ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Освітньо-кваліфікаційний рівень магістр
Спеціальність 121 "Інженерія програмного забезпечення"

ЗАТВЕРДЖУЮ
Завідувач кафедри КН,
_____А.В. Пукас
"____" _____20___року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Петрунів Олександрі Михайлівні

1. Тема роботи: Метод та програмне забезпечення для генерування відеоконтенту в реальному масштабі часу / Method and software for generating video content on real-time

Керівник роботи: Шевчук Р.П.

2. Строк подання студентом роботи

3. Вихідні дані до роботи: мова програмування – Java та програмна платформа Spring для серверної, Bootstrap та Freemarker для клієнтської частини та середовище розробки Intellij IDE.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз сучасного стану питання та постановка задач дослідження, 2. Розробка методу для генерації відеоконтенту, 3. Розробка сервісу генерації відеоконтенту на основі фільтрів, 4. Тестування програмного засобу, 5. Економічна частина

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

Структурна схема функціонування веб-застосунку MovieGenerator, діаграма класів веб-застосунку MovieGenerator, блок-схема алгоритму генерації псевдовипадкових чисел, блок-схема авторизації користувача в системі, діаграма прецедентів веб-застосунку Movie Generator

РЕЗЮМЕ

Кваліфікаційна робота містить 68 сторінок, 39 рисунків, 3 таблиці, 3 додатоки та 17 джерел в переліку посилань.

Мета і задачі дослідження. Основна мета роботи полягає в розробці програмного забезпечення для генерування відеоконтенту в реальному маштабі часу, яке дозволить скоротити час процесу відбору фільмів користувачами, шляхом генерації відеоконтенту за допомогою різних фільтрів.

Об'єктом дослідження є процес генерування та візуалізації відеоконтенту в реальному маштабі часу.

Предмет дослідження: методи та алгоритми генерування та візуалізації відеоконтенту в реальному маштабі часу на основі фільтрів.

Наукова новизна отриманих результатів. Удосконалено метод генерації відеоконтенту, який на відміну від існуючих, використовує розширені фільтри для генерування відеоконтенту, що дозволяє розширити функціональність процедури пошуку відео для користувача.

Ключові слова: відеоконтент, генерація, кінематографія, програмне забезпечення, рандомізація, візуалізація, кваліфікаційна робота.

RESUME

The qualification work contains 68 pages, 39 figures, 3 tables, 3 appendices and 17 sources in the list of references.

The purpose and objectives of the research. The main goal of the work is to develop software for the generation of video content in real time, which will reduce the time of the process of selecting movies by users, by generating video content using various filters.

The object of research is the process of generating and visualizing video content in real time.

Research subject: methods and algorithms for generating and visualizing video content in real time based on filters.

Scientific novelty of the obtained results. The video content generation method has been improved, which, unlike the existing ones, uses advanced filters to generate video content, which allows to expand the functionality of the video search procedure for the user.

Keywords: video content, generation, cinematography, software, randomization, visualization, qualification work.

ЗМІСТ

ВСТУП

Актуальність теми. Нам відомо, що серед всіх видів мистецтва кіноіндустрія посідає важливе місце в сучасному світі, а отже, і в житті людини. Кінематографія впливає на розвиток суспільства, ти самим формуючи свідомість глядача. Фільм формує світогляд людини, збагачує або обкрадає її духовно, насичує емоціями. Звідси випливає ще одна родзинка кіно – психологічний вплив на людину.

Ми живемо в епоху пікового вмісту. Розважальний продукт створюється настільки багато і такими темпами, що проблема «що дивитися» начебто вже не існує. За результатами аналізу досліджуваного матеріалу встановлено, що кінематограф має велике значення в житті людини як розважальна функція; він також виконує пізнавальну та розвиваючу функцію. Кінематограф допомагає глибше проникнути в драматургію і прозову творчість великих класиків, знайомить з творами сучасних авторів.

За результатами аналізу досліджуваного матеріалу встановлено, що кінематограф має велике значення в житті людини як розважальна функція; він також виконує пізнавальну та розвиваючу функцію. Кінематограф допомагає глибше проникнути в драматургію і прозову творчість великих класиків, знайомить з творами сучасних авторів[1].

Актуальність даної роботи полягає в оптимізації процесу відбору фільмів шляхом реалізації генератора відеоконтенту на основі фільтрів.

Мета і задачі дослідження. Основна мета роботи полягає в розробці програмного забезпечення для генерування відеоконтенту в реальному маштабі часу, яке дозволить скоротити час процесу відбору фільмів користувачами, шляхом генерації відеоконтенту за допомогою різних фільтрів.

Відповідно до поставленої мети необхідно виконати наступні завдання:

- проаналізувати предметну область;
- розробити методику генерації відеоконтенту;

- розробити та впровадити метод для генерегуван відеоконтенту на базі плівкової фільтрації;

- розробити програмне забезпечення для генерування відеоконтенту в реальному масштабі часу;

- провести тестування програмне забезпечення для генерування відеоконтенту в реальному масштабі часу.

Об'єктом дослідження є процес генерування та візуалізації відеоконтенту в реальному маштабі часу.

Предмет дослідження: методи та алгоритми генерування та візуалізації відеоконтенту в реальному маштабі часу на основі фільтрів.

Методи дослідження: Для вирішення поставлених у кваліфікаційній роботі завдань використано: принципи об'єктно-орієнтованого програмування; методи теорії алгоритмів; методи теорія інформації.

Наукова новизна отриманих результатів.

Удосконалено метод генерації відеоконтенту, який на відміну від існуючих, використовує розширені фільтри для генерування відеоконтенту, що дозволяє розширити функціональність процедури пошуку відео для користувача.

Практична цінність отриманих результатів полягає в тому, що відповідно до проведених теоретичних досліджень та отриманих наукових результатів розроблено програмне забезпечення для генерування відеоконтенту в реальному масштабі часу.

Особистий внесок магістранта

Всі результати отримані автором самостійно.

Апробація результатів. Основні положення та результати кваліфікаційної роботи обговорювались на семінарі «Комп'ютерні інформаційні технології», що проходив 29 листопада 2022 року в м. Тернопіль.

Публікації.

Петрунів О.М. Програмне забезпечення для генерування відеоконтекту в реальному маштабі часу // Матеріали школі-семінару молодих вчених і студентів «Комп'ютерні інформаційні технології» . — Тернопіль : ФО-П Шпак В.Б., 2022.

РОЗДІЛ І
АНАЛІЗ СУЧАСНОГО СТАНУ ПИТАННЯ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

Кіно являється унікальним досягненням людино. За допомогою кіно кожен з нас може на деякий час перенестися в іншу паралель таі пережити велику кількість яскравих найрізноманітніших емоцій, оскільки на сьогодні існує безліч кіножанрів, які можуть заставити людину сміятися або плакати, померти від страху або поєднати всі ці відчуття це в єдиному пориві.

Напевно важко знайти когось, який би б не отримав задоволення від перегляду кінострічки. Коли емоції на найвищому рівні і відповідний настрій, дуже важливо знайти якесь «натхнення» для того , щоб підвищити почуття, які ми відчуваємо на даний час, і це можливо втілити за допомогою сучасного кіно.

При правильному перегляді зображення можуть бути величезні переваги. Багато вчених допускають, що так можна впоратися з багатьма найскладнішими емоційними проблемами.

Кіно навчає нас багато чому: як правильно вести себе в тій чи іншій ситуації, як реагувати на ріноманітні події в житті, сприймати і проаналізувати те, що відбувається. Є багато фільмів, які після перегляду стають культовими, сильно впливають на наше життя, змінюючи його на краще.

Фільми дозволять значно допомогти створити потрібний настрій і стати основою для переосмислення всього, що відбувається. У цьому випадку ми як би дивимося на нього з боку, в результаті відбувається переоцінка цінностей та ідеалів.

За допомогою фільмів можна формувати характер і лінію поведінки дитини, її свідомість. Груповий чи сімейний перегляд фільмів надзвичайно зближує, тому таке проведення часу потрібно практикувати якомога частіше.

Кіно – це цілий світ. Світ прекрасний і жахливий, смішний і сумний, добрий і злий, кольоровий. це світ історії і сучасності, реальності і казки, любові і ненависті. Кіно люблять всі, від дитини до дорослого, оскільки перегляд

фільму – це захоплюючий і пізнавальний процес, кіно приносить задоволення, прикрашає дозвілля, піднімає настрій або, навпаки, наводить на сумні думки, воно здатне викликати у глядачів різні відчуття - сміх чи сльози, смуток чи радість[4].

1.1. Аналіз стану предметної області

З самого початку розвитку кіно вплив цього виду мистецтва на глядача стало цікавити вчених та психологів. Проведені різноманітні дослідження та накопичена поступова інформація дали змогу розробити методи лікування, які базуються на ефекті, який можна отримати від перегляду фільму. Завдяки тому, що у нас є можливість здійснювати аналіз поведінки кіногероїв на екрані, а також переносити проекцію побаченої ситуації на власне життя, кінотерапія стала важливим інструментом, який не тільки здійснює терапію, але й мотивує на розвиток різних здібностей і підвищення навичок професійності.

Ефективність фільмів, як психологічних інструментів, включає наступні фактори. Інтенсивність: історія, сценарій та герої зібрані разом та «упаковані» в обмежений по часу формат. Тривалість: фільм за своєю тривалістю порівнюється з сеансом психотерапії. Навчання: фільми здатні доносити інформацію до глядача у вигляді певної алегорії, подібно до казок і притч . Ідентифікація: багато хто з нас ототожнює себе з деяким героєм фільму. Увага: завдяки тому, що ми бачимо фільми візуально, це сприяє зосередженню уваги на різних образах. Соціальна користь: перегляд та подальше обговорення фільму разом із друзями чи рідними збільшують його цінність, сприяють розвитку відносин та підвищують ефективність терапії.

Кінотерапія надає нам багато переваг, які ми реально відчуваємо. Можна назвати декілька прикладів серед них. Відновлення, коли перегляд кінострічки, дозволяє відновити дію завдяки тому, що ми можемо відволіктися, розслабитися і гарно провести свій час. Робота зі страхами – дивлячись окремі сцени або цілі фільм, кінотерапія допомагає подивитися своїм страхам в

обличчя, щоб навчитися перемагати їх. Усвідомлення проблем – співпереживаючи героям фільму, багато людей усвідомлюють власні проблеми, наявність яких раніше не була очевидною. Уміння переживати ситуацію - фільми є своєрідним симулятором реальності, що дозволяє пережити ситуацію, не маючи при цьму ніяких реальних наслідків. Фільми мають потужне джерело мотивації – дтвлячись фільм, можна побачити героїв, чий приклад буде надихати нас у реальному житті. Розваги - позитивні емоції, які приносять фільми, приносячи користь для фізичного та психічного здоров'я. Сльози та розплач - сумні фільми можуть витягнути глибоко приховані емоції на поверхню. Переосмислення негативних думок - через фільми ви зможете побачити свою ситуацію з іншого боку, що дозволить вам задуматися і змінити своє ставлення до неї. Розвиток креативності – різні точки зору, представлені у фільмах, можуть змінити моделі мислення та заохотити нас до творчості, гнучкості та інноваційного підходу. Фільми допомагають впоратися з втратою – фільми можуть пом’якшити розбите серце та втрату, щоб допомогти вам пережити ці часи.[5]

Кіно відіграє глобальну роль у житті людей, пов'язане з ними набагато сильніше, ніж, скажімо, театр. А пізніше, коли з'явився телевізор, можна навіть не ходити в кінотеатр, а дивитися улюблені фільми вдома. Фільми з дитинства вчать дітей розрізняти добро і зло, жаліти когось або радіти витівкам героїв. А якщо це документальний фільм, то він ще й розповідає багато цікавого та пізнавального, про історію, традиції та культуру різних країн світу, або це програма про тварин, яких дуже багато, і кожна глядач обов’язково знайде щось до душі [6].

Кіноіндустрія постійно розвивається, сьогодні новітні технології, сучасні спецефекти та графіка захоплюють глядача. А якщо 3D, то взагалі відчуваєш, що потрапив з героями чи то на безлюдний острів, чи то на батискаф підводного човна.

У наш час прогрес дійшов до того, що можна навіть завантажувати фільми і дивитися їх ще до того, як вони вийдуть у масовий прокат. Основна мета

кінематографа – додати фарб у повсякденне життя та додати в нього позитиву. І для цього кінематограф робить велику роботу.

Сьогодні практично не можливо знайти людину, яка не любить подивитися вдома захоплюючий серіал чи фільм, або поцікавитися якоюсь касовою новизною в кінотеатрі. Ми можемо просто таким чином деколи провести своє дозвілля. Є такі «кіномани», які без яскравих емоцій та якихось оригінальних сюжетів, якими наділений сучасний кінематограф, не можуть прожити ані дня.

Якщо ми бачимо, улюблений герой показує правильний приклад, який нас мотивує, він може кардинально змінити нас на краще, мотивувати нас на покращення свого життя. Багато молодих дівчат та хлопців копіюють стиль одягу та поведінки, прагнуть бути стильнішими та сучаснішими.

Переглядаючи комедію, де показують веселий заразливий сміх, вона може позитивно впливати на загальний стан здоров'я та продовжити життя.

Фільми – це наша реальність, яка відображена в ньому, а ми ніколи не втомлюємося спостерігати за собою. Кіно ніколи не втратить свою актуальність, так як воно паралельно змінюється з нашим життям, а отже завжди буде цікавим багатьом глядачам.

Для пошуку потрібної інформації в Інтернеті за допомогою ключових слів або якихось фраз ми використовуємо спеціальні програми - пошукові системи.

Вони здатні швидко показувати результати , навіть з мільйоних вебсайтів у мережі, безперестанку сканують Інтернет, індексуючи кожну знайдену сторінку.

Хоча на ринку домінують окремі пошукові системи, існує безліч систем, які ми можемо використовувати. Якщи користувач вводить запит в пошукову систему, то повертається сторінка її результатів (SERP), яка показує знайдені сторінки в порядку їх відповідності результату. В різних пошукових системах метод визначення цього рейтингу є різним.

Пошукові системи дуже часто міняють свій алгоритм, тобто програми, які оцінюють отримані результати, для покращення роботи користувача. Вони намагаються зрозуміти, як користувачі здійснюють пошук, і відповісти їм

найкращим чином на їхні запити. Тим самим надається перевага найякіснішим і найбажанішим сторінкам.

Для функціонування більшості пошукових системпотрібно виконати три основні кроки:

Сканування – для перегляду інформації в Інтернетіпошукові системи використовують програми - павуки, боти або сканери. Вони можуть це виконувати кожних пару днів, тому вміст може бути застарілим доти, поки вони знову не почнуть сканувати ваш вебсайт.

Індексування – здійснюється пошуковою системою за допомогою «ключових слів», де система намагається зрозуміти та класифікувати вміст вебсторінки. При дотриманні найкращих методів SEO можна дати пошуковій системі зрозуміти ваш вміст, для оцінювання відповідних пошукових запитів.

Рейтинг – результати пошуку формуються відповідно деяким факторам. Вони можуть містити щільність ключових слів, швидкодію і посилання. Мета кожної пошукової системи – представити користувачу найбільш бажаний результат.

Більшість пошукових систем для покращення свого рейтингу дають користувачу поради щодо його сторінки, правильні використовувані алгоритми, які будуть охоронятися та часто змінюватися, для уникнення неправильного використання. При дотриманні найкращих способів пошукової оптимізації (SEO), ми можемо переконатися в тому, що:

Пошукові системи можуть легко просканувати ваш вебсайт. Також можна їм запропонувати просканувати новий вміст. Вміст буде проіндексованим за певними ключовими словами, тому він буде відображатися для відповідних результуючих пошукових запитів.

При пошуку необхідних даних, ми можемо побачити різницю між швидким і повільнішим додатком, яка полягає в точному використанні пошукового алгоритму. Цей алгоритми є основним і базовим кроком при обчисленні і виконується за допомогою покрокового методу, щоб визначити місця розміщення конкретних даних серед всієї колекції.

Для завершення процедури пошуковими алгортмами використовується ключі пошуку, і очікується, що буде повернено статус успішного або неуспішного (у логічному значенні true або false).

1.2 Аналіз програмного забезпечення для відтворення відеоконтенту

На теперішній час можна знайти безліч різних вебсайтів для створення фільмів, які мають свої переваги та недоліки. Нижче наведемо приклади.

Free-generator – (рис. 1.1) – генератор, який дозволяє швидко знайти цікаві фільми. Для цього потрібно виконати всього три простих кроки:

- вибрати жанр: один або всі;
- вибрати країну;
- встановити період.

Для отримання результату, необхідно натиснути «генерувати фільм». Тут ми можемо побачити всю потрібну інформацію: назва фільму, рік випуску, тривалість, опис сюжету, а також рейтинг «Пошуку фільмів» та IMDb. [7]. Недоліком цього сайту є відсутність інтерфейсу українською мовою та некоректне генерування фільмів.

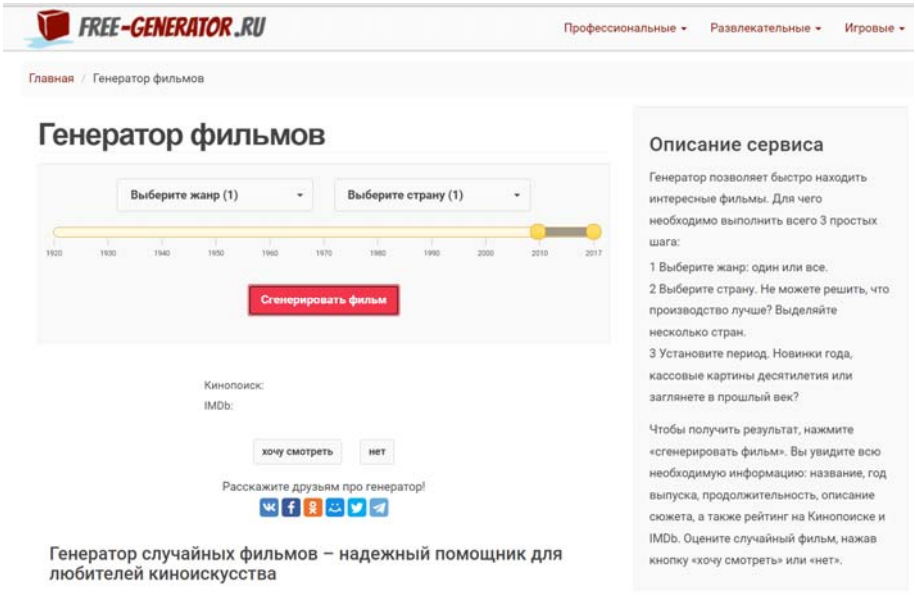

Рис. 1.1. Інтерфейс Free-generator

Generator-online (Рис.1.2) – це генератор фільмів онлайн. Дана програма обирає варіант із високим рейтингом із десятків тисяч фільмів у 19 жанрах, а також з понад 100-річною історією світового кіно.

Наефективніший метод знайти хороший фільм - це в сервісі задати параметри фільму. Потрібно встановити такі параметри: жанр і рік випуску, натиснути «Створити результат» і ви отримаєте інформацію з коротким описом потрібного вам фільму. [8]. Недоліком є застарілий інтерфейс і база даних, яка створює фільми з низьким рейтингом. Також мало інформації про згенерований фільм, і неможливо переглянути трейлер до фільму

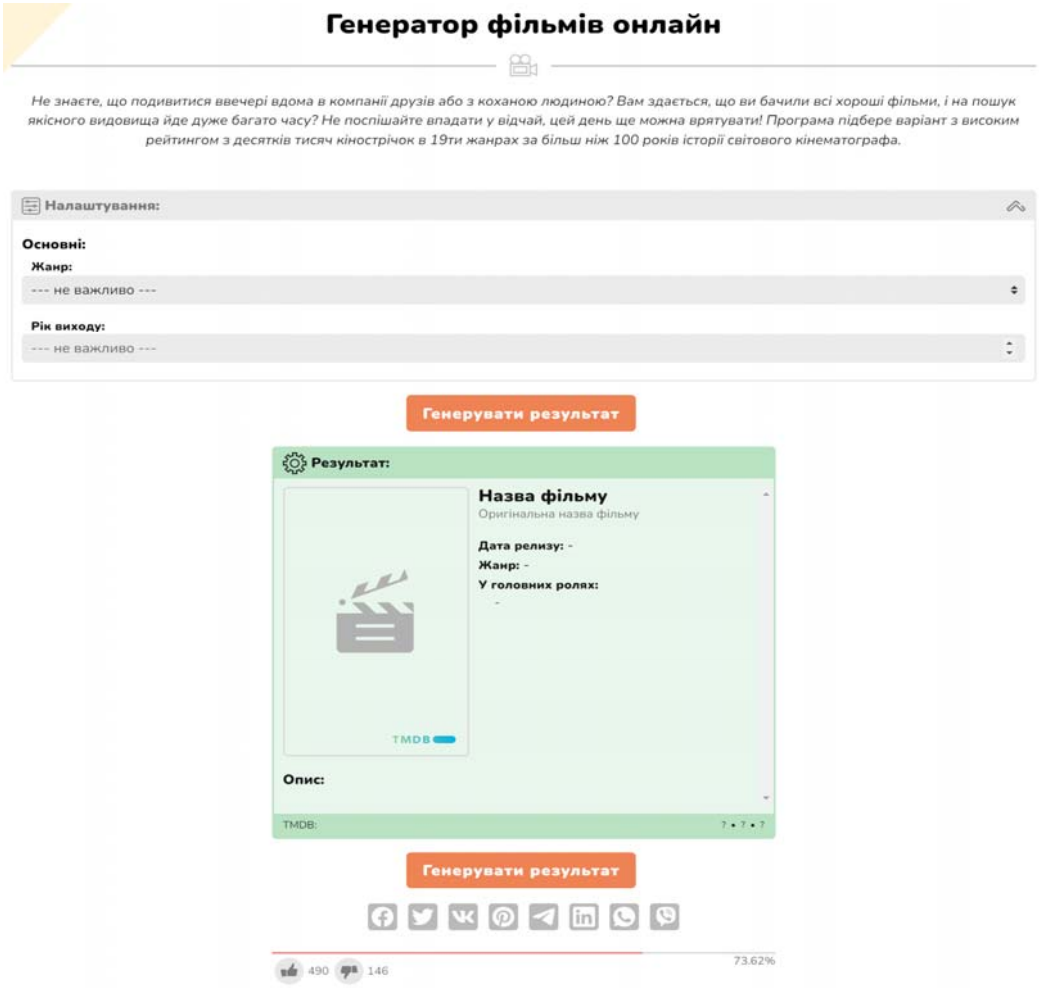

Рис. 1.2. Інтерфейс Generator-online

Coolgenerator (рис. 1.3) — генератор випадкових фільмів. Він містить 5000 найкращих фільмів з 1902 року до наших днів. Ці фільми містять майже всі жанри фільмів, пригодницькі, бойовики, комедії тощо. Ви можете отримати

випадковий фільм за допомогою цього генератора.

Користуватися цим генератором дуже просто, спочатку вибираєте фільтри, такі як: виберіть жанр фільму, рік випуску, рейтинги. У переліку виведених фільмів кожен з них містить назву фільму, рік випуску, рецензію на фільм, рейтинг, що може допомогти вам вибрати фільм. Недоліком є застарілий візуальний інтерфейс, англомовна розкладка, занадто мало фільтрів.

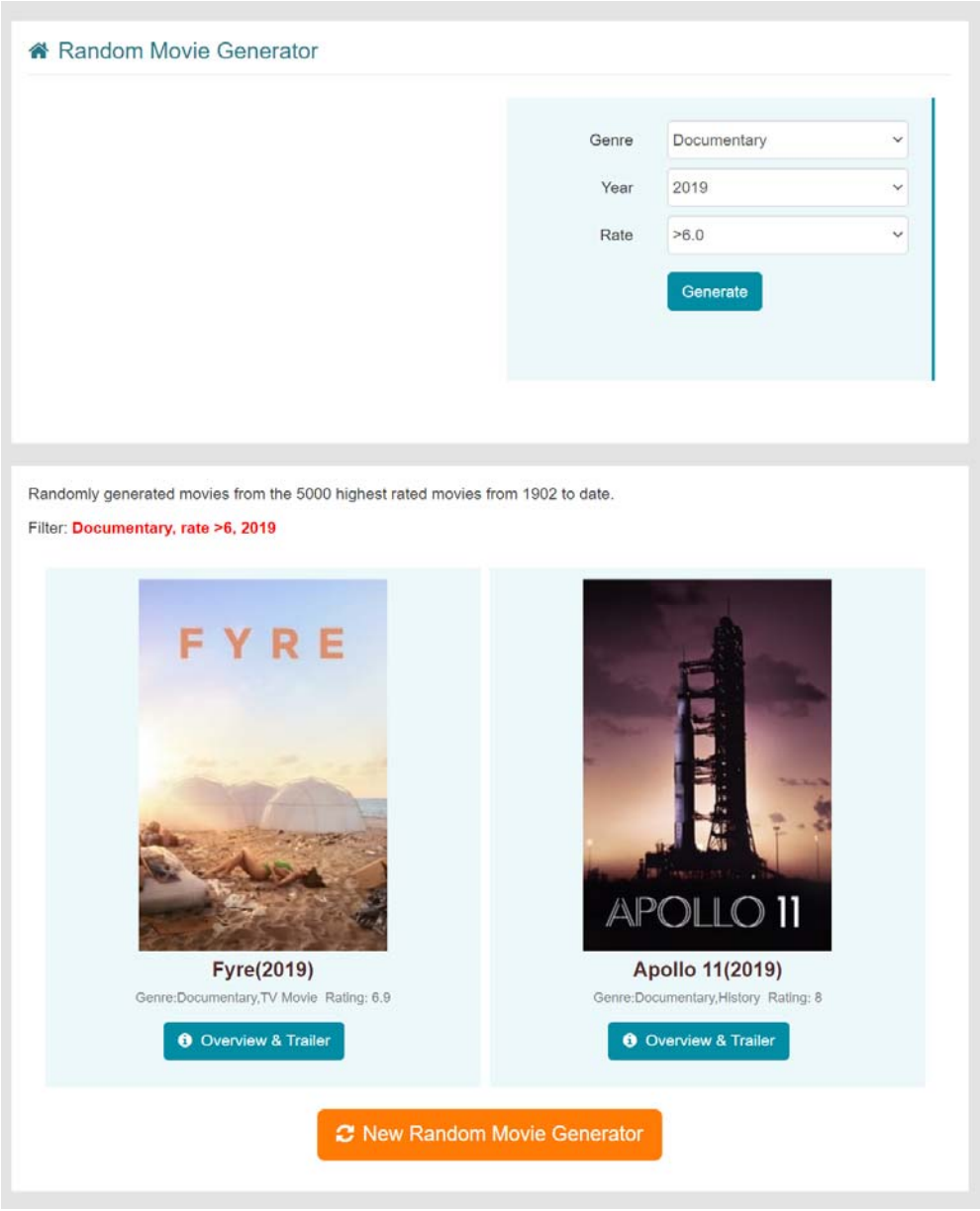

Рис. 1.3. Інтерфейс Cool generator

Порівняння аналогів з розроблюваним сервісом генерації подано в таблиці 1.1.

Таблиця 1.1 – Порівняльні характеристики програмних продуктів для генерування відеоконтенту

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

Враховуючи недоліки вищевказаних сайтів, можна зробити висновок про доцільність розробки власної реалізації. Буде створено україномовний вебсайт для генерації відеоконтенту на основі різних фільтрів.

1.3 Аналіз методів і засобів реалізації програмного забезпечення

Випадкові номери зазвичай використовуються для створення таких додатків, як кістки для настільних ігор, азартних програм тощо. Зазвичай генерація випадкових чисел потребує багато часу. В мові програмування Java цього можна досягнути, використовуючи три методи. Вони розглядаються нижче в розділі «Функції генератора випадкових чисел на Java».

У Java випадкові числа можна генерувати за допомогою 3 методів:

– math. випадковий метод;
– java.util.Random class;
– клас ThreadLocalRandom.

Клас Java Math надає низку методів для роботи з обчисленнями, такими як

логарифми, середнє значення, експоненціальний і так далі. Random() — це один із методів, який повертає додатне подвійне значення між 0,0 і 1,0, де 0,0 — включно, а 1,0 — виключно. Цей метод можна використовувати з параметрами або без них. Якщо параметри вказані, згенероване випадкове число буде в межах зазначеного параметра.

Клас Java.util.Random призначений для генерування випадкових чисел різних типів даних, таких як float, long, integer, double, Boolean тощо. Також можна передати діапазон чисел як аргументи для генерації випадкового числа в цьому діапазоні. Щоб використовувати цей клас, вам потрібно імпортувати клас Random java.util (java.util.Random). Після імпортування цього класу потрібно створити екземпляр і викликати такі методи, як next long (), nextInt () тощо, використовуючи цей екземпляр.

Клас ThreadLocalRandom - це спеціалізований тип класу Random, представлений у версії Java 1.7. ThreadLocalRandom.current (). NextInt () є одним із поширених методів, які використовуються для генерації випадкових чисел. Зазвичай використовується в багатопоточних програмах.

Java містить багато функцій, які ми можемо використовувати в своїх програмах. Це допоможе нам скоротити час обробки та кількість рядків коду. Генерація випадкових чисел є одним із завдань, де можна використовувати окремі функції. Цей документ охоплює різні методи досягнення цього[9].

Рандомізація — це розташування чи вибірка об'єктів у довільному порядку. Для випадкового вибору чисел при експерименті, можна використовувати лотерею або таблицю випадкових чисел. Також в якості прикладу можна використовувати рандомізацію для вибору послідовності окремих дослідів при їх плануванні тощо.

Рандомізація дозволяє контролювати систематичний вплив різних неконтрольованих факторів, а також забезпечує об'єктивність при виборі потрібного об'єкту.

Плануючи проведення дослідів, окремі з них потрібно проводити в певній послідовності, що встановлюється за допомогою таблиці випадкових чисел або

якоїсь процедури, яка забезпечує випадковий характер проведення дослідів. Рандомізація позволяє контролювати систематичний вплив неконтрольованих факторів [10].

Псевдовипадковою називається така послідовність, яка здається несистематичною та випадковою, але насправді вона була згенерованою суто детермінованим процесом, який нам відомий як псевдовипадковий генератор. Таким генераторам переважно задається початкове значення і при допомозі певних алгоритмів отримуються випадкові послідовності. У цьому випадку псевдовипадкові генератори можуть розглядатися як розповсюджувачі випадковості. Комп'ютери є машинами, які мають свій термін роботи, іпостійно роблять тільки те, на що вони запрограмовані, і відповідно немає потреби посилатися на комп'ютери, як на джерело справжньої випадковості. Найкраще, що робить комп'ютер, це генерування псевдовипадкової послідовності, яка, хоч і виглядає випадковою, насправді такою не являється. Випадкова послідовність насправді бути згенерована тільки за допомогою апаратної реалізації генератора, який може використовувати окремі фізичні явища, щоб отримати випадкові числа, наприклад, шум, який може бути створений напівпровідниковими пристроями, інтервали між перериваннями пристрою або натисканнями клавіш, молодші біти оцифрованого звуку, температура повітря тощо. У сучасних потужних військових криптосистемах використовуються генератори випадкових чисел (RANG), які являють собою певні плати або зовнішні пристрої, що підключаються до комп'ютера через порт уведення-виведення. Незважаючи на труднощі, які можуть виникати при проектуванні генераторів псевдовипадкових чисел (PSN), їх масово використовують в прикладних комп'ютерних програмах, і вони легко інтегруються з усіма типами комп'ютерних систем. Тому більшість прикладних комп'ютерних програм на даний час використовують HVDC для генерації потрібних випадкових даних.

Випадкові числа являються головним елементом, який забезпечує обмежений доступ до інформації. Також вони є основним елементом криптографії, протоколів безпеки, цифрового підпису, та основними гарантіями

надійності під час спілкування[11].

Генератор псевдовипадкових чисел — це певний алгоритм, що генерує послідовність чисел, елементи якої вже не залежать один від одного і підчиняються заданому рівномірному розподілу.

Проаналізувавши методи та засоби використання сервісу генерації відеоконтенту на основі фільтрів, було прийнять рішення для використання псевдовипадкового методу випадкових чисел, оскільки він дозволяє найшвидше отримувати дані з сервера, записувати їх та виконувати будь які інші операції з даними за допомогою API.

1.4 Постановка задачі роботи

Проаналізувавши стан послуг генерації, порівнявши існуючі аналоги та проаналізувавши методи і засоби реалізації програмного продукту, можна визначити наступні завдання, які потрібно виконати для створення програмного продукту:

- розробити методику генерації відеоконтенту;
- розробити та впровадити метод для генерегуван відеоконтенту на базі плівкової фільтрації;
- розробити програмне забезпечення для генерування відеоконтенту в реальному масштабі часу;
- провести тестування програмне забезпечення для генерування відеоконтенту в реальному масштабі часу.

Висновки до розділу I

У цьому розділі розглянуто аналіз стану програмного забезпечення для генерації відеоконтенту, який показав актуальність створення власного програмного забезпечення. Розглядалися такі аналоги як: Free-generator, Generator-online, Coolgenerator. Порівняння вже створених аналогів показало, що

є потреба у розробці власного сайту. Також проведено аналіз методів реалізації програмного продукту. Вибрано методи роботи з генерацією псевдовипадкових чисел. У результаті були розроблені основні завдання, які необхідно виконати при створенні програмного забезпечення.

РОЗДІЛ ІІ
АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ГЕНЕРАЦІЇ ВІДЕОКОНТЕНТУ

2.1. Розробка структурної схеми програмного забезпечення

Структурна схема вебзастосунку показує призначення головних структурних модулів програми, їхню взаємодію та інтерфейс з користувачем.

У вебзастосунку для генерування відеоконтенту відкриваються наступні модулі: база даних, модуль роботи з базою даних, інтерфейс браузера, модуль формування html сторінок, обробник вебзапитів клієнта.

Для входу в програму відкивається головне меню, з якого починається налаштування програми по інших її складових частинах.

В системній інженерії та розробці програмного продукту використовується функціональна блок-схема, яка описує функції програмного забезпечення та взаємозв’язки системи.

Крім того блок-схема може використовувати додаткові символи для схематичного відображення заданих властивостей.

Функціональні блок-схеми були використані в різноманітних додатках, починаючи від системної інженерії ідо створення програмного забезпечення. Вони стали необхідними при проектуванні складених систем для того, щоб «досконало зрозуміти з зовнішнього дизайну роботу нинішньої системи та взаємозв’язок кожної частини з цілим».

Функціональні блок-схеми можна згрупувати по різних специфічних типах. Наприклад, блок-схема функціонального потоку комбінує функціональну блок-схему та звичайну. На основі використання певних методів функціональних блок-схем було побудовано багато методологій створення програмного забезпечення. З галузі промислових обчислень прикладом є Function Block Diagram (FBD) – це графічна мова, яка призначена для проєктування програмованих логічних контролерів[12].

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

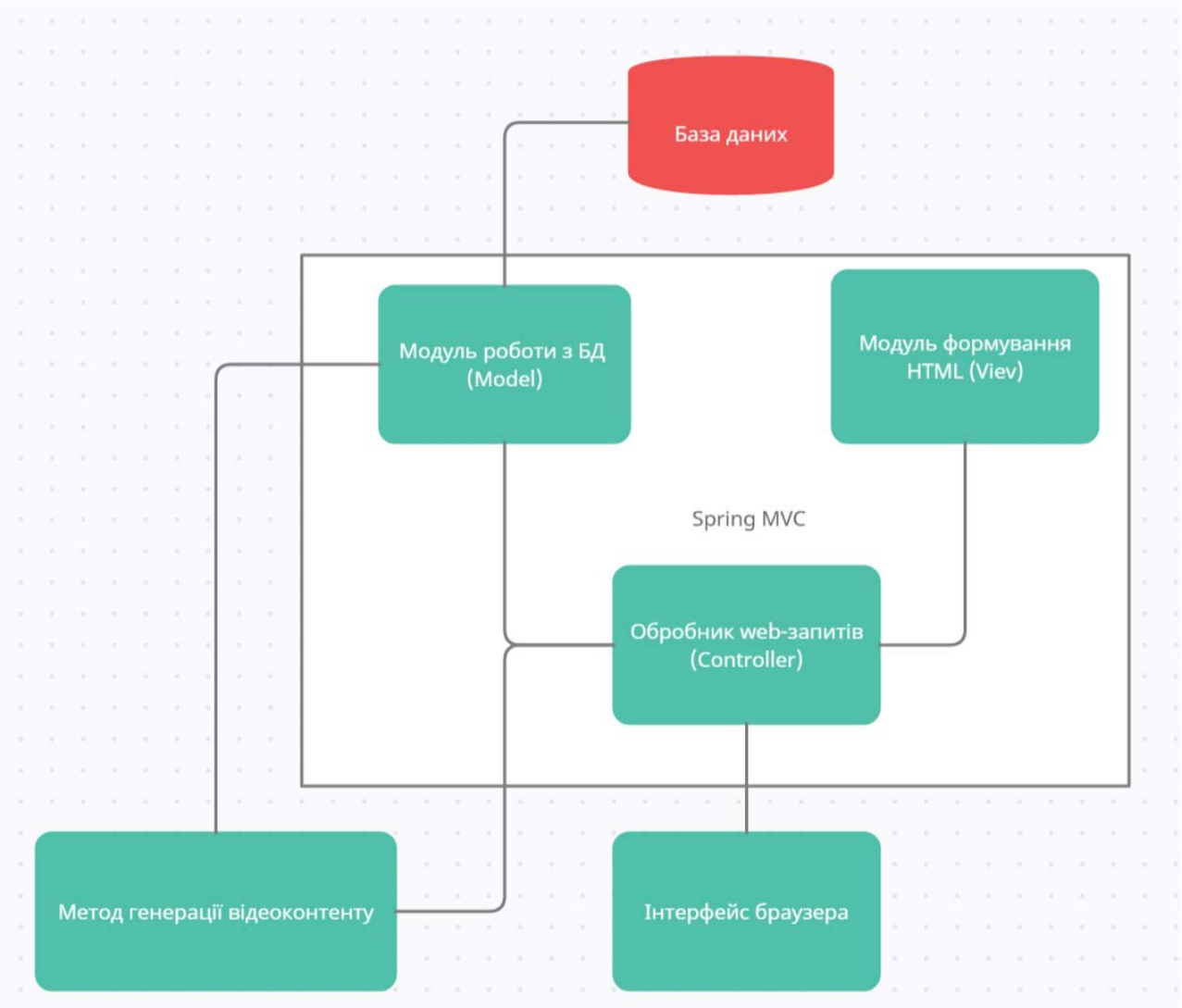

Рис.2.1. Структурна схема функціонування веб-застосунку MovieGenerator

Користувач здійснює запит на генерацію та пошук, обробник webзапитів в свою чергу передає його в модуль роботи з базою даних, який виконує та налаштовує запит під базу даних, і з неї бере ту інформацію,яка йому потрібна, а потім, повертаючись в обробник web-запитів, переходить в модуль формування HTML, який формує інтерфейс для користувача.

При розробці складних і великих програм програміст повинен оперувати спеціальними прийоми, щоб отримати раціональну структуру програми, яка значно скорочує обсяг програмування.

Ієрархічність модулів програми показана у схемі ієрархії. Але дана схема не відображає порядок виклику модулів чи функціонування програми. Вона, зазвичай, доповнюється специфікацією функцій, які виконуть модулі.

Перед розробкою схеми ієрархії, необхідно розробити зовнішні специфікації програми та функціональний опис програми разом з описом змінних носіїв даних. Особливу увагу слід звернути на ієрархії типів структурованих даних та їхні коментарі.

Декомпозиція програми на підпрограми здійснюється по принципу від цілого до частини. Створення функціонального опису та схеми ієрархії є ітераційним процесом, а вибрати оптимальний варіант можна за багатьма критеріями. Розподіл повинен забезпечувати зручний порядок введення частин в експлуатацію.

Для схем ієрархії можна задати який небудь топологічний малюнок. При введенні додаткового модуля, що може не виконувати ніяких корисних функцій згідно алгоритму програми, фрагменти , які мають вертикальні виклики можуть перетворитися на виклики одного рівня. Функція нового модуля може бути лише для моніторингу, тобто викликів інших модулів у порядку[13].

Структурна схема - це вид графічної моделі, яка містить сукупність елементарних частин об'єкта та зв'язків між ними. Під елементарною частиною розуміється частина об'єкта чи системи керування, яка виконує елементарну функцію.

Будь-яка складна структура програмної системи може бути представлена як комбінація попарно зв'язаних між собою частин. Існують всього три різновиди таких зв'язків: паралельний, послідовний та зворотній.

При паралельному з'єднанні вхідне значення з'єднання є спільним для обох ланок, а вихідне формується в результаті підсумовування вихідних даних. Передаточна функція зв'язку дорівнює сумі передаточних функцій ланок.

При послідовному з'єднанні вихідне значення однієї ланки є вхідним значенням для іншої, і, отже, її передаточна функція є добутком двох ланок.

При наявності зворотнього зв'язку, одна з частинок системи передає вихідний сигнал другої ланки назад на свій вхід, де він сумується з вхідним впливом або від нього віднімається. Канал, по якому сигнал з виходу системи повторно подається на її вхід, називається зворотним зв'язком, причому в одному випадку зворотний зв'язок вважається позитивним, а в іншому - негативним [14].

Діаграма класів на деталізованому рівні абстракції описує взаємодію частин програми одна з однією, їх ієрархічну структуру та функції.

Основні види класів:

1) Конфігуратори: MvcConfig, Application, WebSecurityConfig. Відповідає за загальне налаштування веб-додатку. Вони визначають основні параметри та конфігурації програми, такі як авторизація, локалізація, конфігурація шаблону тощо.

2) Репозиторії: GenreRepo, MovieRepo, MovieRepoCustom, MovieRepoImpl, UserRepo. Вони несуть відповідальність за роботу з базою даних, з допомогою них формуються SQL запити до бази даних.

3) Сутності: фільм, жанр, роль, користувач. Вони відповідають за представлення даних у таблицях бази даних, зв’язок із програмним кодом та опис основної функціональності екземплярів даних класів.

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

Діаграма класів - це основна логічна модель проєктованої системи. Вона представляє статичні аспекти програми. Діаграма класів застосовується не тільки для візуального представлення, опису та документування різних аспектів системи, а також для створення коду програмного додатка.

Діаграма класів описує атрибути та методи класу, а також обмеження, які накладаються на систему. Їх широко використовують в моделюванні об'єктно-орієнтованих систем, так як вони є основними єдиними діаграмами UML, які відображають безпосередньо з об'єктно-орієнтованими мовами.

Діаграма класів включає сукупність класів, інтерфейсів, кооперацій асоціацій та обмежень. Вона показує структурні відносини, які існують між класами.

Такі UML діаграми, як діаграма активності та діаграма послідовності, можуть показувати лише послідовність дій в програмі, однак діаграма класів зовсім інша. Це найпопулярніша серед діаграм UML у спільноті програмістів.

Діаграма класів вебзастосунку зображена на рисунку 2.2.

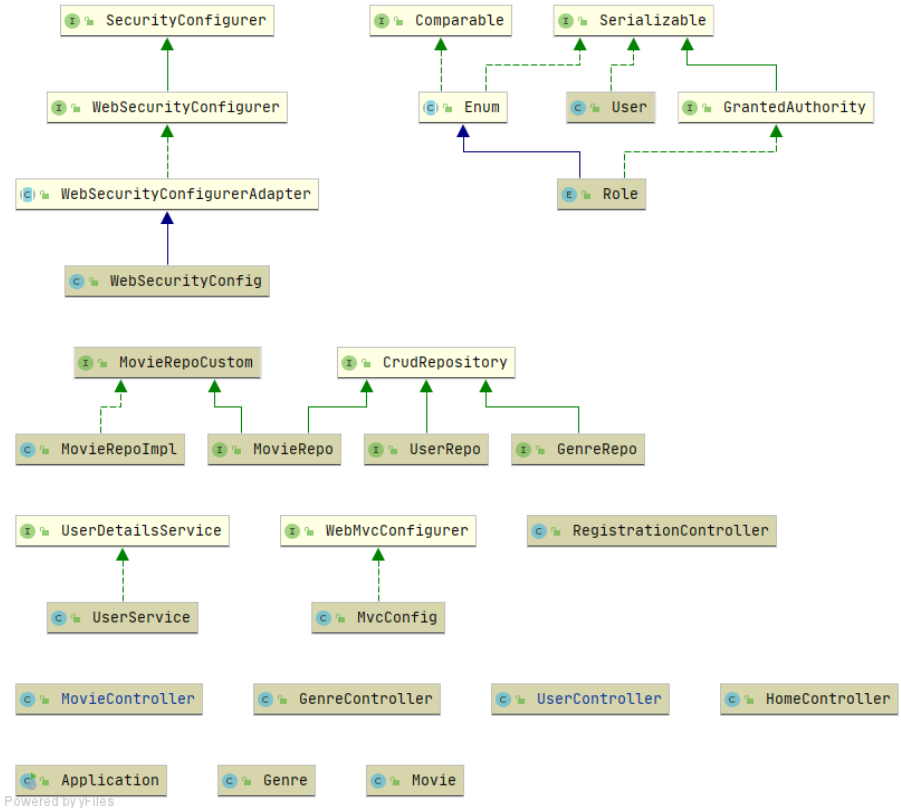

Рис. 2.2. Діаграма класів веб-застосунку MovieGenerator

2.2. Розробка методу генерації відеоконтенту

Метод пошуку - це правило для визначення стратегії пошуку інформації, що міститься в певній структурі даних. Дані можуть представлятися рядком, масивом, зв’язаним списком, хеш-таблицею, деревом пошуку, неструктурованим набором тощо.

Вхідними даними алгоритмів для пошуку є ключ пошуку, тобто поле запису, за яким здійснюється пошук, і функція, яка визначає правильність знайденого результату до заданого ключа. Це може бути логічна функція «True» або «False», функція обчислення еквівалентності чи релевантності тощо. Пошук потібного завдання здійснюється за допомогою графічного або аналітичного опису, що базується на таких поняттях, як теорії графа, моделі, алгебра та алгебраїчні системи.

Ефективність алгоритму пошуку залежить від типу використовуваних даних та методів їх обробки. Оцінюється згідно складності алгоритму або максимального часу виконання. Складність визначається максимальною кількістю операцій порівняння ключів пошуку для найбільш песимістичного випадку. Це значення позначається як On, де n – кількість виконаних порівнянь. Варіантами виконання алгоритмів пошуку є найкращий, середній і найгірший варіанти. Значення середнього варіанту поведінки пошукового алгоритму показує корисність алгоритму.

Можна також задати пошук інформації залежно від завдання, а саме: пошук у структурах, пошук семантичної інформації, пошук оптимального рішення, пошук шляху, пошук шаблону даних, пошук подібності чи відмінності в мультимедійних даних - пошук обличчя, шаблону розпізнавання тощо Алгоритми пошуку часто використовують разом з алгоритмами сортування.

Алгоритми знаходження значення масиву даних можуть не мати особливих вимог до вхідних даних, і ми порівняємо такі алгоритми між собою. Якщо про вхідні дані щось відомо заздалегідь (наприклад, що вони розташовані в алфавітному порядку), то до таких даних можна застосувати інші швидші алгоритми.

За допомогою розширених фільтрів можна розширити функціональність пошуку для користувача. Ввівши необхідну кількість фільтрів, ми прискорюємо і покращуємо пошук мінімум в 1,5 рази.

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";
    }
```

Рис. 2.3. Фрагмент коду реалізації методу генерації відеоконтенту

2.3. Розробка алгоритму роботи програмного забезпечення

Основним модулем для даного вебдодатку є випадкова генерація нових рекомендованих фільмів для перегляду. На рисунку 2.4 зображена блок схема роботи алгоритму генерації псевдовипадкових чисел.

Метод діаграми пріоритету (PDM) — це візуальне представлення проєкту, в якому зображені дії, що залучені в проєкт. Це метод побудови мережевої діаграми розкладу проєкту, який використовує блоки/вузли для представлення діяльності та з’єднує їх за допомогою стрілок, які показують залежності.

Метод діаграми пріоритетів (PDM) являється інструментом для планування діяльності в плані проєкту. Це метод побудови мережевої діаграми розкладу проєкту, який використовує спеціальні блокидля представлення видів

діяльності, які називаються вузлами, та з’єднує їх стрілками, які показують залежності. Його також називають методом активності на вузлі (AON).

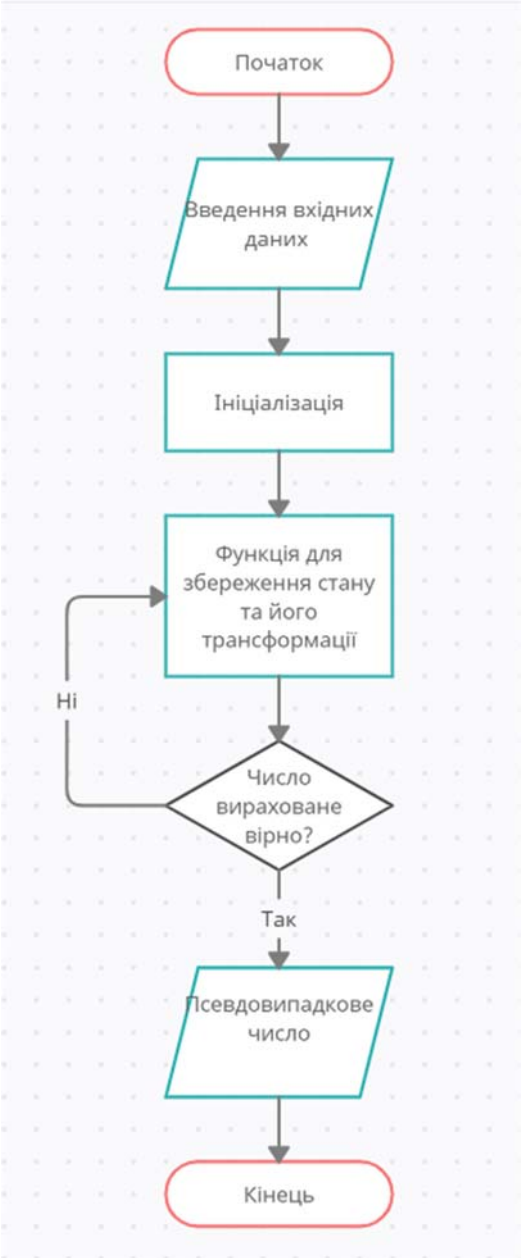

Рис. 2.4. Блок-схема алгоритму генерації псевдовипадкових чисел

На рисунку 2.5 зображено блок-схему авторизації користувача в системі.

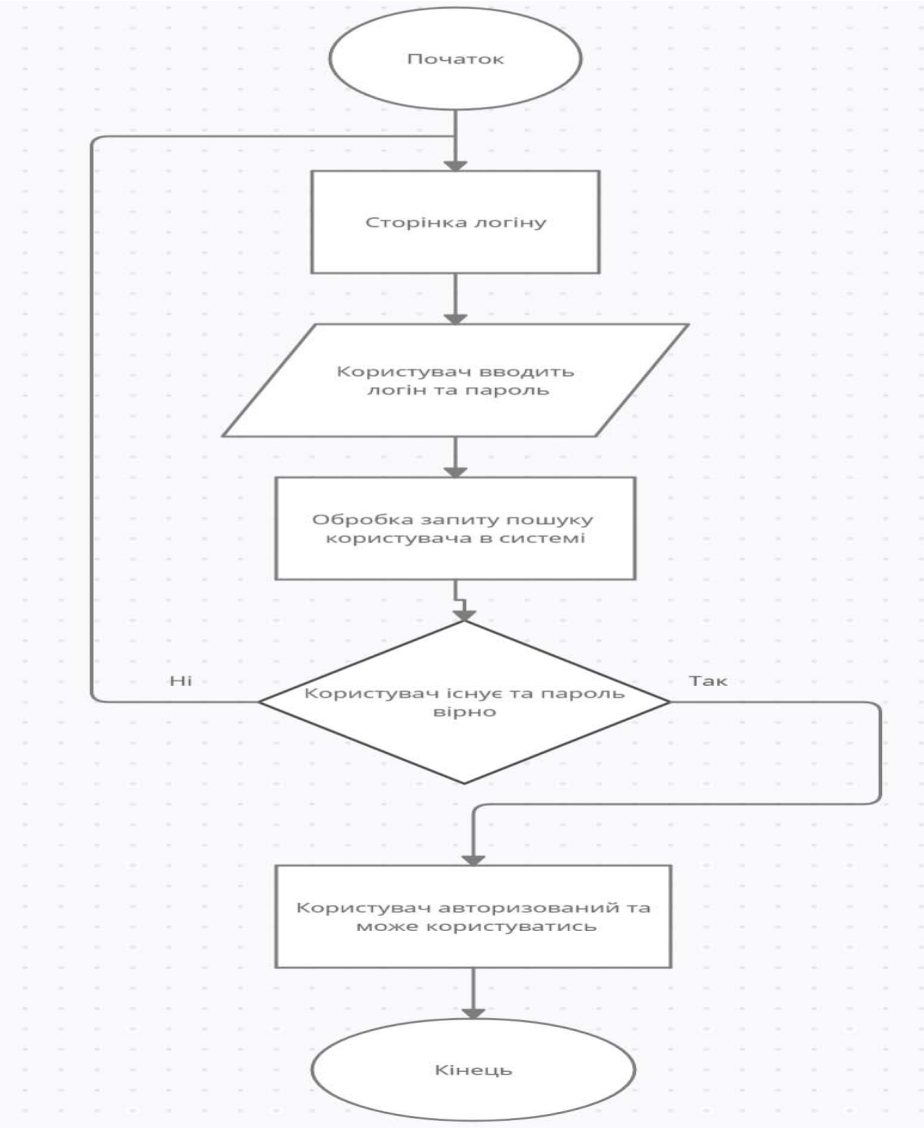

Рис. 2.5. Блок-схема авторизації користувача в системі

Діаграма варіантів використання наочно зображує різні сценарії взаємодії акторів (користувачів) з прецедентами (варіантами використання) та описує функціональні аспекти системи (бізнес-логіку).

Діаграми випадків відіграють важливу роль не тільки в спілкуванні між збирачами вимог до проекту та потенційними користувачами.

Діаграми прецедентів, доповнені бізнес-логікою та детальними специфікаціями прецедентів, як вихідна інформація, успішно використовуються учасниками розробки проекту на всіх його фазах (генезис, дизайн, програмування, тестування, документація).

Діаграма прецедентів, показана на рисунку 2.6, показує взаємодію користувача з відношенням, його можливі дії та реакцію системи на них.

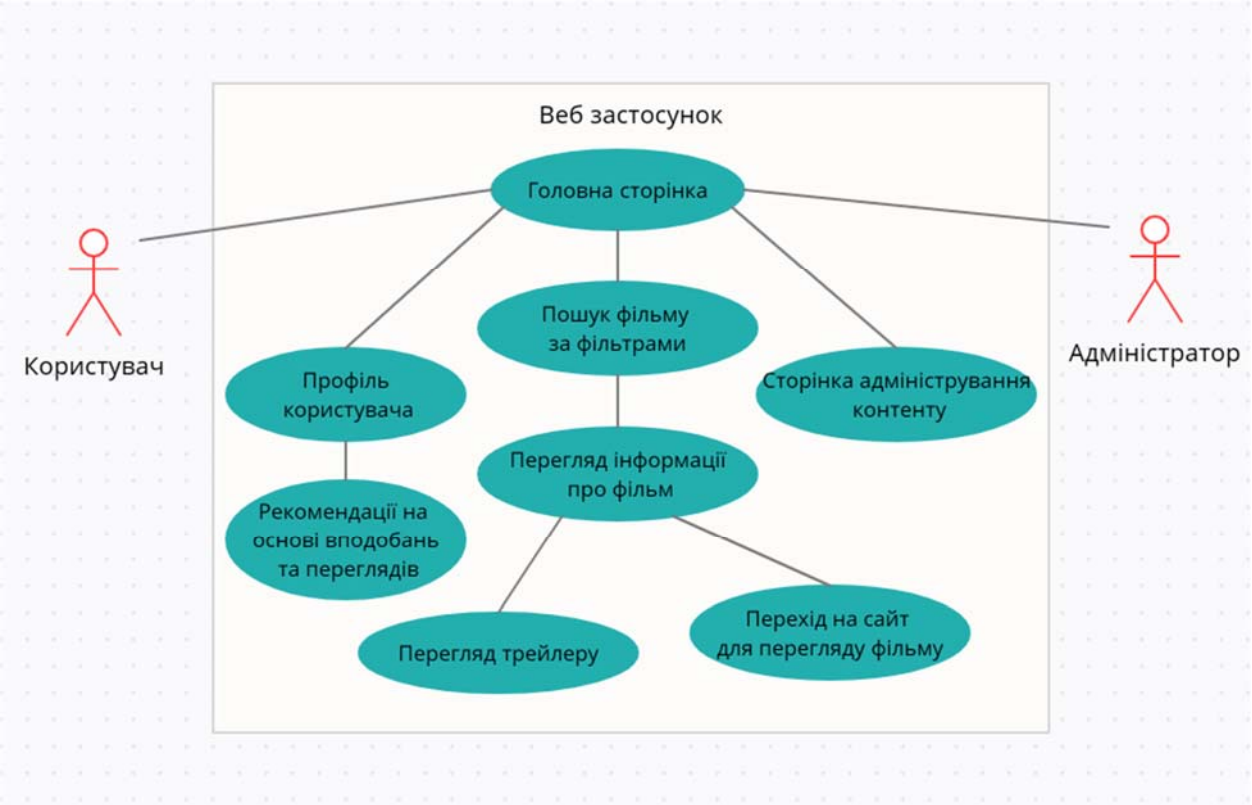

Рис. 2.6. Діаграма прецедентів веб-застосунку Movie Generator

Загальна UML діаграма класів вебзастосунку зображена на рисунку 2.7. В ній зображені всі класи, з атрибутами та методами

Таким чином, дослідивши та удосконаливши метод генерації відеоконтенту, який за допомогою розширених фільтрів, дозволяє розширити функціональність пошуку для користувача. При введенні потрібної кількості фільтрів, ми зменшимо час вибору та покращимо пошук в мінімум 1,5 рази. Провівши опитування 25 людей, які користовувалися таким пошуком, ми вони підтвердили його унікальність та були в цьому зацікавленні.

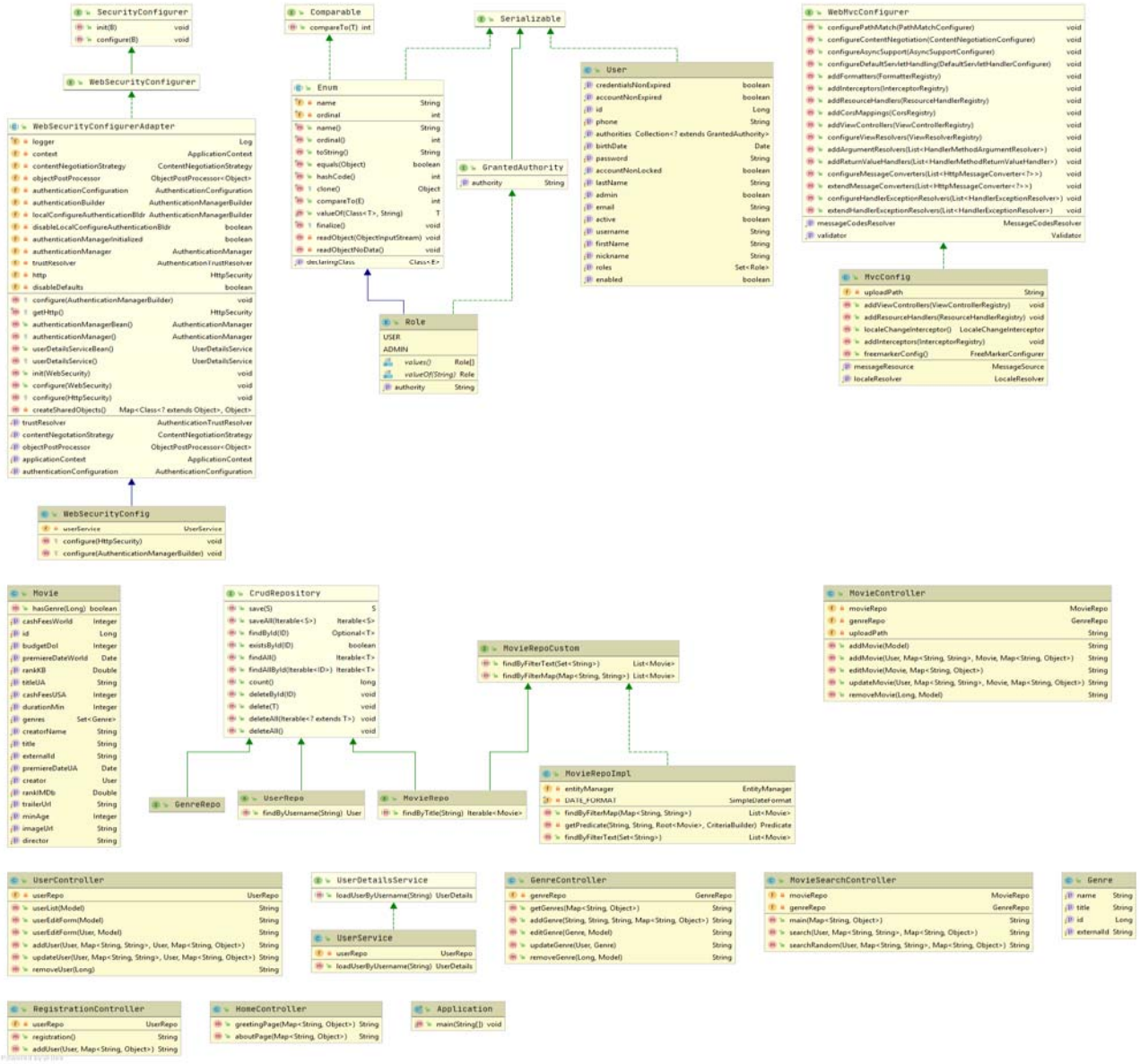

Рис. 2.7. Загальна UML діаграма класів веб застосунку

Висновки до розділу II

У цьому розділі розроблено архітектуру та спроектовано базові функції програмного забезпечення для генерації відеоконтенту. Розроблено діаграму функціональної структури, діаграму класів, діаграму прецедентів та блок-схему алгоритму генерації псевдовипадкових чисел.

Удосконалено метод генерації відеоконтенту, який на відміну від існуючих, використовує розширені фільтри для генерування відеоконтенту, що дозволяє розширити функціональність процедури пошуку відео для користувача.

РОЗДІЛ ІІІ
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ГЕНЕРУВАННЯ ВІДЕОКОНТЕНТУ В РЕАЛЬНОМУ МАШТАБІ ЧАСУ

3.1 Обгрунтування вибору мови програмування

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

В принципі, ви можете написати програму будь-якою мовою. Питання в тому, чи буде він працювати ефективно і без збоїв? Тому для вирішення різноманітних завдань необхідно вибирати ті мови програмування, які найбільш відповідають.

Java — це мова програмування, створена за традиціями C і C++, але з іншою надією: «напиши один раз, запускай всюди». Це мова, яка не залежить від платформи, тому програми можуть працювати на різних видах комп’ютерів. Ще одна особливість, яка виділяє Java, це її API, який є бібліотекою з тисяч класів. Мова Java є спрощеною версією C++, але величезний API іноді ускладнює вивчення мови. для розробників програмного забезпечення та IT-фахівців це добре відома мова.

Java багатопотокова, що означає, що кілька завдань можуть виконуватися одночасно, і користувачі можуть створювати інтерактивні програми, які працюють без проблем. Завдяки його функціям безпеки можна розробляти системи без вірусів і несанкціонованого доступу. Методи автентифікації засновані на шифруванні з відкритим ключем.

Java є об’єктно-орієнтованою мовою програмування та має платформу віртуальної машини, з допомогою якої можна створювати скомпільовані програми, які функціонують практично на кожній платформі. Ява пообіцяв: «Напиши один раз, біжи куди завгодно».

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 основні принципи об’єктно орієнтованого прорамування:

– Абстракція;
– інкапсуляція;
– спадкування;
– поліморфізм.

Java не має складних функцій, таких як покажчики, перевантаження операторів, множинне успадкування, явний розподіл пам’яті, тому вважається є однією з простих мов програмування.

Компілятор Java здатний виявити навіть ті помилки, які нелегко виявити іншими мовами програмування.

У java відсутні вказівники, тому ми не можемо отримати доступ до масивів за межами, тобто виникає виняток ArrayIndexOutOfBound, якщо ми намагаємося це зробити.

Ви можете створювати розподілені програми за допомогою мови програмування Java. Віддалений виклик методів і Enterprise Java Beans використовуються для розробки розподілених програм на Java.

Мова Java підтримує багатопотоковість. Це функція Java, яка дозволяє одночасно запускати декілька частин програми для максимального використання ЦП.

JavaScript — це легка мова програмування («мова сценаріїв»), яка використовується для створення вебсторінок. З допомогою JavaScript можна вставляти динамічний текст у HTML. Вона також відома, як мова браузера. JavaScript(JS) не має подібнсті і не пов’язана з мовою Java. Обидві мови мають синтаксис, схожий на C, і широко використовуються в клієнтських і серверних веб-додатках, але є лише деякі подібності.

JavaScript була створена насамперед для маніпулювання DOM. Раніше сайти були переважно статичними, після створення JS почали створювати динамічні вебсайти. Функції в JavaScript є об’єктами. Вони, як і будь-який інший об’єкт, можуть мати властивості та методи. Їх можна передати як аргументи іншим функціям. Може працювати з датою та часом. Виконує перевірку форми, навіть якщо форми створено за допомогою HTML. Компілятор не потрібен.

Подібно до серверних мов сценаріїв, таких як PHP і ASP, код JavaScript можна вставляти в HTML вебсторінки. Вихід на стороні сервера відображається в HTML, але JavaScript залишається видимою у вихідному коді вебсторінки. Файл може бути автономним файлом ".js", який можна відобразити у браузері.

Незалежно від того, де розміщується код JavaScript, він завжди виконується в клієнтському середовищі, для того, щоб зекономити значну пропускну здатність і пришвидшити процес виконання.

У JavaScript важливим об’єктом є XMLHttpRequest, який розроблений Microsoft. Виклик об’єкта здійснюється в XMLHttpRequest як асинхронний HTTP-запит до сервера для передачі даних на дві сторони без перезавантаження сторінки. Найбільшою перевагою мови JavaScript є те, що вона підтримується всіма сучасними браузерами та забезпечує еквівалентний результат.

Великі компанії підтримують розвиток громад, створюючи важливі проекти. Прикладом може бути Google (створений фреймворк Angular) або Facebook (створений фреймворк React.js). JavaScript використовується всюди в Інтернеті. JavaScript добре працює з іншими мовами і може використовуватися у величезних програмах. Існує багато проектів з відкритим кодом, які надають корисну допомогу під час додавання розробника JavaScript. Існує багато доступних курсів JavaScript, які дозволяють швидко та легко розширити знання цієї мови програмування.

Почати свою роботу в JavaScript не складно, тому більшість з нас прагнуть починати свою роботу в IT-секторі з вивчення цієї мови. Вона дає можливість створювати хороші інтерфейси. Є декілька способів використання JavaScript через сервери Node.js. Можна розробити цілу програму в JavaScript від початку

до кінця, використовуючи лише JavaScript. Недоліками JavaScript є те, що вона може не підходити для розробки великих програм, хоча там також будуть використовуватися накладення TypeScript.

Це стосується великих інтерфейсних проєктів. Налаштування часто бувають виснажливим завданням по кількості інструментів, які потрібно об'єднати для того, щоб створити відповідне середовище для проєкту. Це може бути безпосередньо пов'язано з роботою самої бібліотеки. Основним недоліком JavaScript є те, що код програми завжди видимий для всіх, і тому кожен може його переглянути.

Незалежно від того, яку пропорцію даних потрібно швидко інтерпретувати JavaScript, JavaScript DOM (об'єктна модель документа) є повільною і може завжди повільно відображатися в HTML. Якщо виникла помилка в JavaScript, то може бути припинено відтворення всього вебсайту. Браузери є надзвичайно толерантними до помилок JavaScript.

Зазвичай JavaScript використовується різними браузерами по-різному. Це дещо ускладнює читання та запис міжбраузерного коду. Деякі редактори HTML підтримують налагодження, хоча вони є не настільки ефективними, як інші редактори, наприклад C/C++. Тому програмісту складно виявити існуючу проблему.

Такі безперервні перетворення використовують більше часу для перетворення величини в ціле число. Це збільшує час, який потрібний для виконання сценарію, і зменшує його швидкодію.

Ruby — це об'єктно-орієнтована мова, розроблена Юкіхіро Мацумото (також відомим як Matz у спільноті Ruby) у середині 1990-х років у Японії. Все, що є в Ruby являється об'єктом, окрім блоків, але для них також є заміни, а саме procs та lambda. Метою розробки мови Ruby було те, щоб використовувати його як розумний буфер між програмістами (людьми) та базовою обчислювальною технікою. Синтаксис Ruby подібний до синтаксису інших мов програмування, таких як C і Java, тому його легко вивчити розробникам, які програмують на C та Java. В основному він підтримує такі платформи, як Windows, Mac, Linux.

Основою для заснування Ruby було багато інших мов, таких як Perl, Lisp, Smalltalk, Eiffel і Ada. Ruby - це інтерпретована мова сценаріїв, яка означає, що більшість його реалізацій виконують всі інструкції вільно та безпосередньо , без попередньої компіляції програми в конструкції машинною мовою. Розробникам Ruby також надається доступ до потужних RubyGems (RubyGems надає стандартний формат для програм і бібліотек Ruby).

Синтаксис Ruby досить простий та схожий на інші широко використовувані мови програмування, тому програмувати на ньому легко навчитися.

Програмний код, який написаний на Ruby, за розміром невеликий, але елегантний та потужний, оскільки має меншу кількість рядків.

За допомогою Ruby можна просто і швидко створювати вебдодатки, що значно зменшує обсяг роботи. Ruby є безкоштовним, тому його можна легко копіювати, використовувати, змінювати, та вносити необхідні зміни, якщо це потрібно.

Оскільки Ruby динамічна мова програмування, і завдяки цьому немає жорстких правил щодо створення вбудованих функцій, також вона дуже близька до розмовних мов.

Так як Ruby є досить новим і має свою унікальну мову кодування, то це ускладнює програмістам створювати код відразу, але після певної навичок, його легко використовувати. Але більшість програмістів, задля економії свого часу, дотримуються того, що вони вже знають і можуть розробити.

Написаний на Ruby код, складніше налагоджувати, оскільки більше часу він генерується при виконання, тому його стає важко читати під час налагодження.

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

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

Обидві мови використовують різні методи виконання коду. Java спочатку перетворює свій код на машинну мову, для щоб вона могла його зрозуміти, тому через це код Java працює швидше, ніж код Ruby. У Java і Ruby використовується успадкування, і обидві мови мають відкриті, закриті та захищені методи. Розробники та програмісти віддають перевагу Ruby, оскільки в Ruby менше рядків коду, ніж Java.

Тобто мова Ruby може використовуватися і використовується для вирішення різноманітних завдань у різних галузях промисловості. Однак Ruby найбільш використовуть для веб-розробки.

Завдяки Ruby on Rails мова Ruby стала популярною для створення веб-додатків. Це вплинуло на веб-розробку загалом та інші фреймворки. Нижче про це розповідають експерти.

У «рейках» були реалізовані інноваційні можливості, які включають безшовну інтеграцію з базою даних, створення уявлень для прискорення розробки, міграцію. Пізніше ці можливості були застосовані в інших фреймворках, включаючи Django, Laravel та Phoenix.

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

Ruby застосовується не тільки для розробки вебдодатків. На Ruby написана утиліта командного рядка Homebrew, програмне забезпечення для створення віртуального середовища розробки Vagrant, програмне забезпечення для інформаційної безпеки Metasploit та інші відомі програми.

Основними відмінностями Java від Ruby є наступні:

– Java повинна бути скомпільована перед запуском програми, а в Ruby не потрібно компілювати код;

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

– змінні у 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 у Java, так і у Ruby, оскільки воно оголошується різними ключовими словами, тобто нульове значення у Java оголошується ключовим словом «null», а Ruby воно оголошується «nil»;

– переторення в Java: об'єкти можуть бути перетворені на інші об'єкти, якщо вони відносяться до типу об'єктів, до яких наводяться. Але у Ruby перетворення не використовується, оскільки змінні динамічно типізовані, а також присвоюються будь-якому іншому типу.

Порівняння мов розробки відображено в таблиці 3.1.

Таблиця 3.1 – 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

Оглянувши, порівняльний аналіз мов програмування, можна зробити висновок, що Java є найкращим способом для розробки вебдодатку. Крім переваг над аналогами, мова програмування, яку ми використовуємо для розробки клієнтської та серверної частин, є однаковою, що значно полегшує процес розробки.

3.2 Обгрунтування вибору середовища розробки

Після вибору мови програмування для розробки програмного продукту, необхідно вибрати середовище розробки IDE.

Середовище розробки програмного забезпечення (SDE) — це середовище, яке призначене для автоматизації або розширення процедур, які задіяні в розробці програмного забезпечення. Це включає багатозадачне програмування, наприклад, керування командою та проєктами, а також великі задачі програмування, наприклад, керування конфігурацією. SDE також підтримує масштабне та довгострокове обслуговування програмного забезпечення.

Основні особливості IDE такі:

- IDE повинна мати можливість завершувати код для визначення функцій і ключових слів мови Java;
- повинна мати потужне керування ресурсами, яке допоможе вам визначити відсутні ресурси, заголовки, бібліотеки тощо;
- хороший інструмент налагодження для повного тестування розробленої програми;
- компіляція та збірка функцій.

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

Це відповідає певним стандартам компанії, тому принцип роботи буде однаковим протягом усієї розробки та допоможе кодувальникам.

Вона постачається разом з хорошими інструментами керування проєктами та документацією для автоматизації багатьох речей.

Середовище корисне для спрощення розробки додатків баз даних.Воно має функції для розробки хорошого інтерфейсу користувача з текстовими полями, кнопками тощо[15].

Примати рішення про те, яка IDE чи редактор буде відповідати нашим потребам, залежить від різних факторів, включаючи характер проєктів або

додатків, які розробляються, процес, який використовує команда розробників, рівень особистості та навики програміста, а також роль в організації. Також важливу роль у виборі IDE або редактора відіграють особисті вподобання та стандартизація інструментів.

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. Вона є доступною як ліцензована версія спільноти Apache 2 , а також у власній комерційній версії. Обидва ці видання можуть використовуватися для комерційного розвитку. На рисунку 3.1 зображено вікно IntelliJ IDEA.

Він надає поради щодо завершення коду, аналізу коду та надійних інструментів рефакторингу. Він має такі важливі інструменти, як система контролю версій, підтримка багатьох мов і фреймворків. Він здатний стежити за контекстом розробника та автоматично запускати відповідні інструменти.

У ньому наведено найбільш релевантні символи, які стосуються поточного контексту. Він постійно переміщує найновіші класи, методи і т.д. і т.п. на початок списку пропозицій. Таким чином завершення коду відбувається швидше. IntelliJ має можливість аналізувати потік даних і вгадувати можливий символ під час виконання.

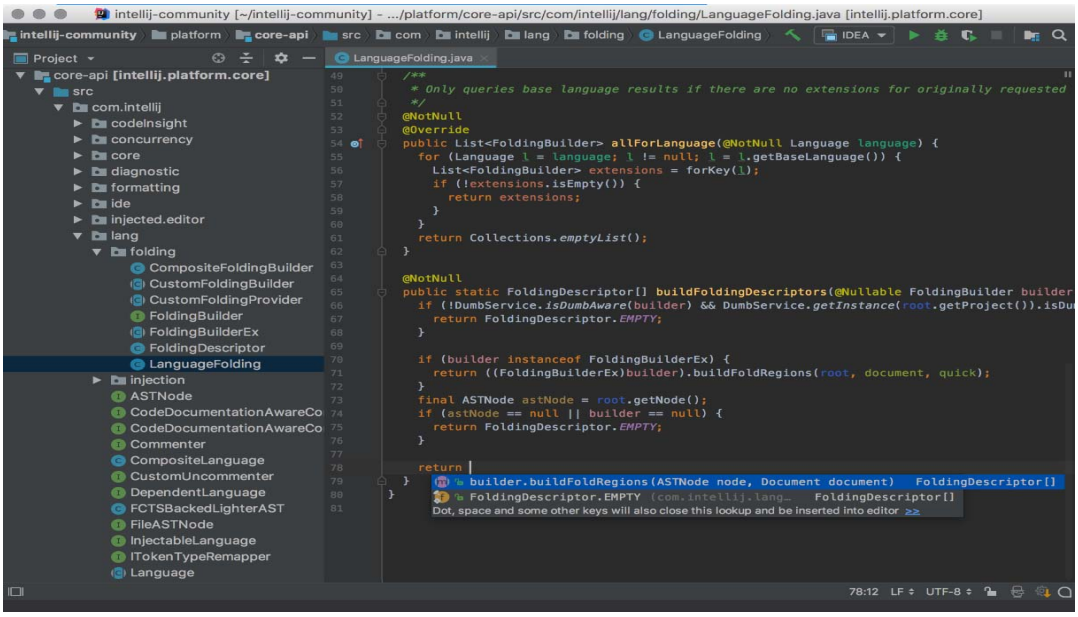

Рис 3.1. Вікно IntelliJ IDEA

Ви можете легко включити фрагменти іншої мови, наприклад SQL, у код Java. IntelliJ пропонує ретельний та ефективний рефакторинг, оскільки він знає все про використання символів. IntelliJ Idea постачається з великою різноманітністю вбудованих інструментів, таких як GIT, контроль версій, декомпілятор, покриття, база даних SQL тощо.

Він має потужний компілятор, який здатний виявляти дублікати, запахи коду, тощо. Він добре інтегрований із серверами додатків.

Плюси:

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

 має потужну функцію налаштування для зміни структури проєкту відповідно до вимог користувача;

 хороший інтерфейс з великою кількістю опцій теми.

Мінуси:

 потребує вдосконалення документація щодо інструментів;

 висока вартість корпоративної версії, а деколи IDE виходить з ладу, якщо це величезна програма[15].

Eclipse — це потужна повнофункціональна IDE із відкритим вихідним кодом, яка широко використовується для розробки програм Java. Eclipse поставляється

з базовим робочим середовищем і розширюваною системою плагінів, за допомогою яких ми можемо налаштувати середовище. Здебільшого він написаний мовою Java. На малюнку 3.2 показано вікно Eclipse.

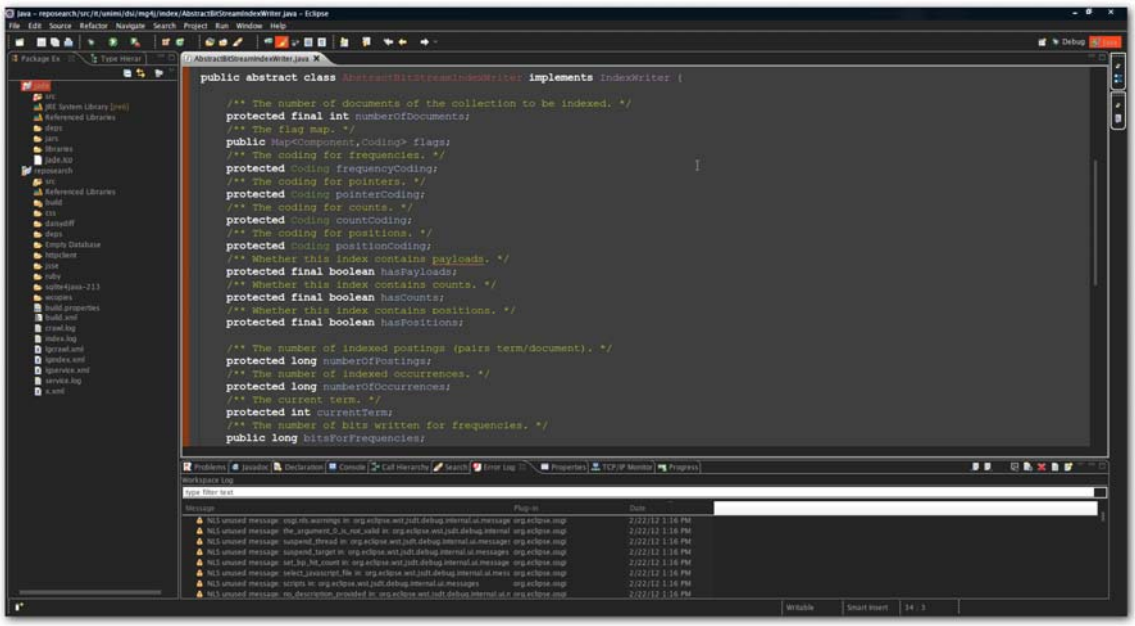

Рис. 3.2. Вікно Eclipse

Оскільки він відкритий, то допомагає розробникам здійснювати різні налаштування та зробити додаток більш надійним. Eclipse базується на основі Java, і тим самим робиться дуже розширюваним, гнучким і сумісним з багатьма мовами, такими як C ++, Groovy, Python, Perl, C # тощо. Цим самим він вважається найкращим вибором для розробників.

Eclipse є крос-платформним і працює на різних операційних системах, таких як Linux, Mac OS та Windows, здійснює підтримку розширення інструментів, редагування, перегляд, рефакторинг та налагодження. Всі ці функції полегшують програмістам розробку додатків.

Eclipse здійснює як локальне, так і віддалене налагодження, допускаючи, що використовується JVM, який підтримує віддалену налагодження. Крім цього Eclipse має велику допомогу та документацію.

Eclipse має власний торговий майданчик, який дозволяє користувачеві завантажувати клієнтські рішення. Він має хороший робочий простір, який

дозволяє розробникам легко ідентифікувати проєкти, папки та файли. Він має сильні рекомендації та функцію налагодження помилок.

Він дозволяє інтегруватися з сервером Apache Maven і контролем версій Git. Це стандартний віджет із підтримкою Gradle. Eclipse має хорошу інтеграцію для створення таких інструментів, як ANT і Maven.

Користувачі можуть розробляти різні додатки на одній платформі, наприклад, веб- та автономні додатки, веб-сервіси тощо. У Eclipse вбудовано надійні рекомендації щодо коду та відладчики.

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, яке підтримується Apache Software Foundation. Корисно для розробки веб-програм, настільних комп'ютерів, мобільних пристроїв, C++, HTML 5 тощо. NetBeans дозволяє розробляти програми з набору модульних програмних компонентів, які називаються модулями. NetBeans працює на Windows, Mac OS, Linux і Solaris.

Він поставляється з гарною архітектурою та вбудованими інструментами, які додають цінність повному SDLC від вимог до проєкту до розгортання. Він має активну спільноту користувачів і розробників по всьому світу. Він містить різні модулі, за допомогою яких добре виконуються функції. Він пропонує плавне та швидке редагування коду.

NetBeans — це мовний редактор, тобто він знаходить помилки, при , і допомагає зі спливаючими вікнами документації та інтелектуальним доповненням коду. Інструмент рефакторингу NetBeans дозволяє програмісту рефакторингувати код, не порушуючи його.

NetBeans також виконує аналіз вихідного коду та надає широкий спектр підказок щодо покращення коду або швидкого його виправлення. Він включає в

себе інструмент розробки GUI Swing, раніше відомий як «Проект Matisse». На рисунку 3.3 показано вікно NetBeans.

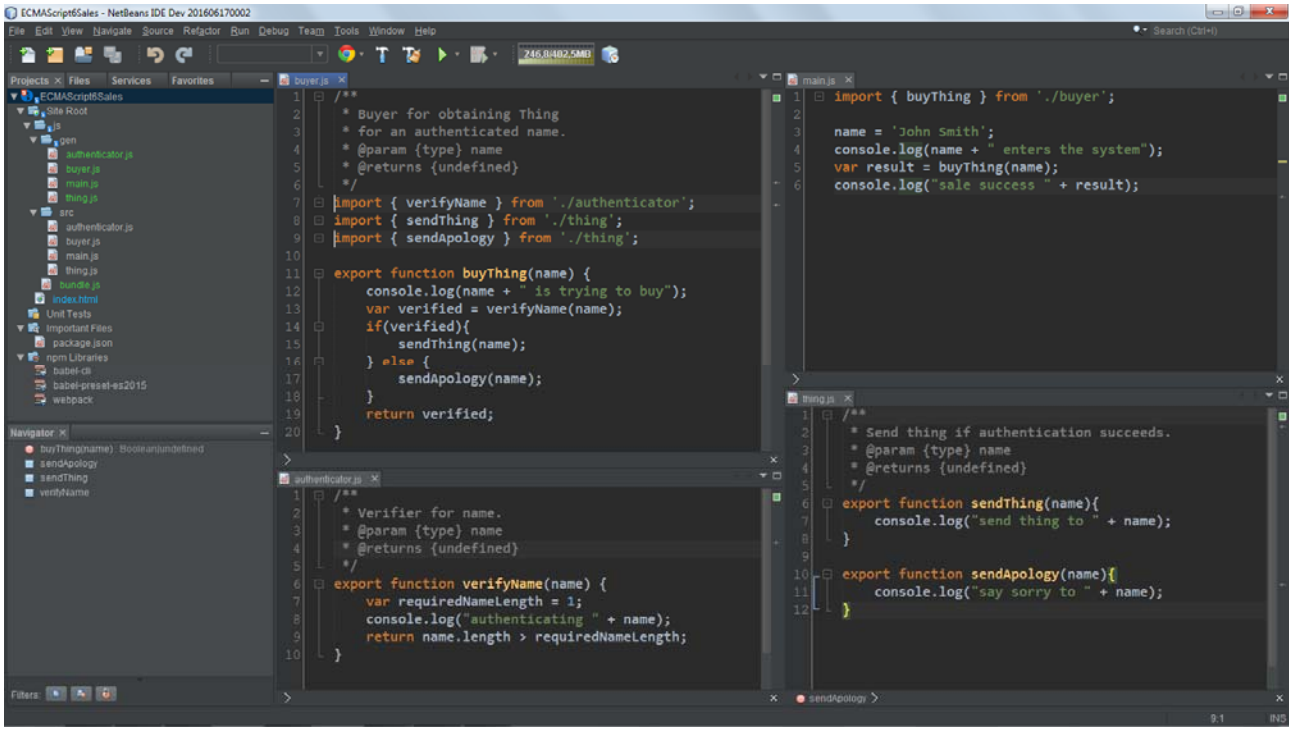

Рис. 3.3. Вікно NetBeans

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. Особливості програмної реалізація

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 3.4.

```
@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;
```

Рис.3.4. 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 3.5.

```
@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<>();
```

Рис.3.5. Програмна реалізація сутності User

3) Genre – відповідає за збереження інформації про жанр, його назву та інші атрибути. . Програмну реалізацію наведено на рисунку 3.6.

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

    private String name;

    private String title;

    private String externalId;

    public Genre() {
    }
```

Рис.3.6. Програмна реалізація сутності Genre

4) Role – відповідає за збереження інформації про роль користувача в системі. Програмну реалізацію наведено на рисунку 3.7.

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

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

Рис.3.7. 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 3.8.

```
@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) {...}

}
```

Рис.3.8. 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 3.9.

```
@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) {...}

}
```

Рис.3.9. 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) {...}
}
```

Рис.3.10. 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) {...}
}
```

Рис.3.10. 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 3.11.

```
@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) {...}
}
```

Рис.3.11. 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 3.12.

```
@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"; }
}
```

Рис.3.12. 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 3.13.

```
@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() {...}

}
```

```
@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 {...}
}
```

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

Рис.3.13. 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 3.14 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>
```

Рис.3.14. 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 3.15.

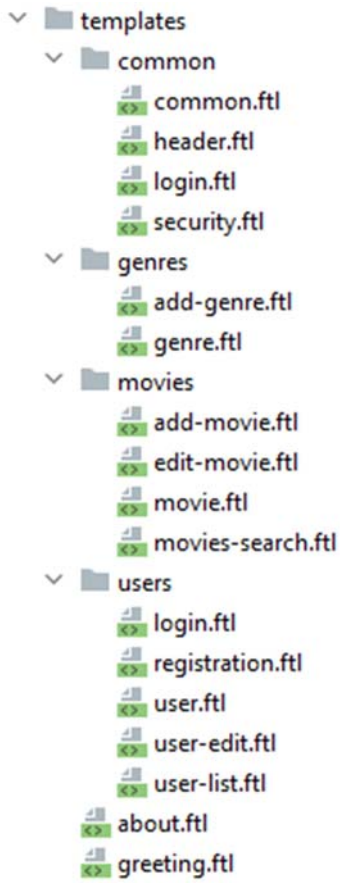

Рис.3.15. Файлова структура компонентів

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 3.16.

```
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
```

Рис.3.16. 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 3.17.

```
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', '');
        });
    });
}
```

Рис.3.17. Фрагмент програмної реалізації перегляду трейлеру

Одним з головних частин коду є (рис 3.18) – 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";
    }

}
```

Рис.3.18. 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 3.19.

```
@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";
}
```

Рис.3.19. Фрагмент коду пошуку за фільтрами

3.4. Тестування програмного забезпечення

Тестування будь якого програмного забезпечення — загальне поняття, яке 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].

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 3.20). Зображено вітання до користувача та описано суть сервісу.

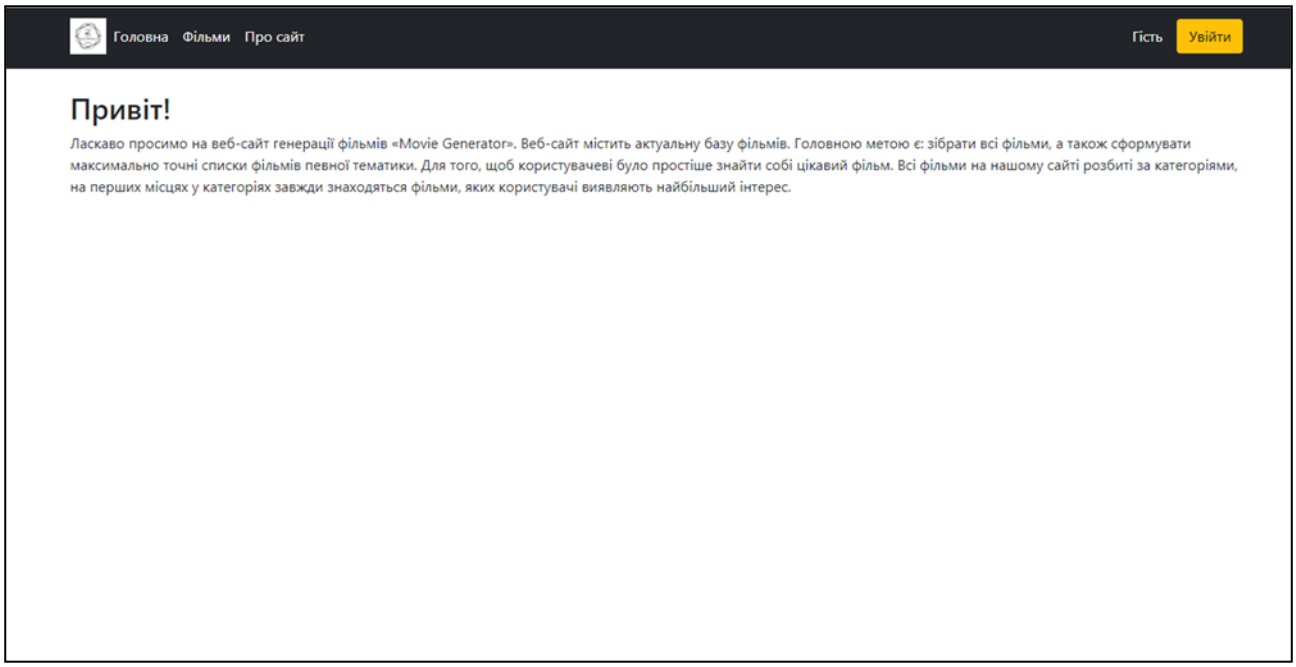

Рис.3.20. Скріншот інтерфейсу головної сторінки

Відкривши «Фільми», отримуємо ось таку сторінку(рис. 3.21), це основна сторінка для пошуку фільмів. Маємо ось такі фільтра: «Назва», «Рік», «Рейтинг».

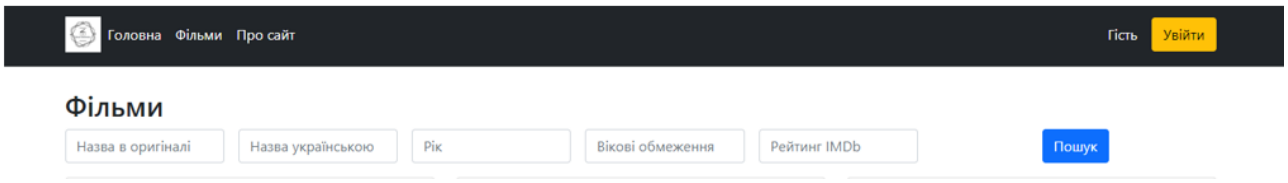

Рис.3.21. Скріншот функціоналу сторінки «Фільми»

Заповнивши поля, отримуємо декілька відповідних фільмів(рис. 3.22), які виводяться на цій же сторінці, з короткою але зрозумілою інформацією.

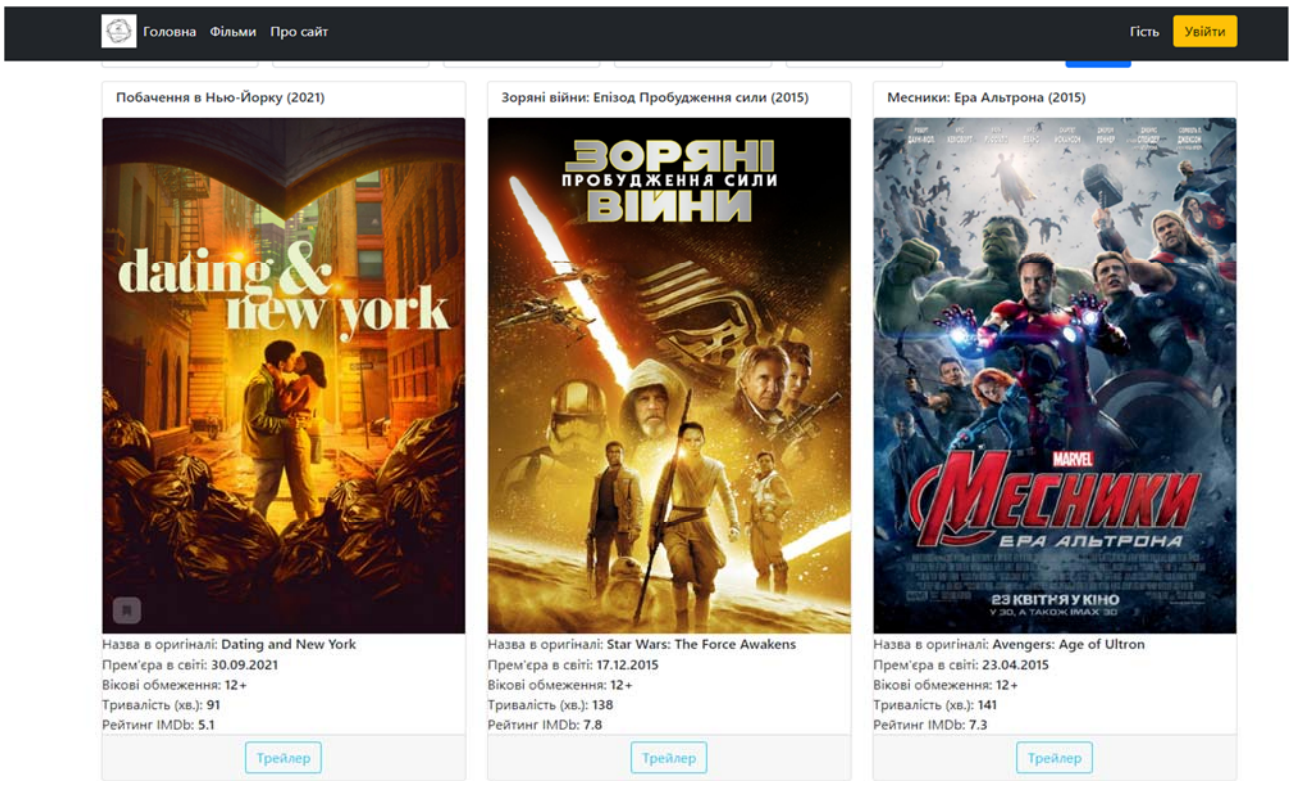

Рис.3.22. Скріншот результату пошуку

Про кожний фільм, є коротка інформація, є афіша, та є можливість подивитись трейлер до фільму(рис 3.23).

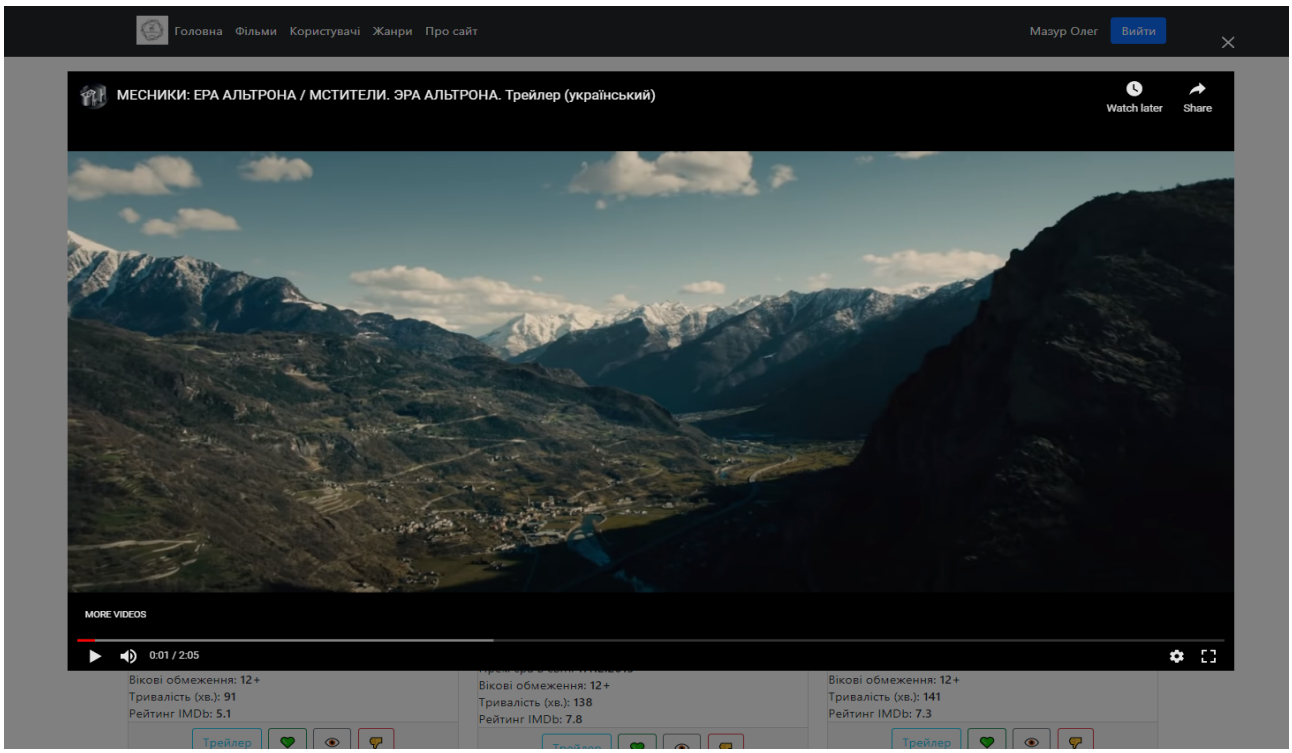

Рис.3.23. Скріншот виводу трейлеру фільму

Також є кнопка «Про сайт», де знаходиться інформація про веб сайт, також правила користування та описана суть реєстрації(рис. 3.24).

Рис.3.24. Скріншот сторінки «Про сайт»

Справа зверху є кнопка «Увійти», тобто є функція реєстрації. Натиснувши її, отримуємо сторінку зображену на рисунку 3.25. Ми бачимо що можна увести логін та пароль, або зареєструватись.

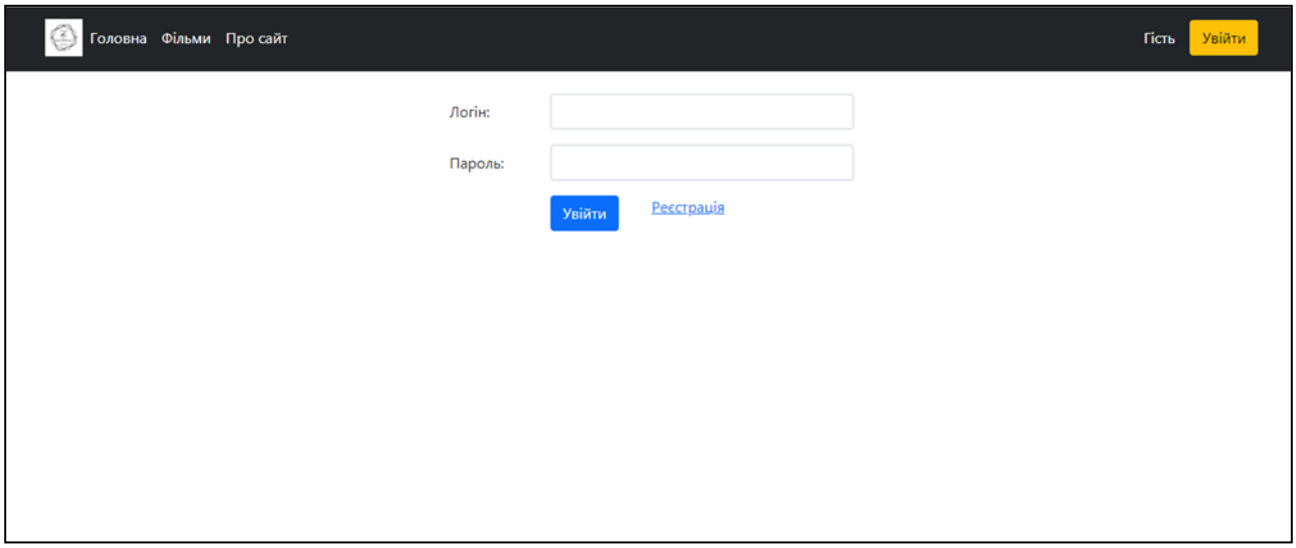

Рис.3.25. Скріншот сторінки входу

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 3.26).

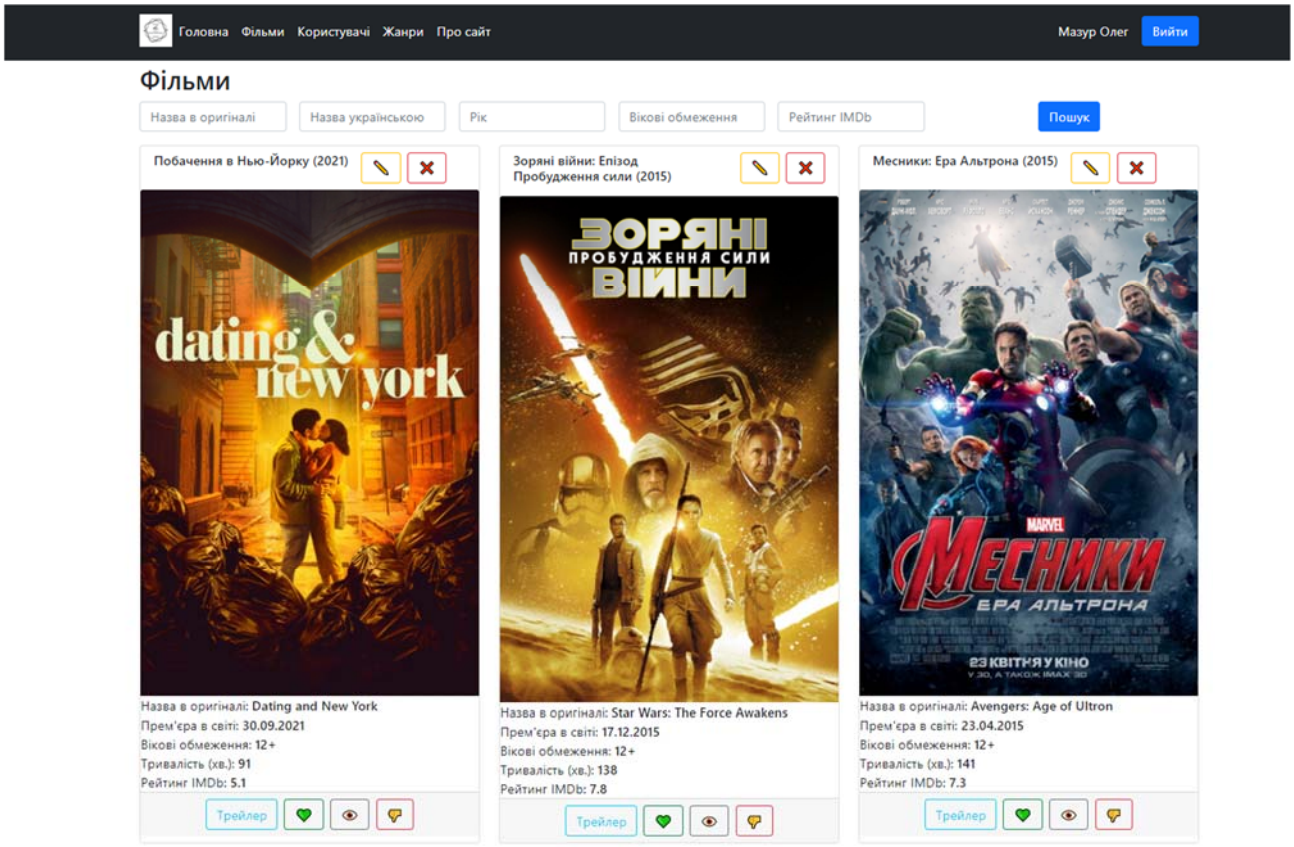

Рис.3.26. Взаємодія користувача з фільмами

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. 3.27).

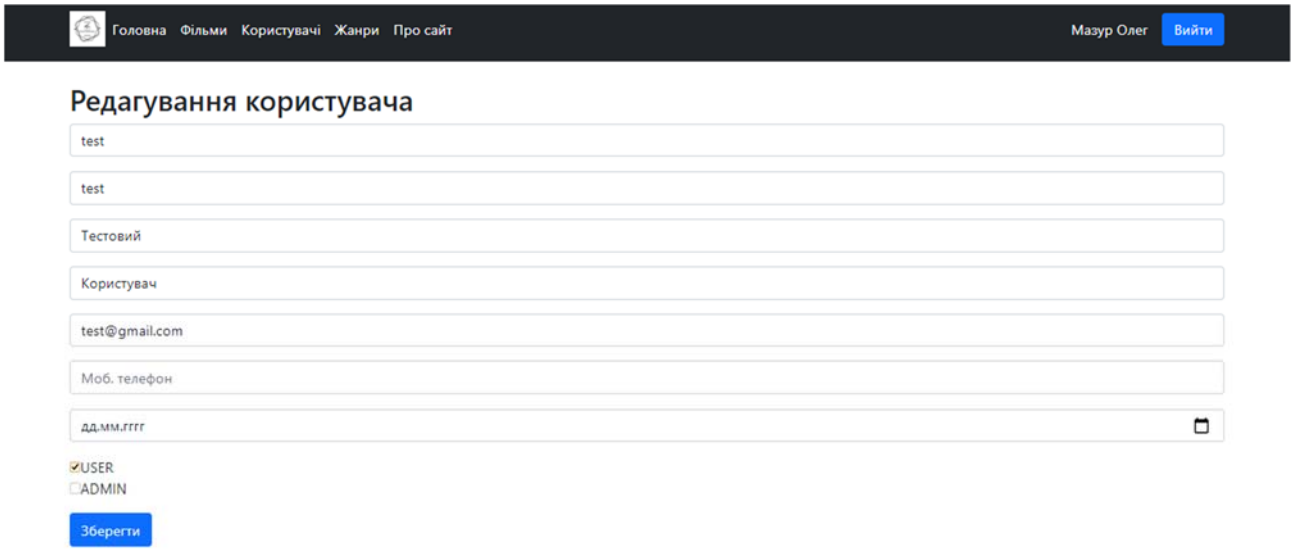

Рис.3.27. Можливість користувача редагування даних

А в адміністратора в свою чергу є можливість перегляду інформації про користувачів, щоб контролювати порядок та ситуацію(рис. 3.28).

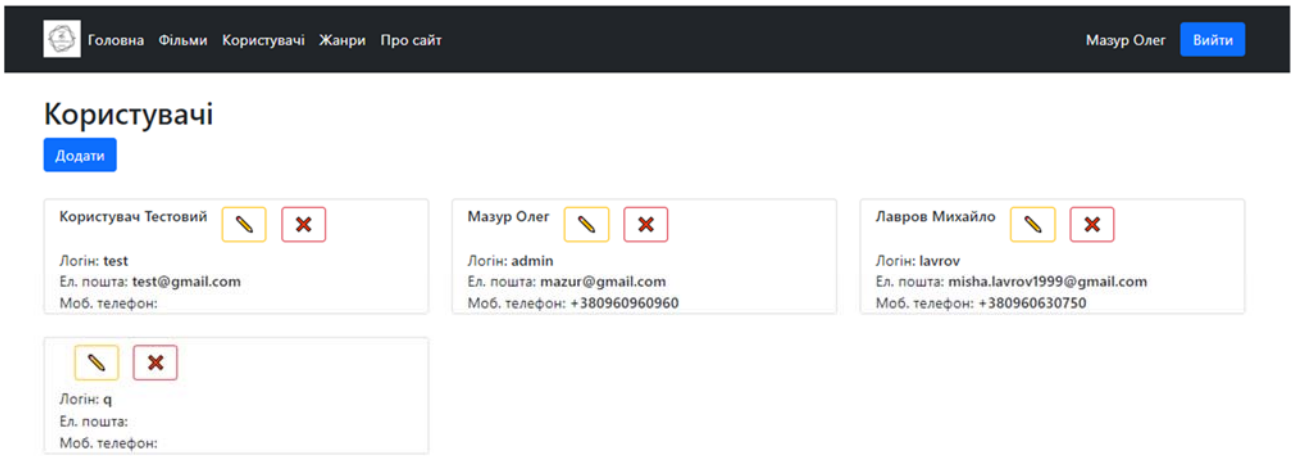

Рис.3.28. Перегляд користувачів адміністратором

Також в адміністратора є можливість моніторингу та редагування інформації про кінофільми. Цю сторінку зображено на рисунку 3.29.

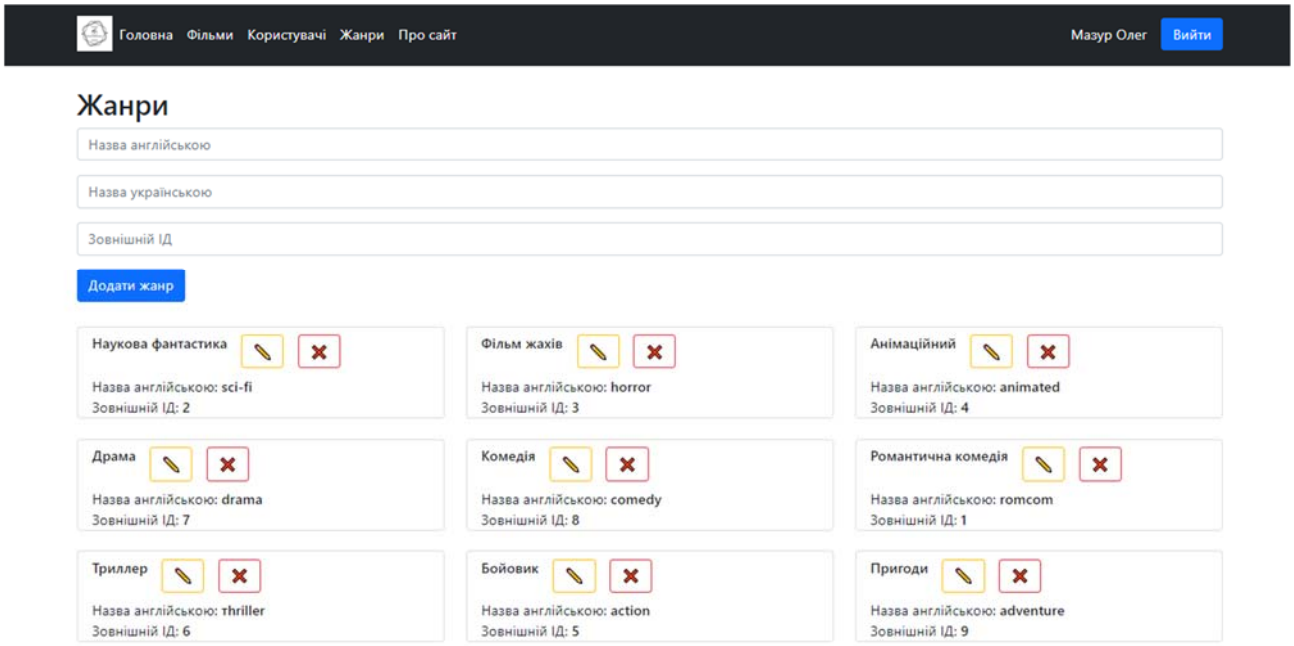

Рис.3.29. Моніторинг та редагування адміністратором кінофільмів

Висновки до розділу III

У цьому розділі 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.

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.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

У кваліфікаційній роботі було розроблено програмне забезпечення для 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 кваліфікаційної роботи.

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

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:

- проаналізовано предметну область;
- розроблено методику генерації відеоконтенту;
- розроблено та впроваджено метод для генерегуван відеоконтенту на базі плівкової фільтрації;
- розроблено програмне забезпечення для генерування відеоконтенту в реальному масштабі часу;
- проведено тестування програмне забезпечення для генерування відеоконтенту в реальному масштабі часу.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кінематограф в житті сучасної людини [Електронний ресурс] – Режим доступу: https://vbusk.com/cikavo/kinematograf-zhizni-sovremennogo-cheloveka.html

2. Кіно [Електронний ресурс] – Режим доступу: https://om-saratov.ru/blogi/13-January-2016-i32652-osnovnaya-polza-ot-prosmotr

3. Кінотерапія [Електронний ресурс] – Режим доступу:https://blog.rt.ru/b2c/kinoterapiya-kak-smotret-filmy-s-polzoi-dlya-zdorovya.htm

4. Кінематограф [Електронний ресурс] – Режим доступу:https://pravlife.org/ru/content/kinematograf-v-zhizni-sovremennogo-cheloveka

5. Генератор фільмів [Електронний ресурс] – Режим доступу: http://free-generator.ru/films.html

6. Генератор онлайн [Електронний ресурс] – Режим доступу: https://generator-online.com/uk/about/

7. Генератор випадкових чисел [Електронний ресурс] – Режим доступу:https://uk.education-wiki.com/7009496-random-number-generator-in-java

8. 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

9. Генератор псевдо випадкових чисел [Електронний ресурс] – Режим доступу:https://conf.ztu.edu.ua/wp-content/uploads/2017/06/123-1.pdf

10. Функціональна блок-схема [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Functional_block_diagram

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. Соммервілла, І. Інженерія програмного забезпечення / пер. з англ. А.А. Мінько, А.А. Момотюк, Г.І. Сингаївська. - М .: Вільямс, 2002. - 624 с., Іл.

ДОДАТОК А
ДЕКЛАРАЦІЯ ДОБРОЧЕСНОСТІ

Я, Петрунів Олександра Михайлівна, підтверджую, що сам написала цю роботу і не використовував жодних інших, окрім цитованих, джерел інформації.

Дослівні вирази або фрази, які цитуються, позначаються як такі; інші недослівні запозичення чи ремінісценції, наведені в тексті цієї роботи, містять актуальну інформацію щодо первинних джерел наведеного контенту. Робота у цій безпосередньо або змістовно аналогічній формі не була раніше публікована чи оприлюднена. Усе зазначене вище посвідчую власноручним підписом.

(підпис) (дата)

ДОДАТОК Б

ЛІСТИНГ ОСНОВНИХ МОДУЛІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

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.AuthenticationManagerBuil
der;
import
org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurit
y;
```

```
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
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 В
КОПІЯ ПУБЛІКАЦІЇ