадження результатів виконаної МКНР здійснюється за формулою:

$$ЗВ = \frac{В}{\eta}, \quad (5.8)$$

де η – коефіцієнт, який характеризує стадію виконання даної НДР.

Оскільки, робота знаходиться на стадії науково-дослідних робіт, то коефіцієнт β = 0,7.

Звідси:

$$ЗВ = \frac{39963,4}{0,7} = 57090,55 \text{ грн.}$$

5.3 Розрахунок економічної ефективності науково-технічної розробки

У даному підрозділі кількісно спрогнозуємо, яку вигоду, зиск можна отримати у майбутньому від впровадження результатів виконаної наукової роботи. Розрахуємо збільшення чистого прибутку підприємства $\Delta П_i$, для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, за формулою

$$\Delta П_i = \sum_1^n \left(\Delta Ц_o \cdot N + Ц_o \cdot \Delta N\right)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\upsilon}{100}\right) \quad (5.9)$$

де $\Delta Ц_0$ – покращення основного оціночного показника від впровадження результатів розробки у даному році.

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки:

$Ц_о$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки:

л – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт *л* = 0,8333.

р – коефіцієнт, який враховує рентабельність продукту. р = 0,25;

х – ставка податку на прибуток. У 2021 році – 18%.

Прибуток від реалізації нової розробки ми зможемо отримувати за рахунок реклами. Припустимо, що ціна на рекламу зросте на 100 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року на 1500 шт., протягом другого року – на 4000 шт., протягом третього року на 8000 шт. Реалізація продукції до впровадження розробки складала 1 шт., а її ціна до складає 250 грн. Розрахуємо прибуток, яке отримає підприємство протягом трьох років.

$$\Delta П_1 = [100 \cdot 1 + (250 + 100) \cdot 1500] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 89700{,}99 \text{ грн.}$$

$$\Delta П_2 = [100 \cdot 1 + (250 + 100) \cdot (1500 + 4000)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 328941{,}01 \text{ грн.}$$

$$\Delta П_3 = [100 \cdot 1 + (250 + 100) \cdot (1500 + 4000 + 8000)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 807255{,}21 \text{ грн.}$$

5.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розрахуємо основні показники, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Розрахуємо величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки.

$$PV = k_{\text{інв}} \cdot \text{ЗВ}, \qquad (5.10)$$

$k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо ($k_{\text{інв}}$ = 2…5).

$$PV = 2 \cdot 57090{,}55 \;\; = 114181{,}1$$

Розрахуємо абсолютну ефективність вкладених інвестицій $E_{абс}$ згідно наступної формули:

$$E_{абс} = (ПП - PV) \qquad (5.11)$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство від реалізації результатів наукової розробки, грн.;

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.12)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДЦКР, грн.;

Т – період часу, протягом якою виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,2;

t – період часу (в роках).

$$\text{ПП} = \frac{89700{,}99}{(1+0{,}2)^1} + \frac{328941{,}01}{(1+0{,}2)^2} + \frac{807255{,}21}{(1+0{,}2)^3} = 772516{,}51 \text{ грн.}$$

$$\text{Е}_{\text{абс}} = (772516{,}51 - 114181{,}1) = 658335{,}41 \text{грн.}$$

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього користуються формулою:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.13}$$

$T_ж$ – життєвий цикл наукової розробки, роки.

$$\text{E}_\text{в} = \sqrt[3]{1 + \frac{658335,41}{114181,1}} - 1 = 1,32 = 132\%$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \tag{5.14}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2021 році в Україні d= (0,14…0,2);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1).

$$\tau_{\min} = 0,18 + 0,05 = 0,23.$$

Так як $E_в > \tau_{\min}$ то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в} \tag{5.15}$$

$$\text{Т}_\text{ок} = \frac{1}{1,32} = 0,8 \text{ роки}$$

Так як $T_{ок} \leq$ 3...5-ти років, то фінансування даної наукової розробки в принципі є доцільним.

5.5 Висновки

Було проведено оцінку комерційного потенціалу методів та засобів сервісу генерації відео контенту на основі фільтрів, які дають можливість зменшення часу вибору кінофільмів користувачами, шляхом генерації відеоконтенту за допомогою різноманітних фільтрів. Комерційний потенціал є вище середнього.

Порівняння з аналогом показало, що нова розробка вносить та зберігає дані користувача, має фільтри, на українській мові та має сучасний інтерфейс, чого нема в аналога.

Витрати на виконання науково-дослідної роботи по кожній з статей витрат складе 39963,4 грн. Загальна ж величина витрат на виконання та впровадження результатів даної НДР буде складати 57090,55 грн.

Вкладені інвестиції в даний проект окупляться через 8 місяців при прогнозованому прибутку 772516,51 грн. за три роки.

ВИСНОВКИ

У магістерській кваліфікаційній роботі був розроблений сервіс генерації відеоконтенту на основі фільтрів, з використанням мови програмування Java.

Було проаналізовано сучасний стан проблеми вибору фільму в компаніях, або для «кіноманів». Розглянуто основні аналоги, визначено їх особливості та недоліки і проведено порівняння з власним програмним продуктом. В результаті порівняння було відображено доцільність розробки магістерської кваліфікаційної роботи.

Було розроблено графічний інтерфейс програмного додатку. Розроблено структурні схеми інтерфейсу робочої сторінки та інтерфейсу програмного додатку.

В результаті проведеного варіантного аналізу було обрано мову програмування Java та програмну платформу Spring для серверної, Bootstrap та Freemarker для клієнтської частини та середовище розробки Intellij IDE.

Було вирішено наступні задачі:

- розробити метод генерації відеоконтенту;
- розробити та розвинути способи фільтрації кінофільмів;
- розробити модулі сервісу генерації відеоконтенту;
- провести тестування програмного продукту.

Було проведено оцінку комерційного потенціалу методів та засобів сервісу генерації відео контенту на основі фільтрів, які дають можливість зменшення часу процесу вибору кінофільмів користувачами, шляхом генерації відеоконтенту за допомогою різноманітних фільтрів. Комерційний потенціал є вище середнього.

Тестування програми довело повну працездатність даного програмного продукту та відповідність поставленому технічному завданню.

Магістерську кваліфікаційну роботу було оформлено відповідно до встановлених вимог.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кінематограф в житті сучасної людини [Електронний ресурс] – Режим доступу: https://vbusk.com/cikavo/kinematograf-zhizni-sovremennogo-cheloveka.html
2. Публікація [Електронний ресурс] – Режим доступу: https://www.onaft.edu.ua/itia
3. Публікація [Електронний ресурс] – Режим доступу: http://ir.lib.vntu.edu.ua/bitstream/handle/123456789/34792/%D0%9C%D0%B0%D0%BA%D0%B5%D1%82_%D1%96%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D0%BA%D0%BE%D0%BD%D1%84%D0%B5%D1%80%D0%B5%D0%BD%D1%86%D1%96%D1%8F%202021_%D0%BE%D1%81%D1%82%D0%B0%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%B9%20%281%29.pdf?sequence=1&isAllowed=y
4. Кіно [Електронний ресурс] – Режим доступу:https://om-saratov.ru/blogi/13-January-2016-i32652-osnovnaya-polza-ot-prosmotr
5. Кінотерапія [Електронний ресурс] – Режим доступу:https://blog.rt.ru/b2c/kinoterapiya-kak-smotret-filmy-s-polzoi-dlya-zdorovya.htm
6. Кінематограф [Електронний ресурс] – Режим доступу:https://pravlife.org/ru/content/kinematograf-v-zhizni-sovremennogo-cheloveka
7. Генератор фільмів [Електронний ресурс] – Режим доступу: http://free-generator.ru/films.html
8. Генератор онлайн [Електронний ресурс] – Режим доступу: https://generator-online.com/uk/about/
9. Генератор випадкових чисел [Електронний ресурс] – Режим доступу:https://uk.education-wiki.com/7009496-random-number-generator-in-java
10. Рандомізація [Електронний ресурс] – Режим доступу:https://uk.wikipedia.org/wiki/%D0%A0%D0%B0%D0%BD%D0%B4%D0%BE%D0%BC%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%8F
11. Генератор псевдо випадкових чисел [Електронний ресурс] – Режим доступу:https://conf.ztu.edu.ua/wp-content/uploads/2017/06/123-1.pdf
12. Функціональна блок-схема [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Functional_block_diagram
13. Структурна схема [Електронний ресурс] – Режим доступу: https://studfile.net/preview/972382/page:12/
14. Структурна схема [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%B0%D1%8F_%D1%81%D1%85%D0%B5%D0%BC%D0%B0

15. IDE [Електронний ресурс] – Режим доступу: https://uk.myservername.com/top-10-best-java-ides-online-java-compilers

16. Тестування [Електронний ресурс] – Режим доступу:https://www.guru99.com/web-application-testing.html

17. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

ДОДАТКИ

Додаток А. Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ

д.т.н., проф. О. Н. Романюк

"___" _____ 2021 р.

Технічне завдання

на магістерську кваліфікаційну роботу

«Розробка методів та засобів сервісу генерації відеоконтенту на основі фільтрів»
за спеціальністю

121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

к.т.н., доцент кафедри ПЗ, Черноволик Г.О.

«___» _____ 2021 р.

Виконав:

студент гр. 1ПІ-20м, Мазур О.В.

«___» _____2021 р.

ВНТУ – 2021

Додаток Б. Протокол перевірки на плагіат

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: **Розробка методу та засобів сервісу генерації відеоконтенту на основі фільтрів**.

Тип роботи: кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, 1ПІ – 20м

Науковий керівник: к.т.н. доц. Черноволик Г. О.

Unicheck	
Оригінальність	**88,4%**
Схожість	11,6%

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений з повним звітом подібності, який був згенерований Системою щодо роботи «Розробка методу та засобів сервісу генерації відеоконтенту на основі фільтрів».

Автор _____ Мазур Олег Віталійович

Опис прийнятого рішення: **допустити до захисту**

Особа, відповідальна за перевірку _____ Черноволик Г. О.
(підпис) (прізвище, ініціали)

Експерт _____ _____
(за потреби) (підпис) (прізвище, ініціали, посада)

Додаток В. Лістинг програмного застосунку

MvcVonfig

```
package com.example.movierec.config;

import freemarker.template.utility.XmlEscape;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.MessageSource;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.context.support.ReloadableResourceBundleMessageSource;
import org.springframework.web.servlet.LocaleResolver;
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
import
org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.springframework.web.servlet.i18n.CookieLocaleResolver;
import org.springframework.web.servlet.i18n.LocaleChangeInterceptor;
import org.springframework.web.servlet.view.freemarker.FreeMarkerConfigurer;

import java.util.HashMap;
import java.util.Locale;
import java.util.Map;
import java.util.Properties;

@Configuration
public class MvcConfig implements WebMvcConfigurer {

    @Value("${upload.path}")
    private String uploadPath;

    public void addViewControllers(ViewControllerRegistry registry) {
        registry.addViewController("/login").setViewName("users/login");
    }

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler("/img/**")
                .addResourceLocations("file:///" + uploadPath + "/");
        registry.addResourceHandler("/static/**")
                .addResourceLocations("classpath:/static/");
    }

    @Bean(name = "localeResolver")
    public LocaleResolver getLocaleResolver() {
        CookieLocaleResolver resolver = new CookieLocaleResolver();
        resolver.setCookieDomain("myAppLocaleCookie");
        // 60 minutes
        resolver.setCookieMaxAge(60 * 60);
        resolver.setDefaultLocale(new Locale("uk"));
        return resolver;
    }

    @Bean
    public LocaleChangeInterceptor localeChangeInterceptor() {
        LocaleChangeInterceptor lci = new LocaleChangeInterceptor();
        lci.setParamName("lang");
        return lci;
    }
```

```
    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(localeChangeInterceptor());
    }

    @Bean(name = "messageSource")
    public MessageSource getMessageResource() {
        ReloadableResourceBundleMessageSource messageResource = new
ReloadableResourceBundleMessageSource();
        messageResource.setBasename("classpath:messages/messages");
        messageResource.setDefaultEncoding("UTF-8");
        messageResource.setUseCodeAsDefaultMessage(true);
        return messageResource;
    }

    @Bean(name = "freemarkerConfig")
    public FreeMarkerConfigurer freemarkerConfig() {
        FreeMarkerConfigurer configurer = new FreeMarkerConfigurer();
        configurer.setTemplateLoaderPaths("/WEB-INF/views/",
"classpath:/templates");
        Map<String, Object> map = new HashMap<>();
        map.put("xml_escape", new XmlEscape());
        configurer.setFreemarkerVariables(map);
        Properties settings = new Properties();
        settings.put("auto_import", "spring.ftl as s");
        configurer.setFreemarkerSettings(settings);
        return configurer;
    }

}
```

WebSecurityConfig

```
package com.example.movierec.config;

import com.example.movierec.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders.Authentic
ationManagerBuilder;
import
org.springframework.security.config.annotation.method.configuration.EnableGlobal
MethodSecurity;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecuri
ty;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConf
igurerAdapter;
import org.springframework.security.crypto.password.NoOpPasswordEncoder;

@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private UserService userService;

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
                .authorizeRequests()
                .antMatchers("/", "/registration", "/about", "/movies/**",
```

```
"/static/**", "/img/**").permitAll()
                .anyRequest().authenticated()
                .and()
                .formLogin()
                .loginPage("/login")
                .defaultSuccessUrl("/movies", true)
                .permitAll()
                .and()
                .logout()
                .permitAll();
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception
{
        auth.userDetailsService(userService)
                .passwordEncoder(NoOpPasswordEncoder.getInstance());
    }
```

GenreController

```
package com.example.movierec.controller;

import com.example.movierec.domain.Genre;
import com.example.movierec.domain.User;
import com.example.movierec.repos.GenreRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

import java.io.IOException;
import java.util.Map;

@Controller
@RequestMapping("/genres")
@PreAuthorize("hasAuthority('ADMIN')")
public class GenreController {

    @Autowired
    private GenreRepo genreRepo;

    @GetMapping("")
    public String getGenres(Map<String, Object> model) {
        Iterable<Genre> genres = genreRepo.findAll();
        model.put("genres", genres);
        return "genres/add-genre";
    }

    @PostMapping("/add")
    public String addGenre(@RequestParam String name,
                           @RequestParam String title,
                           @RequestParam(required = false) String externalId,
                           Map<String, Object> model) throws IOException {
        Genre genre = new Genre();
        genre.setName(name);
        genre.setTitle(title);
        genre.setExternalId(externalId);

        genreRepo.save(genre);

        return "redirect:/genres";
    }
```

```
    @GetMapping("/edit/{genre}")
    public String editGenre(@PathVariable Genre genre, Model model) {
        Iterable<Genre> genres = genreRepo.findAll();
        model.addAttribute("genres", genres);

        model.addAttribute("currGenre", genre);

        return "genres/add-genre";
    }

    @PostMapping("/update")
    public String updateGenre(@AuthenticationPrincipal User user,
                              Genre genre) {
        genreRepo.save(genre);

        return "redirect:/genres";
    }

    @GetMapping("/remove/{movieId}")
    public String removeGenre(@PathVariable Long movieId, Model model) {
        genreRepo.deleteById(movieId);
        return "redirect:/genres";
    }
}
```

HomeController

```
package com.example.movierec.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

import java.util.Map;

@Controller
public class HomeController {

    @GetMapping("/")
    public String greetingPage(Map<String, Object> model) {
        return "greeting";
    }

    @GetMapping("/about")
    public String aboutPage(Map<String, Object> model) {
        return "about";
    }
}
```

MovieController

```
package com.example.movierec.controller;

import com.example.movierec.domain.Genre;
import com.example.movierec.domain.Movie;
import com.example.movierec.domain.User;
import com.example.movierec.repos.GenreRepo;
import com.example.movierec.repos.MovieRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.http.MediaType;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.util.StringUtils;
```

```
import org.springframework.web.bind.annotation.*;

import java.io.IOException;
import java.util.Map;
import java.util.Optional;

@Controller
@RequestMapping("/movies")
public class MovieController {

    @Autowired
    private MovieRepo movieRepo;

    @Autowired
    private GenreRepo genreRepo;

    @Value("${upload.path}")
    private String uploadPath;

    @PreAuthorize("hasAuthority('ADMIN')")
    @GetMapping("/add")
    public String addMovie(Model model) {
        Iterable<Genre> genres = genreRepo.findAll();
        model.addAttribute("genres", genres);

        return "movies/add-movie";
    }

    @PreAuthorize("hasAuthority('ADMIN')")
    @PostMapping(
            path = "/add",
            consumes = {MediaType.APPLICATION_FORM_URLENCODED_VALUE}
    )
    public String addMovie(@AuthenticationPrincipal User user,
                           @RequestParam(required = false) Map<String, String>
form,
                           Movie newMovie,
                           Map<String, Object> model) throws IOException {

        newMovie.setCreator(user);

        updateMovie(user, form, newMovie, model);

        return "redirect:/movies/add";
    }

    @PreAuthorize("hasAuthority('ADMIN')")
    @GetMapping("/edit/{movie}")
    public String editMovie(@PathVariable Movie movie, Map<String, Object>
model) {
        Iterable<Genre> genres = genreRepo.findAll();
        model.put("genres", genres);

        model.put("movie", movie);

        return "movies/edit-movie";
    }

    @PreAuthorize("hasAuthority('ADMIN')")
    @PostMapping(
            path = "/update",
            consumes = {MediaType.APPLICATION_FORM_URLENCODED_VALUE}
    )
    public String updateMovie(@AuthenticationPrincipal User user,
```

```
                                @RequestParam(required = false) Map<String,
String> form,
                                Movie movie,
                                Map<String, Object> model) throws IOException {

        String id = form.get("movieId");
        if (!StringUtils.isEmpty(id)) {
            movie.setId(Long.valueOf(id));
        }
        movie.getGenres().clear();

        for (String key : form.keySet()) {
            if (key.startsWith("genre-")) {
                Long genreId = Long.valueOf(form.get(key));
                Optional<Genre> genre = genreRepo.findById(genreId);
                if (genre.isPresent()) {
                    movie.getGenres().add(genre.get());
                }
            }
        }
        movieRepo.save(movie);

        model.put("movie", movie);

        return "redirect:/movies/edit/" + movie.getId();
    }

    @PreAuthorize("hasAuthority('ADMIN')")
    @GetMapping("/remove/{movieId}")
    public String removeMovie(@PathVariable Long movieId, Model model) {
        movieRepo.deleteById(movieId);
        return "redirect:/movies";
    }

}
```

MovieSearchController

```
package com.example.movierec.controller;

import com.example.movierec.domain.Movie;
import com.example.movierec.domain.User;
import com.example.movierec.repos.GenreRepo;
import com.example.movierec.repos.MovieRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

import java.util.HashMap;
import java.util.Map;

@Controller
@RequestMapping("/movies")
public class MovieSearchController {

    @Autowired
    private MovieRepo movieRepo;

    @Autowired
    private GenreRepo genreRepo;

    @GetMapping("")
```

```
    public String main(@RequestParam(required = false, defaultValue = "") String
filter,
                       Map<String, Object> model) {
        Iterable<Movie> movies;

        if (filter != null && !filter.isEmpty()) {
            movies = movieRepo.findByTitle(filter);
        } else {
            movies = movieRepo.findAll();
        }

        Map<String, String> filterText = new HashMap<String, String>() {{
            put("title", "mov");
            put("titleUA", "мов");
        }};
//        movieRepo.findByFilterMap(filterText);

        model.put("movies", movies);
        model.put("filter", filter);

        return "movies/movies-search";
    }

    @GetMapping("/search")
    public String main(@AuthenticationPrincipal User user,
                       @RequestParam(required = false) Map<String, String> form,
                       Map<String, Object> model) {
        Iterable<Movie> movies;
//
//        if(filter != null && !filter.isEmpty()) {
//            movies = movieRepo.findByTitle(filter);
//        } else {
        movies = movieRepo.findAll();
//        }
//
//        Map<String, String> filterText = new HashMap<String, String>(){{
//            put("title", "mov");
//            put("titleUA", "мов");
//        }};
//        movieRepo.findByFilterMap(filterText);

        model.put("movies", movies);
//        model.put("filter", filter);

        return "movies/movies-search";
    }
}
```

RegistrationController

```
package com.example.movierec.controller;

import com.example.movierec.domain.Role;
import com.example.movierec.domain.User;
import com.example.movierec.repos.UserRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.util.StringUtils;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;

import java.util.Collections;
import java.util.Map;

@Controller
```

```
public class RegistrationController {

    @Autowired
    private UserRepo userRepo;

    @GetMapping("/registration")
    public String registration() {
        return "users/registration";
    }

    @PostMapping("/registration")
    public String addUser(User user, Map<String, Object> model) {
        if (StringUtils.isEmpty(user.getUsername()) ||
                StringUtils.isEmpty(user.getPassword())) {
            return "users/registration";
        } else {
            User userFromDb = userRepo.findByUsername(user.getUsername());

            if (userFromDb != null) {
                model.put("message", "msg.registration.user-exists");
                return "users/registration";
            }

            user.setActive(true);
            user.setRoles(Collections.singleton(Role.USER));
            userRepo.save(user);
        }

        return "redirect:users/login";
    }
}
```

UserController

```
package com.example.movierec.controller;

import com.example.movierec.domain.Role;
import com.example.movierec.domain.User;
import com.example.movierec.repos.UserRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.MediaType;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.util.StringUtils;
import org.springframework.web.bind.annotation.*;

import java.util.*;
import java.util.stream.Collectors;

@Controller
@RequestMapping("/users")
@PreAuthorize("hasAuthority('ADMIN')")
public class UserController {
    @Autowired
    private UserRepo userRepo;

    @GetMapping
    public String userList(Model model) {
        List<User> users = userRepo.findAll();
        model.addAttribute("users", users);
        return "users/user-list";
    }
```

```
    @GetMapping("/add")
    public String userEditForm(Model model) {
        model.addAttribute("roles", Role.values());

        return "users/user-edit";
    }

    @GetMapping("/edit/{user}")
    public String userEditForm(@PathVariable User user, Model model) {
        model.addAttribute("currUser", user);
        model.addAttribute("roles", Role.values());

        return "users/user-edit";
    }

    @PostMapping(
            path = "/add",
            consumes = {MediaType.APPLICATION_FORM_URLENCODED_VALUE}
    )
    public String addUser(@AuthenticationPrincipal User u,
                          @RequestParam(required = false) Map<String, String>
form,
                          User user,
                          Map<String, Object> model) {
        User userFromDb = userRepo.findByUsername(user.getUsername());

        if (userFromDb != null) {
            return "redirect:/users";
        }

        user.setActive(true);
        user.setRoles(Collections.singleton(Role.USER));
        userRepo.save(user);
        updateUser(user, form, user, model);
        return "redirect:/users";
    }

    @PostMapping(
            path = "/update",
            consumes = {MediaType.APPLICATION_FORM_URLENCODED_VALUE}
    )
    public String updateUser(@AuthenticationPrincipal User u,
                             @RequestParam(required = false) Map<String, String>
form,
                             User user,
                             Map<String, Object> model) {

        String id = form.get("userId");
        if (!StringUtils.isEmpty(id)) {
            user.setId(Long.valueOf(id));
        }


        Set<String> roles = Arrays.stream(Role.values())
                .map(Role::name)
                .collect(Collectors.toSet());

        user.getRoles().clear();
        for (String key : form.keySet()) {
            if (roles.contains(key)) {
                user.getRoles().add(Role.valueOf(key));
            }
        }
```

```
        user.setActive(true);
        userRepo.save(user);

        return "redirect:/users";
    }

    @GetMapping("/remove/{userId}")
    public String removeUser(@PathVariable Long userId) {
        userRepo.deleteById(userId);
        return "redirect:/users";
    }

}
```

Genre

```
package com.example.movierec.domain;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Genre {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String name;
    private String title;
    private String externalId;
    public Genre() {
    }
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }

    public String getExternalId() {
        return externalId;
    }
    public void setExternalId(String externalId) {
        this.externalId = externalId;
    }
}
```

Movie

```
package com.example.movierec.domain;
```

```
import org.springframework.format.annotation.DateTimeFormat;
import javax.persistence.*;
import java.util.Date;
import java.util.HashSet;
import java.util.Set;

@Entity
public class Movie {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String externalId;
    private String title;
    private String titleUA;
    private Integer durationMin;
    private Integer minAge;
    private Integer budgetDol;
    private Integer cashFeesWorld;
    private Integer cashFeesUSA;
    @DateTimeFormat(pattern = "yyyy-MM-dd")
    private Date premiereDateWorld;
    @DateTimeFormat(pattern = "yyyy-MM-dd")
    private Date premiereDateUA;
    private Double rankKB;
    private Double rankIMDb;
    private String imageUrl;
    private String trailerUrl;
    private String director;

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "user_id")
    private User creator;

    @ManyToMany
    @JoinTable(
            name = "movie_genres",
            joinColumns = {@JoinColumn(name = "genre_id ")},
            inverseJoinColumns = {@JoinColumn(name = "movie_id")}
    )
    private Set<Genre> genres = new HashSet<>();

    public Movie() {
    }
    public Movie(String title, User creator) {
        this.title = title;
        this.creator = creator;
    }
    public String getCreatorName() {
        return this.creator != null ? creator.getUsername() : ">unknown
creator<";
    }
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getExternalId() {
        return externalId;
    }
    public void setExternalId(String externalId) {
        this.externalId = externalId;
    }
```

```
public String getTitle() {
    return title;
}
public void setTitle(String title) {
    this.title = title;
}
public String getTitleUA() {
    return titleUA;
}
public void setTitleUA(String titleUA) {
    this.titleUA = titleUA;
}
public Integer getDurationMin() {
    return durationMin;
}
public void setDurationMin(Integer durationMin) {
    this.durationMin = durationMin;
}
public Integer getMinAge() {
    return minAge;
}
public void setMinAge(Integer minAge) {
    this.minAge = minAge;
}
public Integer getBudgetDol() {
    return budgetDol;
}
public void setBudgetDol(Integer budgetDol) {
    this.budgetDol = budgetDol;
}
public Integer getCashFeesWorld() {
    return cashFeesWorld;
}
public void setCashFeesWorld(Integer cashFeesWorld) {
    this.cashFeesWorld = cashFeesWorld;
}
public Integer getCashFeesUSA() {
    return cashFeesUSA;
}
public void setCashFeesUSA(Integer cashFeesUSA) {
    this.cashFeesUSA = cashFeesUSA;
}
public Date getPremiereDateWorld() {
    return premiereDateWorld;
}
public void setPremiereDateWorld(Date premiereDateWorld) {
    this.premiereDateWorld = premiereDateWorld;
}
public Date getPremiereDateUA() {
    return premiereDateUA;
}
public void setPremiereDateUA(Date premiereDateUA) {
    this.premiereDateUA = premiereDateUA;
}
public Double getRankKB() {
    return rankKB;
}
public void setRankKB(Double rankKB) {
    this.rankKB = rankKB;
}
public Double getRankIMDb() {
    return rankIMDb;
}
public void setRankIMDb(Double rankIMDb) {
```

```
        this.rankIMDb = rankIMDb;
    }
    public String getImageUrl() {
        return imageUrl;
    }

    public void setImageUrl(String imageUrl) {
        this.imageUrl = imageUrl;
    }
    public String getTrailerUrl() {
        return trailerUrl;
    }
    public void setTrailerUrl(String trailerUrl) {
        this.trailerUrl = trailerUrl;
    }
    public User getCreator() {
        return creator;
    }
    public void setCreator(User creator) {
        this.creator = creator;
    }
    public String getDirector() {
        return director;
    }
    public void setDirector(String director) {
        this.director = director;
    }
    public Set<Genre> getGenres() {
        return genres;
    }
    public void setGenres(Set<Genre> genres) {
        this.genres = genres;
    }
}
```

Role

```
package com.example.movierec.domain;

import org.springframework.security.core.GrantedAuthority;

public enum Role implements GrantedAuthority {
    USER,
    ADMIN;

    @Override
    public String getAuthority() {
        return name();
    }
}
```

User

```
package com.example.movierec.domain;

import org.springframework.format.annotation.DateTimeFormat;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import javax.persistence.*;
import java.util.Collection;
import java.util.Date;
import java.util.HashSet;
import java.util.Set;

@Entity
```

```
@Table(name = "usr")
public class User implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String username;
    private String password;
    private boolean active;
    private String nickname;
    private String email;
    private String firstName;
    private String lastName;
    private String phone;
    @DateTimeFormat(pattern = "yyyy-MM-dd")
    private Date birthDate;
    @ElementCollection(targetClass = Role.class, fetch = FetchType.EAGER)
    @CollectionTable(name = "user_role", joinColumns = @JoinColumn(name =
"user_id"))
    @Enumerated(EnumType.STRING)
    private Set<Role> roles = new HashSet<>();
    public boolean isAdmin() {
        return getRoles().contains(Role.ADMIN);
    }
    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return getRoles();
    }
    @Override
    public boolean isAccountNonExpired() {
        return true;
    }
    @Override
    public boolean isAccountNonLocked() {
        return true;
    }
    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }
    @Override
    public boolean isEnabled() {
        return isActive();
    }
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public boolean isActive() {
        return active;
    }
```

```
    public void setActive(boolean active) {
        this.active = active;
    }
    public Set<Role> getRoles() {
        return roles;
    }
    public void setRoles(Set<Role> roles) {
        this.roles = roles;
    }
    public String getNickname() {
        return nickname;
    }
    public void setNickname(String nickname) {
        this.nickname = nickname;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    public Date getBirthDate() {
        return birthDate;
    }
    public void setBirthDate(Date birthDate) {
        this.birthDate = birthDate;
    }
    public String getPhone() {
        return phone;
    }
    public void setPhone(String phone) {
        this.phone = phone;
    }
}
```

Додаток Г. Ілюстративна частина

РОЗРОБКА МЕТОДУ ТА ЗАСОБІВ СЕРВІСУ ГЕНЕРАЦІЇ ВІДЕОКОНТЕНТУ НА ОСНОВІ ФІЛЬТРІВ

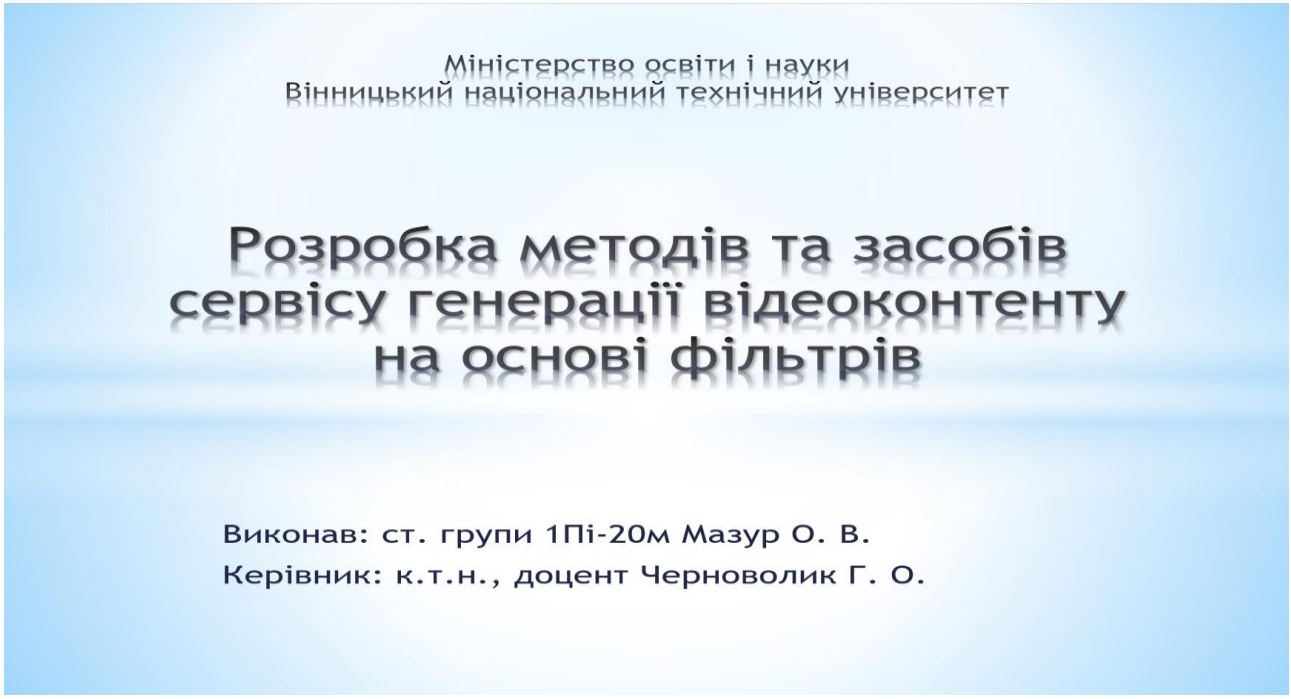

Рисунок Г.1 – Плакат 1. Титульний лист презентації

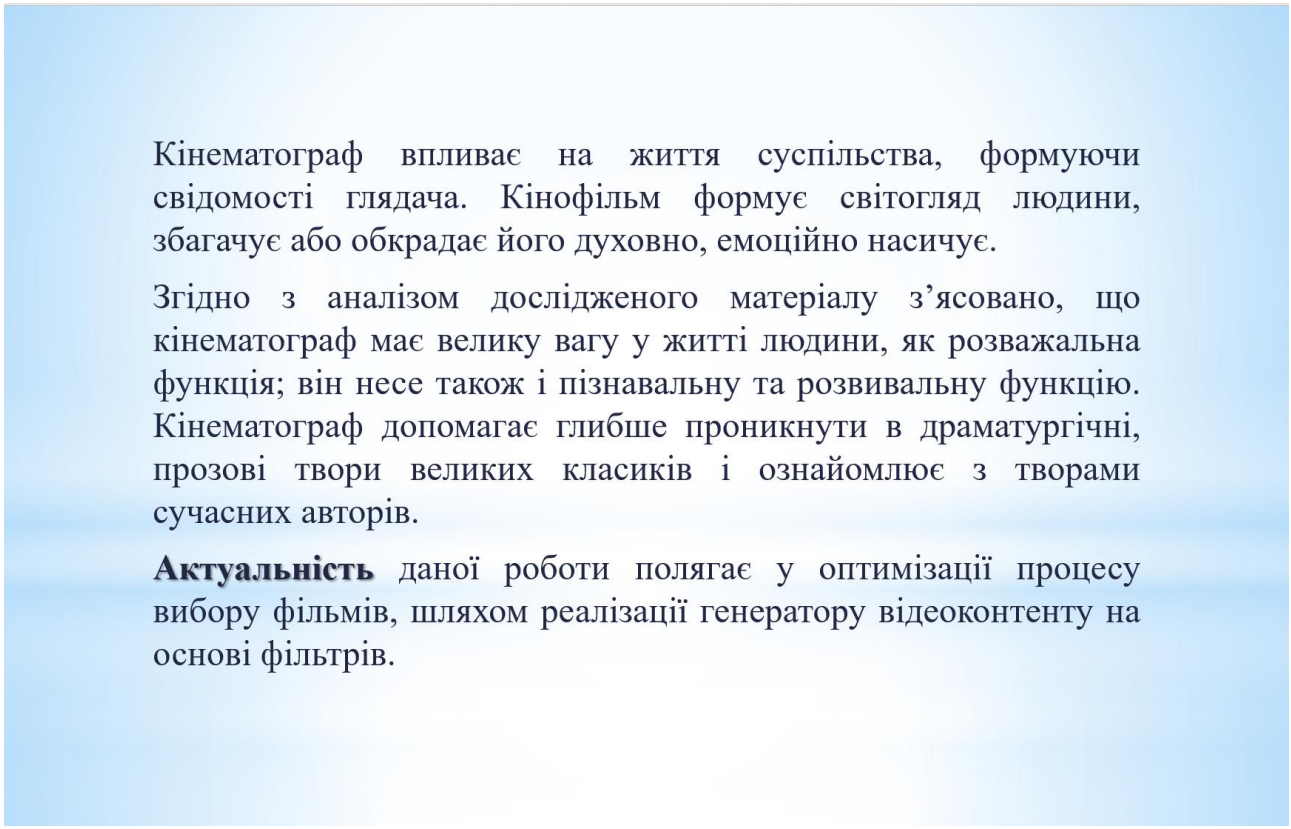

Рисунок Г.2 – Плакат 2. Актуальність роботи

Мета та завдання дослідження. Основною метою роботи є зменшення часу вибору кінофільмів користувачами, шляхом генерації відеоконтенту за допомогою фільтрів.

Відповідно до поставленої мети потрібно виконати такі **задачі**:

- Розробити метод генерації відеоконтенту;
- Розробити та реалізувати способи фільтрації кінофільмів;
- Розробити програмні модулі сервісу генерації відеоконтенту;
- Провести тестування програмного продукту.

Рисунок Г.3 – Плакат 3. Мета та завдання дослідження

Об'єкт дослідження – процес реалізації генерації та візуалізації відеоконтенту із допомогою програмного засобу.

Предмет дослідження – методи та засоби сервісу генерації відеконтенту на основі фільтрів.

Рисунок Г.4 – Плакат 4. Об'єкт та предмет дослідження

Методи дослідження:

- методи проектування програмного сервісу для фільтрації відеоконтенту
- методи теорії алгоритмів для розробки алгоритмів і розробки сервісу
- методи генерації випадкових чисел, для генерації послідовності номерів

Наукова новизна отриманих результатів полягає в наступному:

Подальшого розвитку набув метод генерації відеоконтенту, який відрізняється від існуючого, використанням розширених фільтрів, що дозволяє розширити функціонал пошуку для користувача.

Рисунок Г.5 – Плакат 5. Наукова новизна

Практична цінність одержаних результатів полягає у тому, що на основі проведених теоретичних досліджень і отриманих наукових результатів розроблено програмний засіб для актуального та багатофункціонального сервісу генерації відеоконтенту.

Апробація матеріалів магістерської кваліфікаційної роботи. Основні положення й результати досліджень було подано на:

XIV Міжнародна науково-практична конференція «Інформаційні технології і автоматизація - 2021» - 15 жовтня 2021 р.

Електронні інформаційні ресурси: створення, використання, доступ. Пам'яті Олексія Петровича Стахова. Збірник матеріалів Міжнародної науково-практичної Інтернет конференції 9-10 листопада 2021 р. – Суми/Вінниця: НІКО/ВНТУ, 2021. – 223 с.

Рисунок Г.6 – Плакат 6. Практична цінність

Рисунок Г.6 – Плакат 7. Схема методу генерації відеоконтенту

Рисунок Г.8 – Плакат 8. Структурна схема функціонування веб застосунку

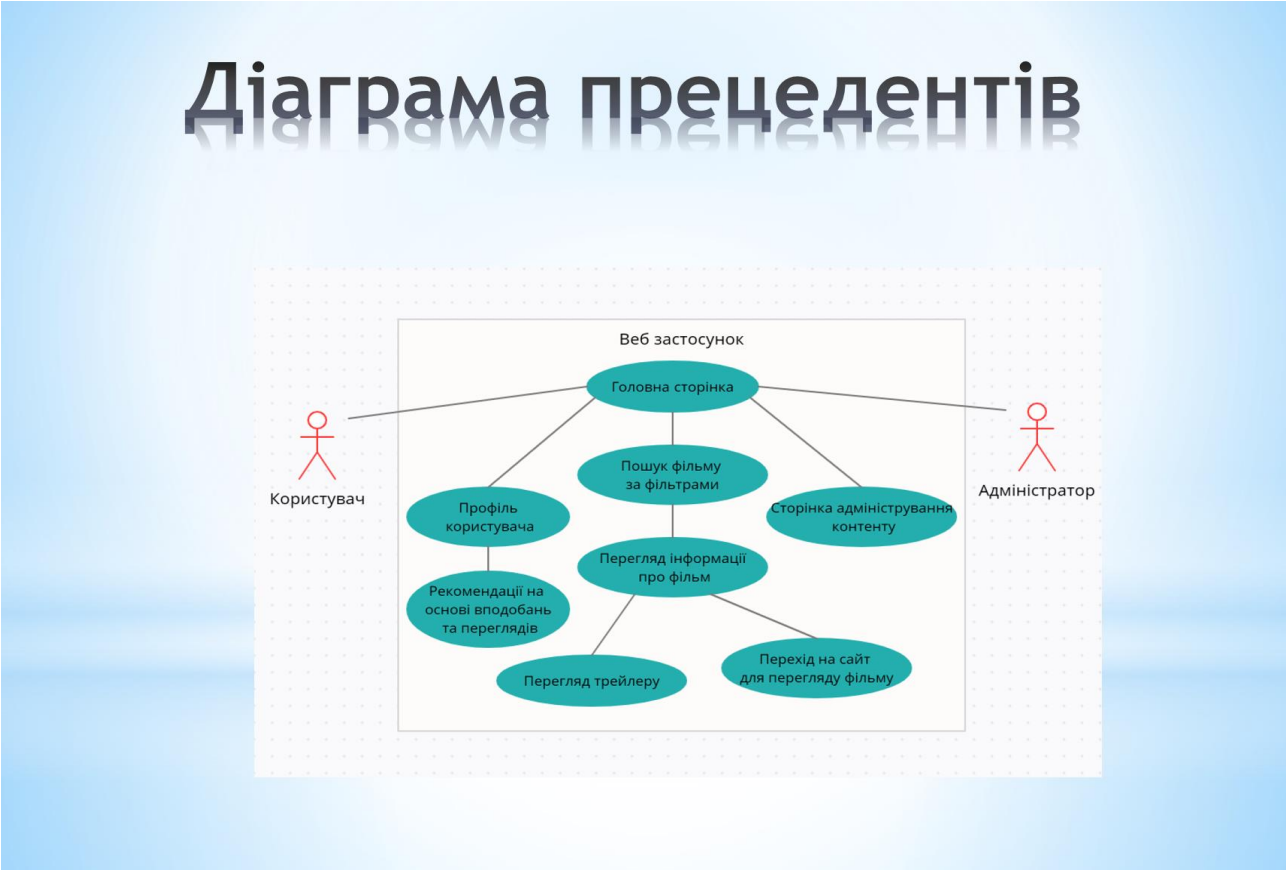

Рисунок Г.9 – Плакат 9. Діаграма прецедентів

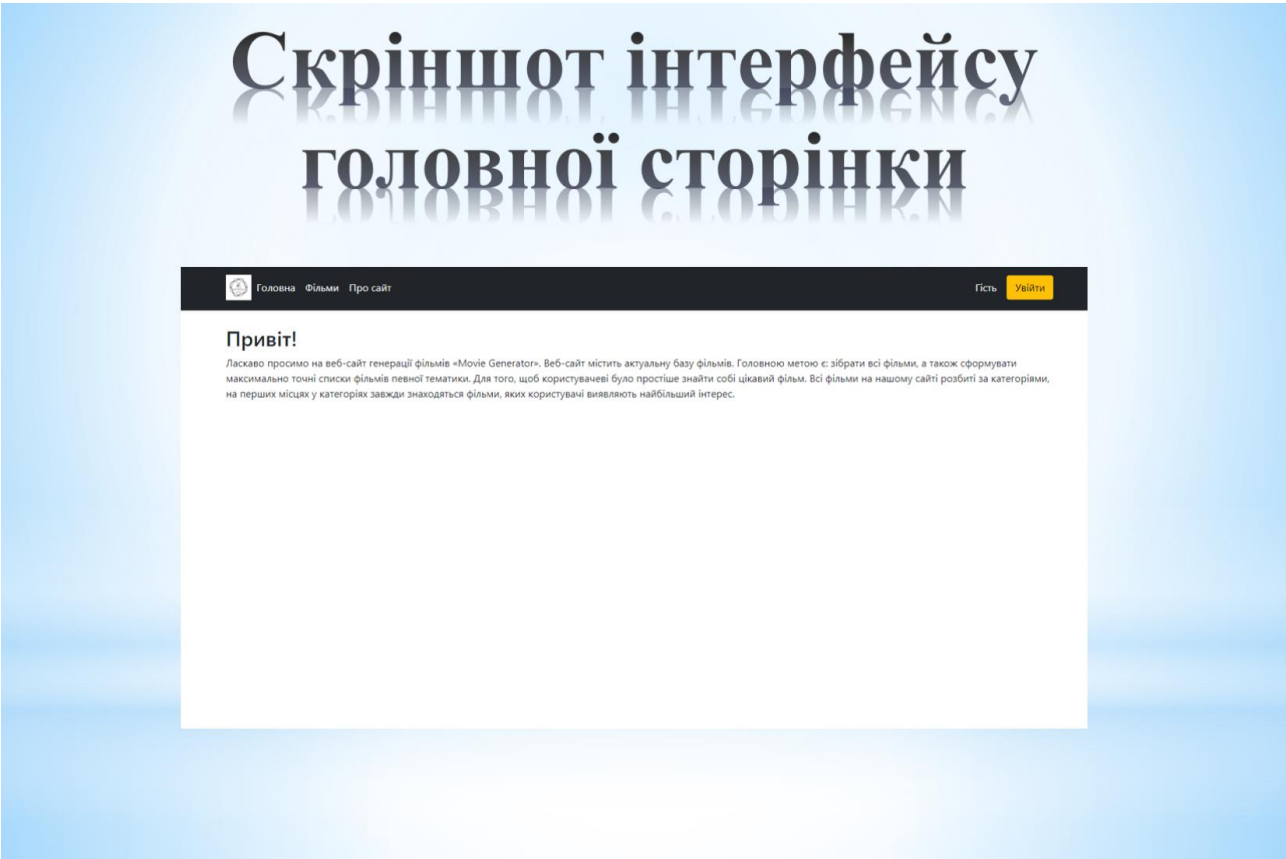

Рисунок Г.10 – Плакат 10. Інтерфейс головної сторінки

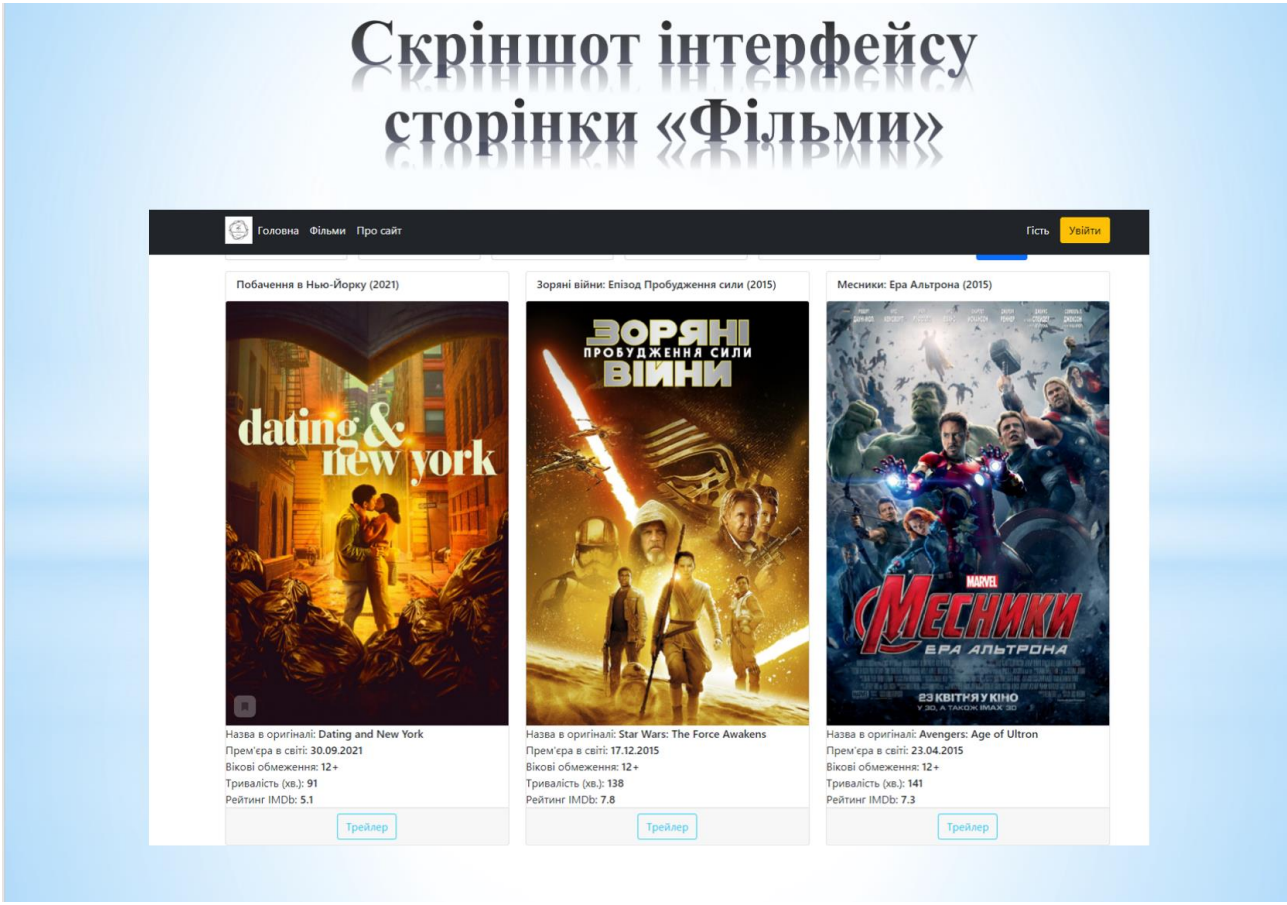

Рисунок Г.11 – Плакат 11. Інтерфейс сторінки «Фільми»

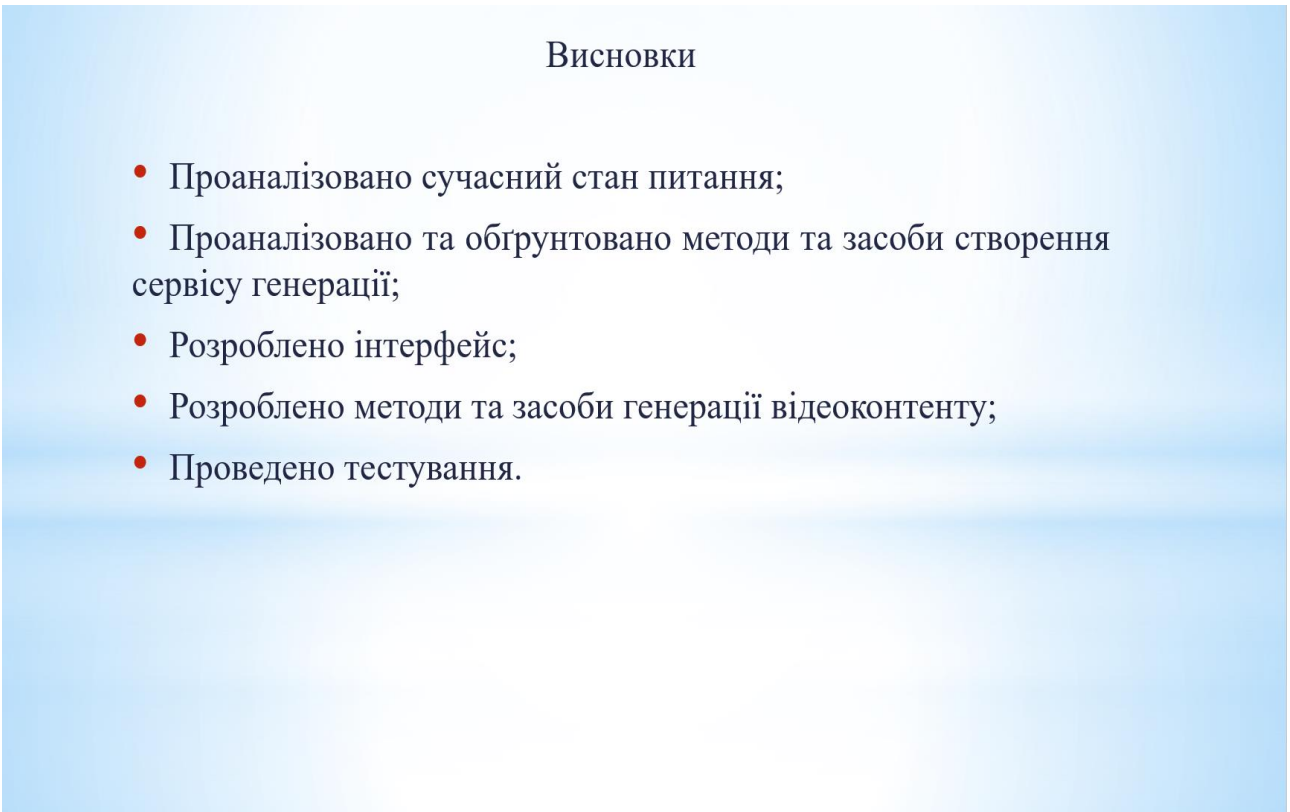

Рисунок Г.12 – Плакат 12. Висновки

Рисунок Г.13 – Плакат 13. Кінець доповіді