Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ КЛАСИФІКАЦІЇ ОБ'ЄКТІВ НА ОСНОВІ МАШИННОГО НАВЧАННЯ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-18-1
Ходонович А.Б
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Сакало Є.С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Ходоновичу Андрію Борисовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка мобільного додатку для класифікації об'єктів на основі машинного навчання

затверджена наказом університету від 16 травня 2022 року No 541Ст

2. Термін подання студентом роботи до екзаменаційної комісії 28 травня 2022 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, дані інтернет-мережі, фреймворк Laravel мови програмування PHP, середовище розробки PhpStorm

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд фреймворків для розробки вебсайтів
2. Розробка бази даних
3. Розробка веб застосунку

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Схематичне зображення бази даних, візуалізація прототипу вебсайта, діаграми прецедентів.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Бєлова Н.В.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	18.04.2022	
2	Аналіз завдання, підбір літератури	18.04.22-20.04.22	
3	Аналіз літератури з досліджуваної проблеми	20.04.22-27.04.22	
4	Аналіз методів розробки вебсайтів за допомогою фреймворків	27.04.22-30.04.22	
5	Розробка моделі вебсайту	31.04.22-05.05.22	
6	Реалізація вебсайту	06.05.22-22.05.22	
7	Оформлення пояснювальної записки	22.05.22-28.05.22	
8	Перевірка на плагіат	28.05.22	
9	Рецензування	29.05.22	
10	Підготовка презентації та доповіді	29.05.22-30.05.22	
11	Занесення роботи в електронний архів	31.05.22	
12	Попередній захист кваліфікаційної роботи	01.06.22	

Дата видачі завдання 18 квітня 2022 р.

Студент _____

(підпис)

Керівник роботи _____ _____доц. Сакало Є.С._____

(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи: 44 с., 9 табл., 23 рис., 18 джерел.

WEB-ЗАСТОСУНОК, FRAMEWORK LARAVEL, MVC, БАЗА ДАННИХ, MIX LARAVEL.

Об’єктом роботи є розробка вебсайту для перегляду фільмів, серіалів та інше.

Метою роботи було створення сайту за допомогою мови програмування PHP та фреймворка Laravel.

У роботі були розглянуті основи побудови інтернет-сайтів за допомогою концепції MVC. Були досліджені особливості побудови вебсайту за допомогою фреймворку Laravel. В результаті виконання роботи був створений сайт з можливістю реєстрація на сайті, перегляду фільмів, додавання вподобаних в улюблене, можливість залишити коментар під фільмом.

WEB APPLICATION, FRAMEWORK LARAVEL, MVC, DATABASE, MIX LARAVEL.

The object of the work is the development of a website for watching movies, TV shows, and so on.

The goal of the work was to create a website using the PHP programming language and the Laravel framework.

The paper discusses the basics of building internet sites using the MVC concept. The features of building a website using the Laravel framework were investigated. As a result of completing the work, a website was created with the ability to register on the site, watch movies, add your favorite ones to your favorite ones, and leave a comment under the movie.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

HTML – HyperText Markup Language (мова розмітки гіпертексту)

HTTP – HyperText Transfer Protocol

CRUD – create, read, update, delete

БД – база даних

ВСТУП

Internet розвивається досить стрімко. Швидко зростає кількість видань, присвячених мережі, що віщує широке її поширення навіть в далеких від техніки областях. Internet перетворюється з великої іграшки для інтелектуалів в повноцінне джерело різноманітної корисної інформації для будь-якої категорії користувачів.

За статистикою на 2021 рік число користувачів Інтернету склало 5 мільярдів осіб та за прогнозами буде постійно зростати. Тому для того, щоб встигати за швидкістю розвитку нових технологій, у вебпрограмістів часто недостатньо часу для реалізації повсякденних завдань з нуля, таких як: аутентифікація, API, тестування, дебагінг, кешування, робота з БД і т.д. саме для спрощення і швидкого вирішення цих завдань були придумані вебфреймворки. Вебфреймворк – це каркас програми, набір інструментів, стандартів і, в певній мірі, робочий простір.

Також, через те, що користувачів Інтернету з кожним роком стає все більше, ще однією гострою проблемою останніх років стає оптимізація сайтів під велику кількість запитів. Тому створення вебсайту, який передбачає роботу з великими навантаженнями є однією з головних задач веб програміста.

1 ТЕОРЕТИЧНІ АСПЕКТИ ДОСЛІДЖУВАНОЇ РОБОТИ

1.1 Використовувані технології

Для створення сайту використовувалися такі технології, як: мова програмування PHP і його фреймворк Laravel, а також мова запитів MySQL для створення бази даних. Зупинимося на PHP детальніше.

1.1.1 PHP

Препроцесор гіпертексту або ж PHP, є широко використовуваним мовою сценаріїв загального призначення з відкритим вихідним кодом. PHP створювався спеціально для ведення Web-розробок і може використовуватися безпосередньо в HTML-коді.

На сьогоднішній день PHP є найбільш поширеною мовою вебпрограмування. Переважна маса сайтів і вебсервісів в Інтернеті написано за допомогою PHP. За деякими оцінками, PHP застосовується більш ніж на 80% сайтів, серед яких такі сервіси, як Facebook, Wikipedia, Yahoo, VK, та інші.

PHP був створений в 1994 році данським програмістом Расмусом Лердорфом і спочатку представляв собою набір скриптів на мові Perl.

Пізніше цей набір скриптів був переписаний в інтерпретатор на мові C.і з самого виникнення PHP являв собою зручний набір інструментів для спрощеного створення вебсайтів і вебзастосунків.

Особливістю PHP вважається те, що він є серверною мовою і тому вся обробка скриптів на цій мові проводиться на стороні сервера. У момент запиту браузера користувача на скачування сторінки, сервер отримує команду на обробку PHP сценаріїв. Після обробки сервер віддає на виході заново побудовану HTML сторінку без будь-яких натяків на код PHP.

1.1.2 MySQL

MySQL представляє систему управління реляційними базами даних (СУБД). Сьогодні це одна з найпопулярніших систем управління базами даних.

Початковим розробником даної СУБД була шведська компанія MySQL AB. У 1995 році вона випустила перший реліз MySQL. У 2008 році компанія MySQL AB була придбана компанією Sun Microsystems, а у 2010 році вже компанія Oracle поглинула Sun і тим самим отримавши права на торгову марку MySQL. Тому MySQL нині розвивається під егідою Oracle. MySQL має кросплатформенність, є дистрибутиви під найрізноманітніші ОС, в тому числі найбільш популярні версії Linux, Windows, MacOS.

Переваги MySQL:

– відкритий вихідний код. Поширюється безкоштовно для домашнього застосування;

– простота. MySQL легко встановлюється, має доступний інтерфейс, а різноманітність плагінів і додаткових додатків спрощує роботу з БД;

– функціонал. Містити практично весь необхідний набір інструментів, який може стати в нагоді при розробці будь-якого проекту;

– безпека. Багато систем безпеки вже вбудовані й працюють за замовчуванням;

– масштабованість. Може використовуватися в роботі як з малим, так і з великим обсягом даних;

– швидкість. Є однією з найшвидших серед наявних на сучасному ринку.

1.2 Вибір фреймворку

1.2.1 PHP фреймворки

Вебфреймворк – це платформа для створення сайтів і вебзастосунків, що полегшує розробку й об’єднання різних компонентів великого програмного проекту. За рахунок широких можливостей в реалізації бізнес-логіки та високої продуктивності ця платформа особливо добре підходить для створення складних сайтів, бізнес-додатків і вебсервісів.

Розглянемо декілька найбільш популярних фреймворків:

Laravel – це безкоштовний PHP фреймворк з відкритим вихідним кодом, створений Тейлором Отвеллом для розробки вебзастосунків за архітектурним шаблоном MVC. Laravel – це спроба об’єднати все найкраще, що є в інших PHP фреймворках, а також Ruby on Rails, ASP.NET MVC і Sinatra. Фреймворк має досить велику кількість плюсів. Серед них:

– досить непогана і зрозуміла документація;

– навколо фреймворку створена потужна екосистема. Різні курси, конференції, навчальні матеріали дозволяють зібрати навколо фреймворку велику кількість розробників й спонсорів, які зацікавлені в розвитку інструменту і беруть в цьому участь;

– одним з найбільш очевидних плюсів Laravel, є гнучка система маршрутизації, що дозволяє скласти найрізноманітніші перевірки маршруту вебзастосунку. Ви можете виділити маршрути в спеціальні групи, використовувати простір імен, вказати параметри маршруту, використовувати регулярні вирази, налаштувати піддомену маршрутизацію та багато іншого;

– фреймворк містити вбудовані механізми аутентифікації та авторизації Користувачів, яку можна переналаштувати під свої потреби;

– laravel містить зручний механізм обробки помилок і винятків;

– laravel надає з коробки механізми для кешування вебзастосунка за допомогою Memcached і Redis. Крім цього є зручні функції для використання простого файлового кешування даних;

– laravel надає чистий і простий API поверх популярної бібліотеки SwiftMailer з драйверами для SMTP, Mailgun, SparkPost, Amazon SES і sendmail, щоб зробити відправку пошти через локальну або хмарну службу за вибором. У тому числі є механізм для побудови черг відправки пошти.

Yii – це безкоштовний об’єктно-орієнтований компонентний full-stack PHP фреймворк. В основі Yii лежить інший фреймворк – PRADO, написаний на ASP. NET і згодом перенесений на PHP. Незабаром після побудови нової архітектури, фреймворк PRADO був перейменований на Yii. Назва фреймворку є абревіатурою слова «Yes It Is» Прабатьком фреймворку є китайський розробник Qiang Xue.

Yii можна використовувати для розробки будь-якого виду вебзастосунка. Завдяки своїй основі компонентів, архітектурі й складної підтримки кешування, фреймворк підходить для розробки великомасштабних проектів, таких як портали, форуми, системи управління контентом (CMS), систем електронної комерції, RESTful вебсервісів і т.д.

Як і багато інших PHP фреймворків, Yii використовує архітектурний патерн MVC, так само будучи full-stack фреймворком, надаючи безліч перевірених і готових до використання функцій: будівник запитів і ActiveRecord для реляційних і NoSQL баз даних, RESTful API, підтримку багаторівневого кешування і т.д.

До плюсів фреймворку можна віднести:

– yii сприяє швидкому прототипуванню вебпрограми. Він відноситься до інструментів RAID розробки;

– компонент програми i18n дозволяє здійснювати автоматичний переклад повідомлень вебзастосунки;

– вбудована підтримка автоматичної валідації форм і виведення повідомлень про помилки на основі даних з моделей вебзастосунки;

– хороша документація російською мовою;

– містить вбудовану і дуже зручну debug панель.

1.2.2 Порівняння фреймворків

Розглянемо основні вимоги до фреймворків. Вимоги мають на увазі необхідний набір інструментів і технологій для роботи фреймворку. Зазвичай вимоги у всіх сучасних PHP фреймворків однакові, але бувають і відмінності, адже вони реалізують в собі різні правила, патерни й технології. Список вимог по Yii та Laravel наведені в таблиці 1.1.

Таблиця 1.1 – Версії, які підтримують фреймворки

Вимоги	Yii 2	Laravel 9
Версія PHP	>=5.4.0	>=8.0

Як бачимо з таблиці Yii підтримує досить велику кількість версій PHP, на відміну від Laravel, який підтримує версію PHP тільки 8.0 або вище. Але так як наш вебсайт буде базуватися на PHP 8.1 то для нас не буде проблем у використанні Laravel.

Порівняємо архітектуру фреймворків. Обидва фреймворки для організації коду використовують архітектурний патерн MVC.

MVC розшифровується як модель-представлення-контролер (від англ. model-view-controller). Це спосіб організації коду, який передбачає виділення блоків, що відповідають за вирішення різних завдань. Один блок відповідає за дані програми, інший відповідає за зовнішній вигляд, а третій контролює роботу програми.

Компоненти MVC:

Модель – цей компонент відповідає за дані, а також визначає структуру програми. Наприклад, якщо ви створюєте To-Do додаток, код компонента model визначатиме список завдань і окремі завдання.

Представлення – цей компонент відповідає за взаємодію з користувачем. Тобто код компонента view визначає зовнішній вигляд програми та способи його використання.

Контролер – цей компонент відповідає за зв’язок між model і view. Код компонента controller визначає, як сайт реагує на дії користувача. По суті, це мозок MVC-додатку.

За архітектурою дані фреймворки ідентичні, бо реалізують один і той же архітектурний патерн. Він дозволяє швидко реалізувати проект, а також легко супроводжувати й масштабувати його надалі

Порівняємо повноту документації, надану розробниками по їх продуктам. Важливим фактором при виборі фреймворку є його документація. Через багато років у Laravel з’явилася досить потужна документація. Деякі моменти написані не найпростішою мовою, але в загальному і цілому документація досить зрозуміла. У Yii навпаки ж через стільки років розробки поступилася документації Laravel, але вона все ще добре написана, і буде зрозуміла навіть новачкам в PHP.

Підтримка спільнотою також важлива, тому розглянемо рівень підтримки у обох фреймворків. Саме спільнота пише розширення, допомагає розробникам розвивати свій фреймворк і усувати баги. Так даному етапі варто враховувати не тільки співтовариство в цілому, але і мовний сегмент. Останні кілька років Laravel став найпопулярнішим PHP фреймворком по всьому світу, за винятком Західної Європи і Росії. Але в Україні та Росії Yii залишається на першому місці, внаслідок чого знайти навчальний матеріал або документацію набагато легше.

Порівняємо підтримку ORM. ORM – технологія програмування, яка пов’язує бази даних з концепціями об’єктно-орієнтованих мов програмування, створюючи «віртуальну об’єктну базу даних». Існують як пропрієтарні, які є приватною власністю авторів або правовласників і не задовольняють критеріям вільного ПЗ (наявності відкритого програмного коду недостатньо), так і вільні реалізації цієї технології. У таблиці 1.2 наведені підтримувані та реалізовані популярні ORM:

Таблиця 1.2 – Підтримка ORM

ORM	Yii 2	Laravel 9
DAO	Йде разом з фреймворком	Йде разом з фреймворком
Active Record Pattern	Active Record	Eloquent ORM
Doctrine 2	Через плагіни	Через плагіни

Active Record – (AR) – шаблон проектування додатків. AR є популярним способом доступу до даних реляційних баз даних в ООП.

Data Mapper – (DM) – це рівень доступу до даних, який виконує двонаправлену передачу даних між постійним сховищем даних (часто реляційною базою даних) і поданням даних в пам’яті (рівень домен). Мета шаблону – зберегти представлення в пам’яті і постійне сховище даних незалежно один від одного і самого модуля відображення даних. Рівень складається з одного або декількох картографів (або об’єктів доступу даних), що виконують передачу даних.

Вибір конкретного підходу Active Record vs Data Mappers + Repository повинен випливати з бізнес завдань.

Справа в тому, що Active Record є фактичний невід’ємною частиною RAD, і мета його – швидко розгорнути рівень доменних моделей (читай бізнес логіки) за існуючою схемою бази (як правило, невеликий). Фактично всі RAD проекти будуються за принципом «Database First», тобто спочатку проектується схема БД, потім вона декларується (Json, Yaml, SQL) і через цю декларацію фреймворк генерує доменні моделі з уже реалізованим CRUD функціоналом через механізм scaffolding.

Це добре працює на маленьких і середніх проектах з невеликими залежностями. Виникають проблеми в разі, якщо проект великий і людей, які працюють на ньому, багато ,і, щоб підтримувати все в робочому стані, люди хочуть покривати код тестами.

Безпека – важлива частина в сучасних додатках, тому зупинимося на ній докладно.

Будь-який продукт повинен захищати дані своїх користувачів і забезпечувати належне управління рівнями доступу для різних ролей користувачів. Надійний захист даних – один з найважливіших критеріїв лояльності клієнтів. Ось чому при виборі конкретного фреймворку для розробки вебзастосунку завжди потрібно враховувати, чи задовольняють його можливості вашим вимогам безпеки. І Yii, і Laravel надають механізми для захисту паролем, аутентифікації, захисту від SQL-ін’єкцій, міжсайтового скриптингу та інших загроз безпеці.

Крім того, Yii пропонує багатофункціональну систему управління доступом на основі ролей. У Laravel за замовчуванням немає вбудованих інструментів контролю доступу, але це можна вирішити за допомогою великої кількості плагінів.

1.3 Постановка задачі

Об’єктом роботи є розробка вебсайту для перегляду фільмів, серіалів та інше.

Метою роботи є створення сайту за допомогою мови програмування PHP та фреймворка Laravel.

Нижче наведені опис предметної області, вимоги користувача до даних і функціоналу інформаційної системи:

- застосунок повинен давати можливість користувачам переглянути будь-який фільм на, який наданий на веб сайті;
- користувач сайту повинен мати можливість:
  - зареєструватися на сайті;
  - переглянути будь-який фільм, який він самостійно вибере;
  - залишити коментар під будь-яким фільмом;
  - додати вподобаний фільм в список улюблених.

– адміністратор сайту повинен мати можливість додавати, редагувати і видаляти будь-які записи в будь-яких таблицях бази даних.

Для досягнення мети необхідно вирішити наступні задачі:

– вивчити основи роботи з фреймворками мови програмування PHP теорію проектування реляційних БД, основні принципи нормалізації;

– побудувати діаграму прецедентів;

– побудувати логічну модель БД;

– сформувати базу даних, використавши СУБД MySQL;

– розробити дизайн веб сайту;

– написати frontend і backend веб застосунку;

– провести тестування.

Для розробки застосунку будуть використані такі інструменти, як:

– MySQL – СУБД;

– PhpStorm – середа розробки;

– PHP – мова програмування;

– Laravel – фреймворк.

2 ПРОЕКТУВАННЯ ВЕБСАЙТУ

Поділимо проектування сайту на кілька етапів:

– перший етап включає в себе побудову діаграми прецедентів;

– другий етап включає в себе побудову бази даних.

2.1 Діаграма прецедентів

Дана діаграма являє собою схему взаємодії користувача з системою, яка показує зв’язок між Користувачем і різними випадками використання, в якому користувач бере участь. Діаграма варіантів використання може ідентифікувати різні типи користувачів системи й різні варіанти використання та часто може супроводжуватися іншими типами діаграм.

Загальні компоненти:

– актори: користувачі, які взаємодіють з системою. Актор може бути людиною, організацією або зовнішньою системою, що взаємодіє з додатком або системою. Вони повинні бути зовнішніми об’єктами, які виробляють або споживають дані;

– прецедент: конкретна послідовність дій і взаємодії між учасниками та системою. Прецедент також можна назвати сценарієм;

– цілі: кінцевий результат більшості випадків використання. Успішна діаграма повинна описувати дії та варіанти, які використовуються для досягнення мети.

На діаграмах UML для зв’язування елементів використовуються різні сполучні лінії, які називаються відносинами. Кожне таке відношення має власну назву і використовується для досягнення певної мети:

– відношення асоціації – служить для позначення специфічної ролі актора при його взаємодії з окремим варіантом використання;

– відношення включення між двома варіантами використання – вказує на те, що задана поведінка для одного варіанту використання включається в якості складеного фрагмента в послідовність поведінки іншого варіанту використання;

– відношення розширення – визначає взаємозв’язок базового варіанту використання з іншим варіантом використання, функціональна поведінка якого використовується не завжди, а тільки при виконанні додаткових умов;

– відношення Узагальнення застосовується в тому випадку, коли необхідно відзначити, що дочірні варіанти використання володіють усіма особливостями поведінки батьківських варіантів.

При побудові діаграми варіантів використання хотілося показати, які основні дії зможе здійснювати користувач, такі як: перегляд зацікавленого контенту, перегляд свого профілю, додавання в вподобане, залишити коментар, можливість змінювати інформацію про себе в профілі. Всі ці дії будуть реалізовані веб сервісом. Діаграма варіантів використання з боку Користувача представлена на рисунку 2.1.

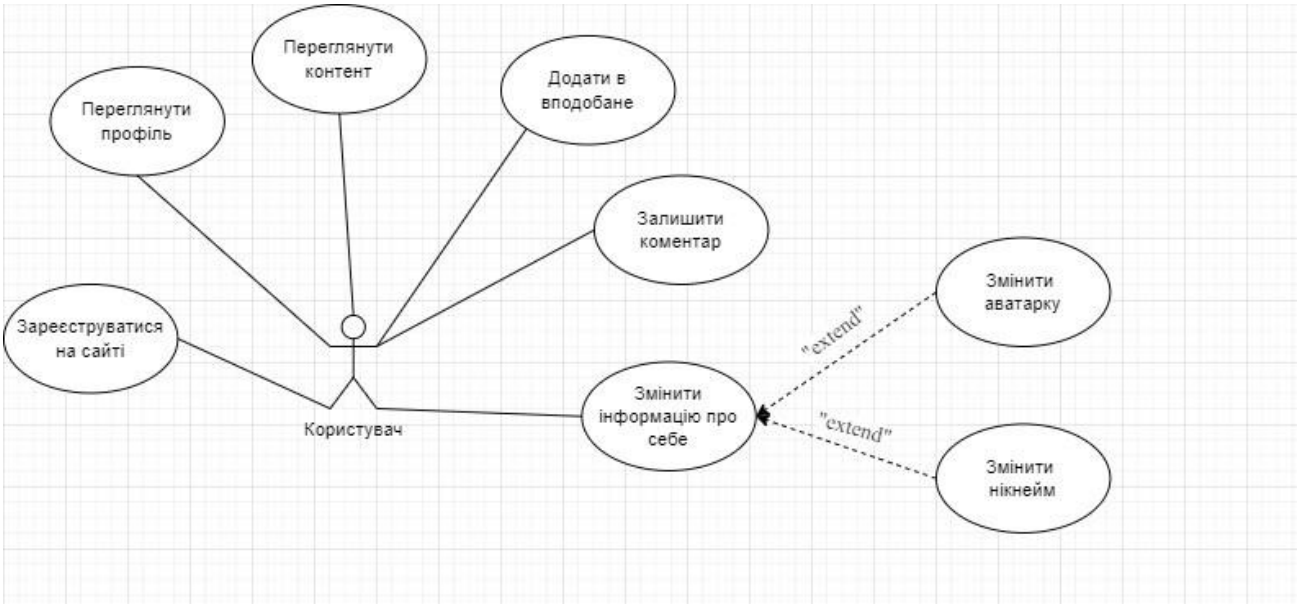

Рисунок 2.1 – Діаграма прецедентів користувача

Конкретизація варіантів використання:

Зареєструватися на сайті:

– основна дійова особа: користувач;
– інші учасники прецеденту: відсутні;
– зв’язки з іншими варіантами використання: відсутні.

Короткий опис: даний варіант використання дозволяє користувачеві зареєструватися на сайті.

Переглянути профіль:

– основна дійова особа: користувач;
– інші учасники прецеденту: відсутні;
– зв’язки з іншими варіантами використання: відсутні.

Короткий опис: даний варіант використання дозволяє користувачеві переглянути свій профіль, в якому можна переглянути свій список вподобаного, а так само змінити ім’я та аватар.

Переглянути контент:

– основна дійова особа: користувач;
– інші учасники прецеденту: відсутні;
– зв’язки з іншими варіантами використання: відсутні.

Короткий опис: даний варіант використання дозволяє користувачеві переглянути вподобаний контент на сайті, такий як фільми, серіали, тощо.

Додати в вподобане:

– основна дійова особа: користувач;
– інші учасники прецеденту: відсутні;
– зв’язки з іншими варіантами використання: відсутні.

Короткий опис: даний варіант використання дозволяє користувачеві добавити у список контент, який йому сподобався, для подальшого ведення статистики

Залишити коментар:

– основна дійова особа: користувач;

– інші учасники прецеденту: відсутні;

– зв’язки з іншими варіантами використання: відсутні.

Короткий опис: даний варіант використання дозволяє користувачеві залишити коментар під любим контентом на сайті.

Змінити інформацію про себе:

– основна дійова особа: користувач;

– інші учасники прецеденту: відсутні;

– зв’язки з іншими варіантами використання: присутні.

Короткий опис: даний варіант використання дозволяє користувачеві змінити у своєму профілі свій нікнейм на сайті, а також змінити аватарку.

Діаграма варіантів використання з боку адміністратора представлена на рисунку 2.2.

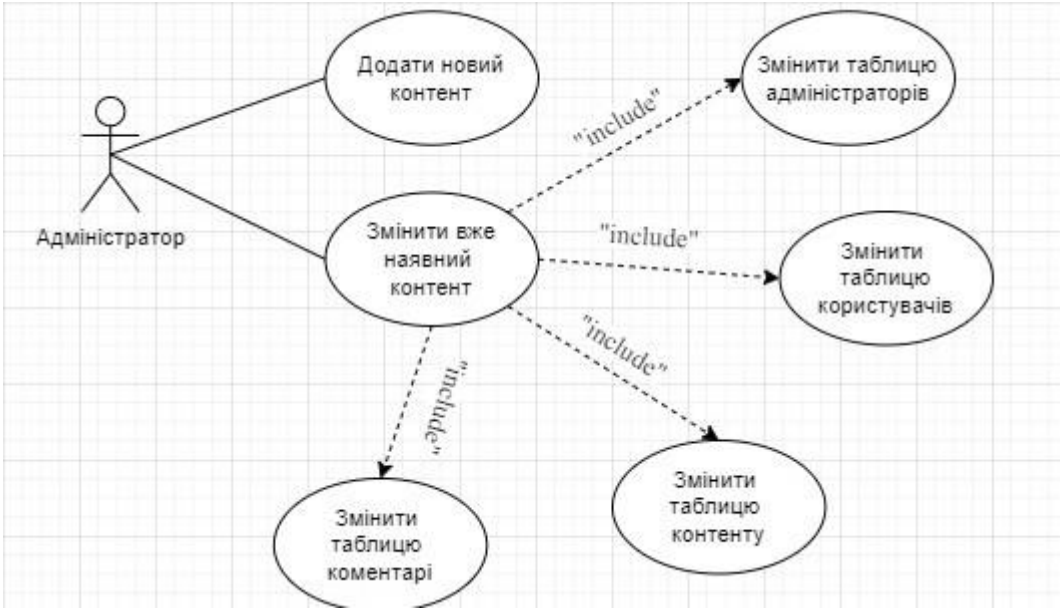

Рисунок 2.2 – Діаграма прецедентів адміністратора

Додати новий контент:

– основна дійова особа: адміністратор;

– інші учасники прецеденту: відсутні;

– зв’язки з іншими варіантами використання: відсутні.

Короткий опис: даний варіант використання дозволяє адміністратору додавати новий контент на сайт.

Змінити вже наявний контент:

– основна дійова особа: адміністратор;
– інші учасники прецеденту: відсутні;
– зв’язки з іншими варіантами використання: присутні.

Короткий опис: даний варіант використання дозволяє адміністратору змінити контнет, який вже є на сайті. Це включає редагування таблиць коментарів, контенту, користувачів та адміністраторів.

2.2 Проектування бази даних

До створення бази даних нам потрібно скласти її фізичну модель.

Фізична модель даних – це модель, описана за допомогою засобів конкретної системи управління базою даних. Фізична модель даних розробляється шляхом додавання особливостей конкретної системи управління базою даних. До таких особливостей можуть ставитися підтримувані системою управління базою даних типи даних, присвоєння імен таблицям, атрибутам, тощо.

Фізична модель даних фактично є готовим технічним завданням на створення бази даних, маючи яку можна реалізувати базу даних в обраній системі управління базою даних.

Наша модель даних складатиметься з 6 сутностей: «User», «Series», «List», «Comments», «UserList», «Admin» (рис. 2.3).

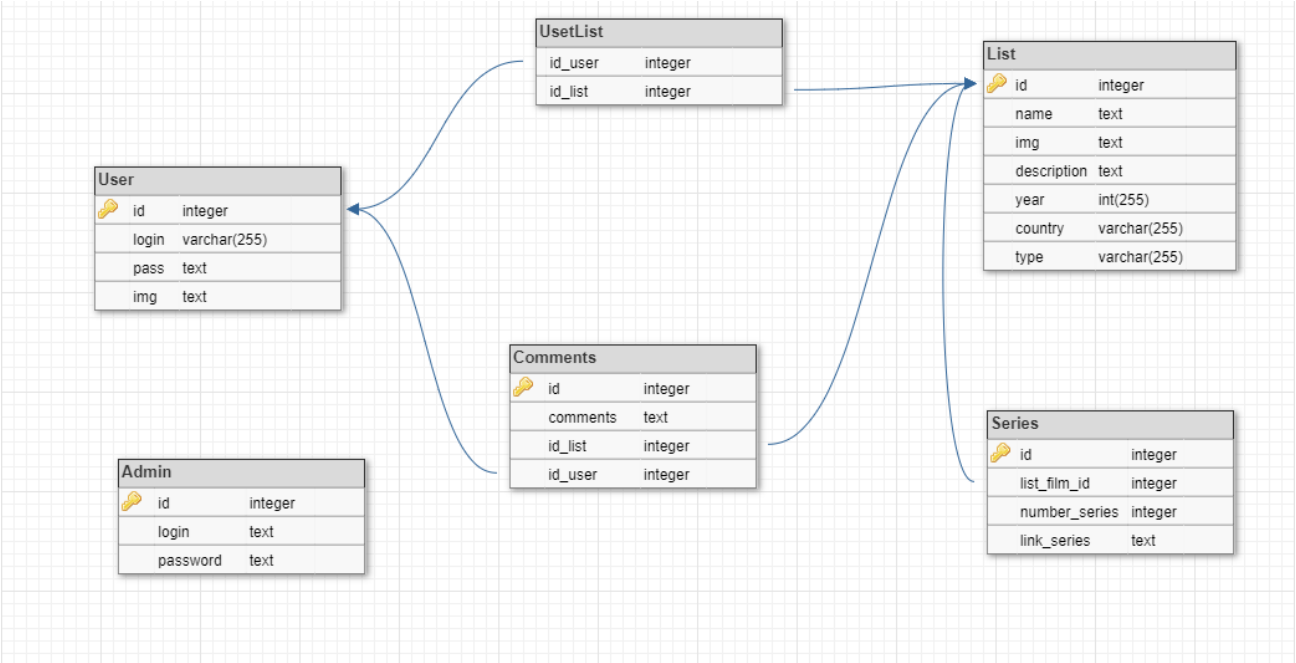

Рисунок 2.3 – Структура бази даних

У таблиці «User» зберігається інформація про користувачів, які зареєструвалися на сайті. Така як: id користувача, логін, пароль та шлях до аватарки. Опис полів таблиці «User» представлено в таблиці 2.1.

Таблиця 2.1 – Таблиця User

Поле	**Тип**	**Ключ**	**NULL**
id	integer	первинний	-
login	varchar	-	-
pass	text	-	-
img	text	-	-

У таблиці «List» зберігається інформація про контент, який представлений на сайті. Така як: id контенту, назва, опис, рік випуску, країна виробник, тип та шлях до постеру. Опис полів таблиці «List» представлено в таблиці 2.2.

Таблиця 2.2 – Таблиця List

Поле	**Тип**	**Ключ**	**NULL**
id	integer	первинний	-
name	text	-	-
img	text	-	-
description	text	-	Так
year	int	-	-
country	varchar	-	-
type	varchar	-	-

У таблиці «Series» зберігається інформація про серії певного серіалу, фільму, тощо. Така як: id серії, id контенту, номер серії та UML-посилання. Опис полів таблиці «List» представлено в таблиці 2.3.

Таблиця 2.3 – Таблиця Series

Поле	**Тип**	**Ключ**	**NULL**
id	integer	первинний	-
list_film_id	integer	зовнішній	-
number_series	integer	-	-
link_series	text	-	-

Так як таблиці «Users» й «List» маю зв’язок «багато до багатьох» була створена сполучна таблиця «UserList». У цій таблиці буде зберігатися інформація про вподобане користувачами. Опис полів таблиці «UserList» представлено в таблиці 2.4.

Таблиця 2.4 – Таблиця UserList

Поле	**Тип**	**Ключ**	**NULL**
id_user	integer	первинний/зовнішній	-
id_list	integer	первинний/зовнішній	-

У таблиці «Comments» зберігається інформація про коментарі користувачів. Така як: id коментаря, коментар, id користувача та id контенту. Опис полів таблиці «Comments» представлено в таблиці 2.5.

Таблиця 2.5 – Таблиця Comments

Поле	**Тип**	**Ключ**	**NULL**
id	integer	первинний	-
comments	text	-	-
id_list	integer	зовнішній	-
id_user	integer	зовнішній	-

У таблиці «Admin» зберігається інформація про адміністраторів сайту. Така як: id адміністратора, логін та пароль. Опис полів таблиці «Admin» представлено в таблиці 2.6.

Таблиця 2.6 – Таблиця Admin

Поле	**Тип**	**Ключ**	**NULL**
id	integer	первинний	-
login	text	-	-
password	text	-	-

3 РОЗРОБКА ВЕБСЕРВІСУ

3.1 Створення бази даних

Для створення бази даних використаємо міграції, які надає Laravel. Для початку потрібно підключитися до phpMyAdmin і зв’язатися з базою. Для локального вебсервера буде використовуватися Open Server.

У phpMyAdmin потрібно створити саму базу. Назвемо її «*popfilm*». Далі у файлі «.env» потрібно зв’язатися з базою (рис.3.1). Всі налаштування для підключення можна подивитися в самому phpMyAdmin.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=popfilm
DB_USERNAME=root
DB_PASSWORD=
```

Рисунок 3.1 – Підключення до бази даних

Далі треба створити файли міграції. Для цього треба скористатися командою *«php artisan make: model User -m –f»*. За допомогою цієї команди відразу ж створюється модель, для роботи з таблицею, а також міграції та факторії для неї.

Для того, щоб створити таблицю використовується метод *create* фасаду *Schema*. Після цього визначмо стовпці таблиці за допомогою методів, які відповідають різним типам стовпців. Нижче наведено приклад створення міграції таблиці User (рис. 3.2).

```php
Schema::create( table: 'users', function (Blueprint $table) {
    $table->bigInteger( column: 'id')->autoIncrement();
    $table->string( column: 'login');
    $table->string( column: 'password');
    $table->string( column: 'img')->nullable();
    $table->timestamps();
});
```

Рисунок 3.2 – Створення таблиці «Users» за допомогою міграції.

За таким же принципом створюємо інші таблиці.

Після цього треба створити тестові записи для таблиці «*List_Films*», щоб було легше розробляти сайт. Для цього потрібно скористатися файлом факторії. Через властивість *faker* факторії мають доступ до PHP-бібліотеки Faker, яка дозволяє зручно генерувати різного роду випадкові дані. Приклад створення можемо побачити на рисунку 3.3.

```php
public function definition()
{
    return [
        'name' => $this->faker->name(),
        'img' => $this->faker->text(),
        'description'=>$this->faker->text( maxNbChars: 200),
        'year'=>$this->faker->year(),
        'country'=>$this->faker->text( maxNbChars: 10),
        'type'=>$this->faker->text( maxNbChars: 8)
    ];
}
```

Рисунок 3.3 – Факторія таблиці «List_Films»

Згенеруємо ці дані. Для цього скористаємося класом-наповнювачем. Згенеруємо 200 записів в базу, а так само треба перевизначити властивість «img», щоб занести в базу шлях до тестового зображення. Також відразу ж згенеруємо профіль адміністратора сайту (рис. 3.4).

```php
public function run()
{
    ListFilm::factory( count: 200)->create([
        'img'=>'images/2poster.jpg'
    ]);
    Admin::factory( count: 1)->create([
        'login'=>'Admin',
        'password'=>bcrypt( value: 'admin')
    ]);
}
```

Рисунок 3.4 – Клас-наповнювач

Треба скористатися командою *«php artisan migrate –seed»*, для того, щоб запустити всі міграції та наповнювачі.

3.2 Метод створення уявлень. Реалізація реєстрації та авторизації на сайті

Почнемо розробку зі створення уявлень реєстрації та авторизації. Так як Laravel використовує шаблонізатор Blade, кожне створюване уявлення повинно мати розширення файлу *.blade.php*. Усі уявлення зазвичай зберігаються в каталозі *resources / views*.

Для того, щоб уникнути дублювання коду треба створити в директорії views папку layouts, де буде зберігатися макети для сайту. Наприклад, щоб не дублювати в кожному файлі код створення хедера та футера – винесемо їх в окремий батьківський шаблон, а потім при визначенні дочірнього шаблону використовувати директиву @extends, щоб вказати, який макет дочірній шаблон повинен успадковувати. Шаблони, що розширюють макет Blade, можуть додавати вміст у секції макета за допомогою директив @section

Для того, щоб відобразити сторінку з реєстрацією, потрібно налаштувати файл маршрутизації. Це робиться в файл *web.php* в директорії

routes. Найпростіші маршрути у Laravel приймають URI і замикання, забезпечуючи нетрудомісткий і виразний метод визначення маршрутів і поведінки без складних конфігураційних файлів маршрутизації (рис 3.5).

```
use Illuminate\Support\Facades\Route;

Route::get('/greeting', function () {
    return 'Hello World';
});
```

Рисунок 3.5 – Найпростіший маршрут у Laravel

Щоб створити контролер потрібно скористатися командою *php artisan make: controllers AuthController*. У контролері створимо функцію *showRegisterForm*, яка відобразить сторінку з формою реєстрації (рис. 3.6).

```
public function showRegisterForm() {
    return view( view: 'register');
}
```

Рисунок 3.6 – Функція для відображення сторінки

Відразу ж створимо функцію register, яка буде відповідати за завантаження нового користувача в базу даних.

До того, як додати новий запис в таблицю потрібно провалідувати дані, які вводить користувач. Для цього використаємо метод *validate*, що надається об’єктом *Request*. Якщо правила валідації будуть пройдені, то код продовжить нормально виконуватися. Однак, якщо перевірка не пройде валідацію, то буде створено виняток *ValidationException* і відповідна відповідь про помилку буде автоматично відправлений назад користувачеві.

Після того, як дані пройшли валідацію потрібно їх вставити в таблицю. Для цього скористаємося моделлю, яка була створена разом з міграціями. Для

того, щоб записати новий дані в таблицю потрібно скористатися методом *create*. Важливо, щоб працював даний метод попередньо потрібно в моделі вказати властивість *fillable*, в якому вказати всі параметри з таблиці, які можуть змінюватися.

Останнє, що залишилося зробити, це авторизуватися користувача. У двох словах, це працює так, що при використанні веб браузера користувач вводить логін і пароль через форму входу. Якщо ці облікові дані вірні, то додаток буде зберігати інформацію про аутентифікованого користувача в сесії користувача. Файл cookie, відправлений браузеру, містить ідентифікатор сесії, щоб наступні запити до додатка могли пов'язати Користувача з правильною сесією. Після отримання файлу cookie сесії, додаток витягує дані сесії на основі ідентифікатора сесії, зазначає, що аутентифікаційна інформація була збережена в сесії, і розглядає користувача як «аутентифікованого». Аутентифікуємо користувача через метод *login* фасаду *Auth*. Кінцевий результат розробки функції реєстрації можна побачити на рисунку 3.7:

```
public function register(Request $request) {
    $data = $request->validate([
        'name'=>['required','string','unique:users,login'],
        'password'=>['required','confirmed']
    ]);

    $user = User::create([
        'login'=>$data['name'],
        'password'=>bcrypt($data['password']),
        'img'=>'/images/avatar_placeholder.png'
    ]);

    if($user) {
        auth( guard: 'web')->login($user);
    }

    return redirect(route( name: 'home'));
}
```

Рисунок 3.7 – Функція реєстрації нового користувача

Залишилося реалізувати функцію авторизації для користувачів, який вже зареєстровані, але не авторизовані. Аналогічно провалідуємо дані, які ввів користувач. Скористаємося методом *attempt* для авторизації. Метод attempt приймає масив значень як свій перший аргумент. Значення в масиві будуть використовуватися для пошуку користувача в таблиці бази даних. Якщо користувач знайдений, то хешований пароль, що зберігається в базі даних, буде порівнюватися зі значенням password, переданим в метод через масив. Якщо два хешованих пароля збігаються, то для користувача буде запущена аутентифікована сесія. Кінцева версія функції представлена на рисунку 3.8.

```
public function login(Request $request) {
    $data = $request->validate([
        'login'=>['required','string'],
        'password'=>['required']
    ]);

    if(auth( guard: 'web')->attempt($data)) {
        return redirect(route( name: 'home'));
    }else {
        return redirect(route( name: 'login'))->withErrors([
            'login'=>'Пользователь не найден, или данные введены не верно'
        ]);
    }
}
```

Рисунок 3.8 – Функція авторизації користувача

Допишемо файл маршрутизації. Передаймо в маршрути функції, які створили в контролері. Також згрупуємо їх за посередником «guest», щоб тільки неавторизовані користувачі могли зареєструватися та увійти на сайт (рис. 3.9).

```php
Route::middleware( middleware: 'guest')->group(function () {
    Route::get( uri: '/login', [\App\Http\Controllers\AuthController::class, 'showLoginForm'])
        ->name( name: 'login');
    Route::post( uri: '/login_process', [\App\Http\Controllers\AuthController::class, 'login'])
        ->name( name: 'login_process');

    Route::get( uri: '/register', [\App\Http\Controllers\AuthController::class, 'showRegisterForm'])
        ->name( name: 'register');
    Route::post( uri: '/register_process', [\App\Http\Controllers\AuthController::class, 'register'])
        ->name( name: 'register_process');
});
```

Рисунок 3.9 – Кінцевий варіант файлу маршрутизації

3.3 Реалізація панелі адміністратора

Для того, щоб адмініструвати сайт потрібно створити адмін панель. Для цього потрібно виконати наступні кроки: створити новий файл маршрутів, а також створити нового захисника. Це потрібно для того, щоб логіка не перепліталася з основним функціоналом сайту і щоб до адмін панелі могли отримати доступ тільки адміністратори сайту.

Почнемо зі створення нового файлу маршрутів. Для цього в каталозі routes треба створити новий файл admin.php. Для того, щоб приєднати його до сайту потрібно перейти в директиву *App\Providers* до файлу *RouteServiceProvider*. Створимо нового посередника з ім’ям «admin». *Prefix* і *name* вказані для того, щоб всі адреси в адмін панелі починалися з admin/... (рис. 3.10).

```php
Route::middleware( middleware: 'admin')
    ->prefix( prefix: 'admin')
    ->name( value: 'admin_')
    ->group(base_path( path: 'routes/admin.php'));
```

Рисунок 3.10 – Підключення нового фалу маршрутизації

Далі додаємо нового охоронця для авторизації адміністраторів. Це дозволить керувати аутентифікацією для окремих частин програми з

використанням абсолютно окремих аутентифікованих моделей або таблиць. Для цього потрібно перейти в директиву *config* до файлу *auth.php*. Тут потрібно в конфігурації «guards» створити нового захисника, а в «providers» вказати модель таблиці, в якій зберігається інформація про адміністраторів. У нашому випадку це модель Admin.

Перейдемо до створення нових уявлень і контролерів. Для адмін панелі створимо окремі директорії в папках *controllers* і *views*. Далі потрібно створити контролер для авторизації адміністраторів. Функція авторизації практично не буде відрізнятися від тієї, що була створена раніше, так що не будемо на цьому зупинятися. Оскільки реєстрації адміністраторів не буде, то й функції реєстрації у так само не буде.

В адмін панелі нам потрібно реалізувати функціонал перегляду, створення, редагування і видалення записів з таблиць. Оскільки вся логіка від таблиці до таблиці буде практично однакова, то розглянемо реалізацію на прикладі таблиці *List_films*. Отже, створимо новий контролер. Цього разу це буде ресурсний контролер. Ресурсні контролери дозволяють полегшити створення CRUD-функцій за рахунок повної уніфікації всіх маршрутів і способів їх обробки. Замість опису 7 різних маршрутів, ресурсна маршрутизація дозволяє вказати всього один (рис. 3.11).

```
use App\Http\Controllers\PostController;

Route::resource('posts', PostController::class);
```

Рисунок 3.11 – Маршрутизація ресурсного контролеру

У проектах де подібних CRUD багато, ресурсний маршрутизатор дуже допомагає. Він не просто скорочує кількість коду, але і дає хорошу уніфікацію. Потрібно менше думати та менше сперечатися. Все вже спроектовано.

Для того, щоб створити контролер потрібно скористатися командою *make: controller* також додавши додатковий параметр *--resource*. Це створить контролер з тими самими сімома маршрутами. У таблиці 3.1 представлені дії, які виконує цей контролер.

Таблиця 3.1 – Дії, що виконуються ресурсними контролерами

Метод	**URI**	**Дія**	**Ім'я маршруту**
GET	/posts	index	posts.index
GET	/posts/create	create	posts.create
POST	/posts	store	posts.store
GET	/posts/{post}	show	posts.show
GET	/posts/{post}/edit	edit	posts.edit
PUT/PATCH	/posts/{post}	update	posts.update
DELETE	/posts/{post}	destroy	posts.destroy

Насамперед зробимо відображення вже наявних записів в базі. Для цього скористаємося функцією *index* в контролері. Також зробимо пагінацію, щоб на одній сторінці відображалося 10 записів з бази. Для цього треба використати метод *paginate*. Метод paginate автоматично встановлює «обмеження» та «зміщення» у запиті на основі поточної сторінки, яку переглядає користувач. За замовчуванням поточна сторінка визначається значенням аргументу page рядка HTTP-запиту. Це значення автоматично визначається Laravel, а також автоматично вставляється в посилання, що генеруються пагінатором.

Метод paginate підраховує загальну кількість записів, що відповідають запиту, перед витяганням записів з бази даних. Це зроблено для того, щоб пагінатор знав, скільки всього сторінок із записами необхідно сформувати.

Кінцевий варіант функції можна побачити на рисунку 3.12.

```
public function index()
{
    $list = listFilm::query()->paginate( perPage: 10);
    return view( view: 'Admin.index',[
        'lists'=>$list
    ]);
}
```

Рисунок 3.12 – Функція відображення записів в базі

Далі реалізуємо функцію додавання нових записів в базу даних. Для цього скористаємося функцією *create* для відображення сторінки створення, а так само функцією *store* для безпосередньо запису в БД. У функції store потрібно валідувати дані, які будуть введені. Оскільки буде завантажуватися зображення, то потрібно змінювати його розмір, для оптимізації завантаження вебсайту. Для цього скористаємося сторонньою бібліотекою *Intervention Image*, яка дозволяє змінити розміри зображення з допомогою всього одного методу *resize*. Всі постери фільмів будуть зберігатися в директорії *public / images*

Після цього створимо новий запис даних за допомогою моделі ListFilm і методу create. В кінці робимо редирект на сторінку з відображенням всіх записів в таблиці (рис. 3.13).

Для реалізації редагування і видалення записів нам знадобляться методи PUT / PATCH та DELETE. Через те, що HTML-форми не підтримують подібні дії нам потрібно буде створити прихований *input* з ім’ям *_method*. Значення, відправлене з полем _method, буде використовуватися як метод HTTP-запиту. Для зручності можна використовувати директиву *@method* шаблонізатора Blade для створення поля введення _method (рис. 3.14)

```php
public function store(Request $request)
{
    $data = $request->validate([
        'name'=>['required','string'],
        'description'=>['required','string'],
        'year'=>['required','numeric'],
        'type'=>['required','string'],
        'country'=>['required','string'],
        'img'=>['required','image']
    ]);

    $imageName = time().'.'.$request->img->getClientOriginalName();
    Image::make($request->img)->resize(400, 560)->save(public_path( path: 'images/').$imageName);

    ListFilm::create([
        'name'=>$data['name'],
        'description'=>$data['description'],
        'year'=>$data['year'],
        'type'=>$data['type'],
        'country'=>$data['country'],
        'img'=>'images/' . $imageName
    ]);

    return redirect(route( name: 'admin_posts.index'));
}
```

Рисунок 3.13 – Функція створення нових записів у таблиці «ListFilm»

```blade
<form action="/example" method="POST">
    @method('PUT')
    @csrf
</form>
```

Рисунок 3.14 – Приклад створення форми с методом PUT

Коротко про те, що таке CSRF у формі. CSRF – це «токен», який Laravel генерує для кожної активної користувацької сесії, керованої додатком. Цей токен використовується для перевірки того, що автентифікований користувач дійсно є особою, яка виконує запити до додатка. Оскільки цей токен зберігається в сесії користувача і змінюється кожен раз при повторному створенні сесії, шкідливий додаток не може отримати до нього доступ.

Кожного разу, коли створюємо HTML-форму «POST», «PUT», «PATCH» або «DELETE» у своєму додатку, розробник повинен включати в форму приховане поле _TOKEN CSRF, щоб посередник CSRF міг перевірити запит. Для зручності можна використовувати директиву Blade @csrf для створення прихованого поля введення, що містить токен.

Для сторінки з оновленням запису використаємо те ж уявлення, що і для створення, але у випадку з оновленням даних занесемо всі дані про запис в input, попередньо отримавши їх з бази.

Після цього в функції update потрібно знову валідувати дані. Після цього можна оновити запис по id через метод *findOrFail ($id)->update ()* передавши в метод update дані масивом. Кінцевий варіант функції можна побачити на рисунку 3.15.

```php
public function update(Request $request, $id)
{
    $data = $request->validate([...]);

    $post = ListFilm::query()->findOrFail($id);

    if(isset($data['img'])) {
        $imageName = time().'.'.$request->img->getClientOriginalName();
        Image::make($request->img)->resize(400, 560)->save(public_path( path: 'images/').$imageName);
    }

    if(isset($imageName)) {
        $poster = 'images/'.$imageName;
    }
    else {
        $poster = $post['img'];
    }

    $post->update([
        'name'=>$data['name'],
        'description'=>$data['description'],
        'year'=>$data['year'],
        'type'=>$data['type'],
        'country'=>$data['country'],
        'img'=> $poster,
    ]);
```

Рисунок 3.15 – Функція оновлення записів в базі

Для видалення записів скористаємося функцією destroy і методом *destroy()* (рис. 3.16).

```
public function destroy($id)
{
    ListFilm::destroy($id);
    return back();
}
```

Рисунок 3.16 – Функція видалення записів у базі

3.4 Опис роботи з вебсайтом

Як тільки користувач потрапляє на сайт, він бачить головну сторінку на якій відображаються рекомендовані фільми, опис сайту. Також у верхній частині екрана є «шапка» сайту з навігаційним меню, а також кнопка реєстрації та входу. З головної сторінки користувач може потрапити на сторінку реєстрації, авторизації, обраного жанру, а так само безпосередньо на сторінку з фільмом (рис. 3.17).

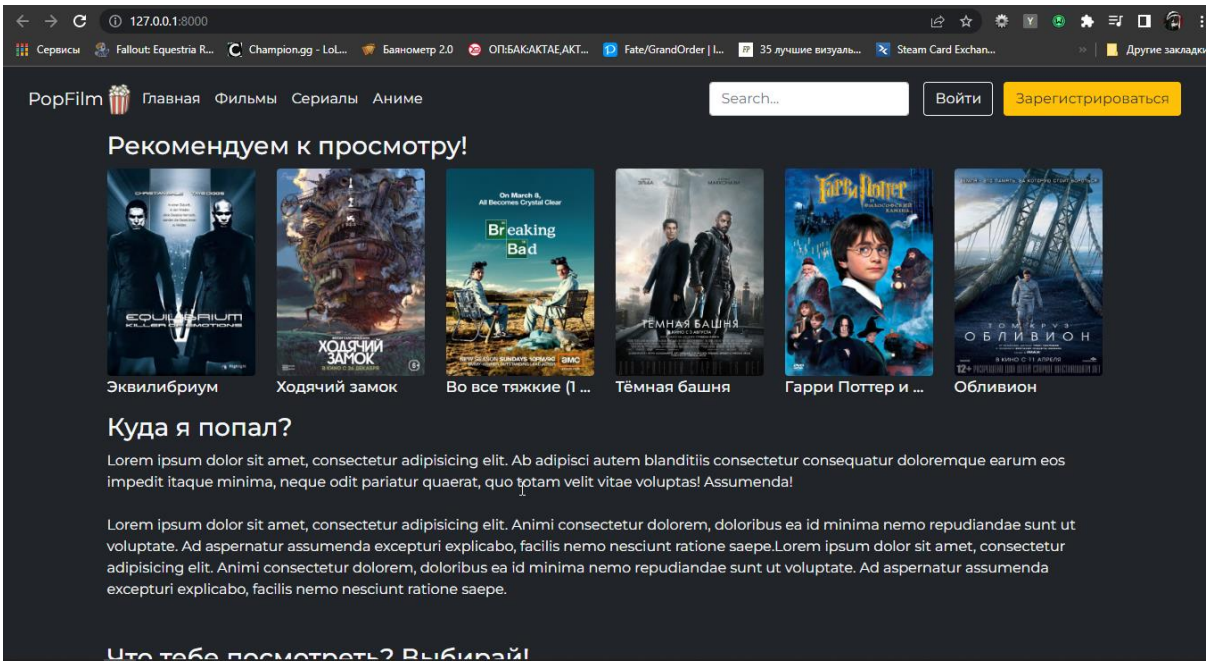

Рисунок 3.17 – Головна сторінка сайту

При натисканні на кнопку зареєструватися або увійти користувач потрапить на відповідну сторінку з формою (рис. 3.18).

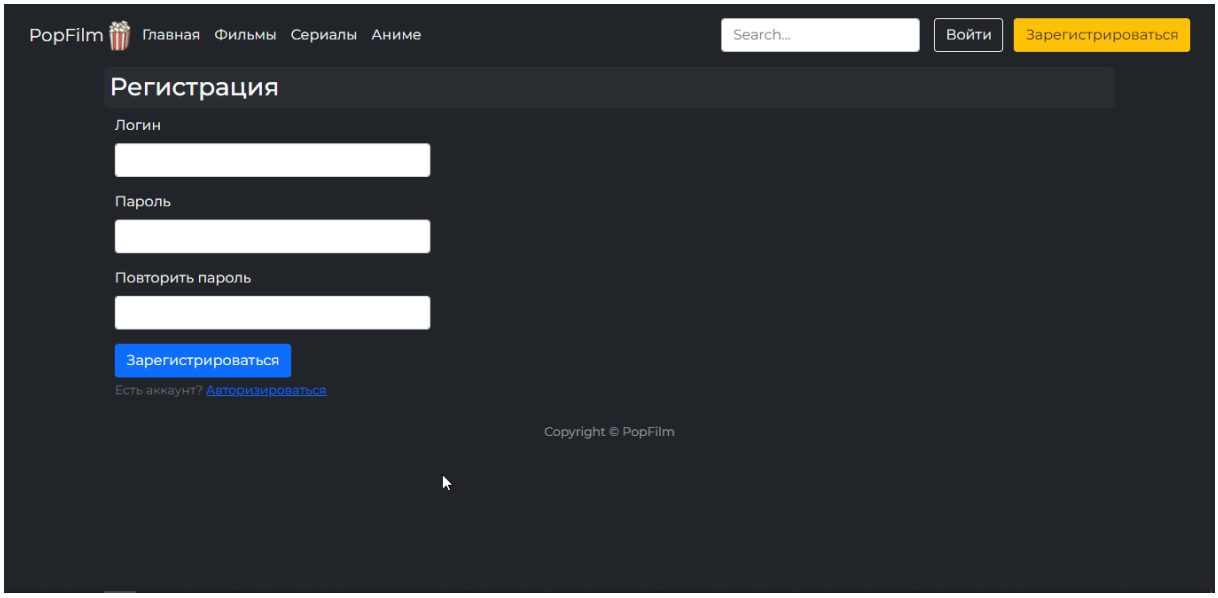

Рисунок 3.18 – Форма реєстрації на сайті

Незалежно від того, авторизований користувач чи ні, він може перейти на сторінку з будь-яким фільмом і переглянути його. Якщо користувач все ж авторизований він може залишати коментарі та додавати фільми в список вподобаного (рис. 3.19).

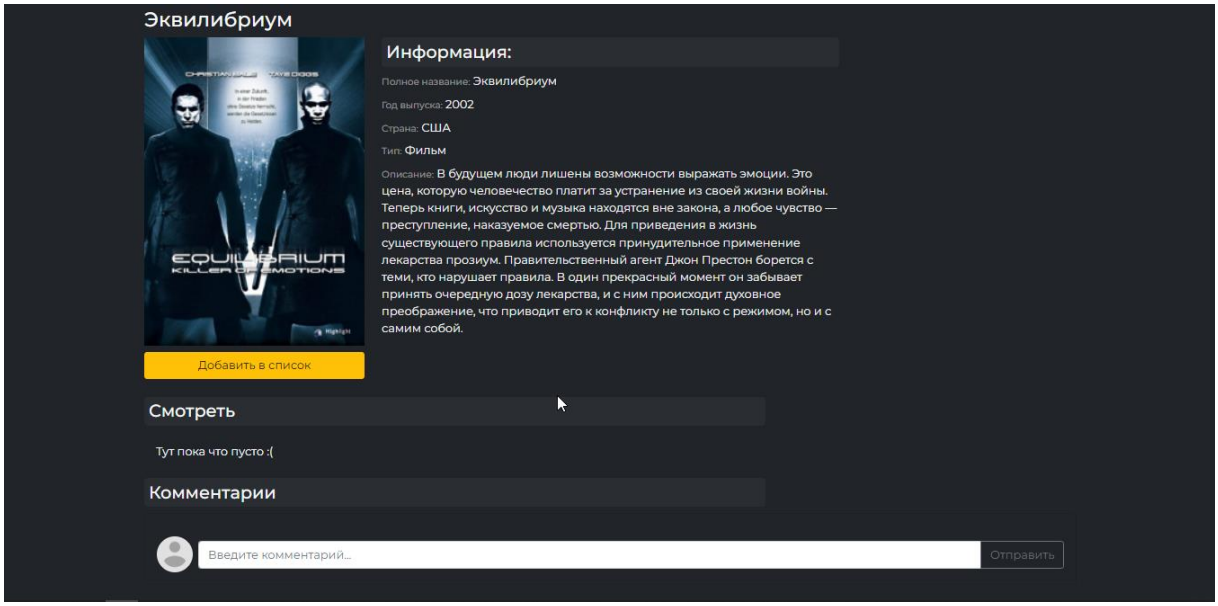

Рисунок 3.19 – Сторінка фільму

Для адміністратора існує своя сторінка авторизації. Щоб потрапити на неї потрібно перейти на сторінку admin/login. Після авторизації адміністратор потрапляє на сторінку з редагуванням таблиць (рис. 3.20).

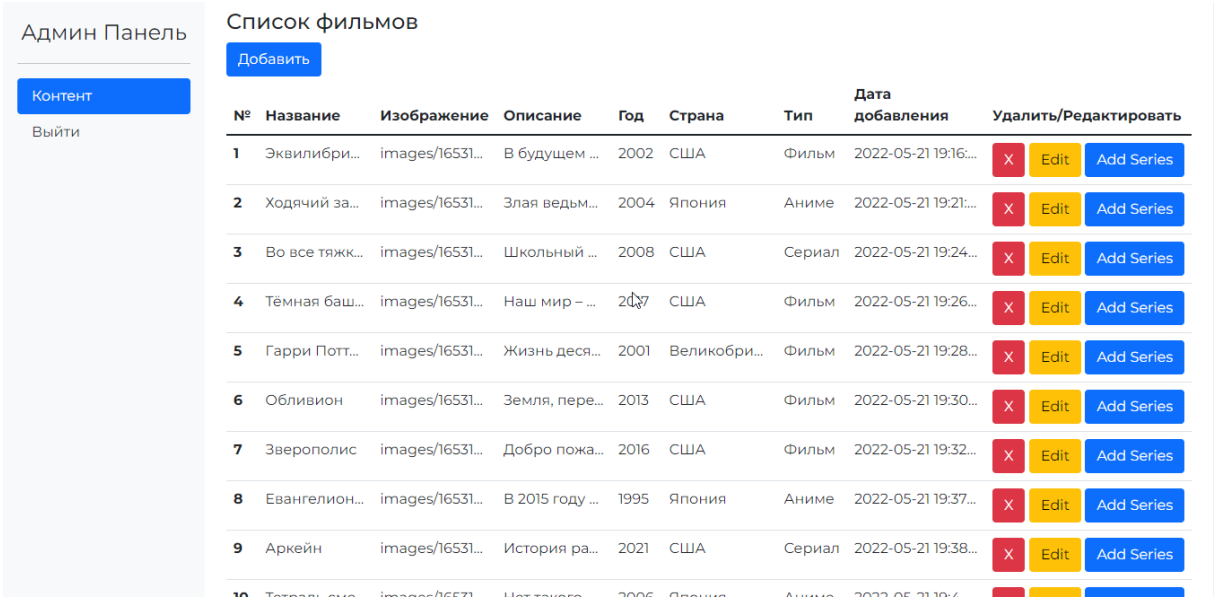

Рисунок 3.20 – Панель адміністратора

ВИСНОВКИ

В рамках кваліфікаційної роботи був розроблений вебдодаток онлайн-кінотеатру.

Даний сайт дозволяє переглядати фільми, серіали, мультфільми. додавати улюблені фільми в список для ведення статистики, а так само залишати коментарі.

Під час створення сайту були поліпшені навички програмування на мові PHP. Також були отримані широкі знання з використання фреймворку Laravel, яка прискорила розробку в кілька разів, а так само зробила код набагато читабельніше. Завдяки фреймворку було реалізовано три рівні доступу: гість, зареєстрований користувач і адміністратор. Була використана маршрутизація, яка була надана фреймворком, що значно підвищила захищеність бази даних.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. McCool, S. (2012). *Laravel Starter* (pp. 63-70). Packt Publishing.
2. Dockins, K. (2017). Design Patterns in PHP and Laravel (pp. 100-120). Apress.
3. Lengstorf, J., & Wald, K. (2010). Pro PHP and jQuery (pp. 20-23). Apress.
4. Nixon, R. (2021). Learning PHP, MySQL & JavaScript (pp. 267-273). «O'Reilly Media, Inc.».
5. Safronov, M., & Winesett, J. (2014). Web application development with Yii 2 and PHP (pp. 37-39). Packt Publishing Ltd.
6. Potencier, F. (2020). Symfony 5: The Fast Track. Symfony SAS (pp. 10-11).
7. Stauffer, M. (2019). Laravel: Up & running: A framework for building modern php apps (pp. 25-26). O'Reilly Media.
8. Gabardo, A. C. (2017). Laravel para ninjas (pp. 7-18). Novatec Editora.
9. He, R. Y. (2015, January). Design and implementation of web based on Laravel framework. In 2014 International Conference on Computer Science and Electronic Technology (ICCSET 2014) (pp. 301-304). Atlantis Press.
10. Ullman, L. (2012). PHP Advanced and Object-oriented Programming: Visual Quickpro Guide. Peachpit Press.
11. Cui, W., Huang, L., Liang, L., & Li, J. (2009, November). The research of PHP development framework based on MVC pattern. In 2009 Fourth International Conference on Computer Sciences and Convergence Information Technology (pp. 947-949). IEEE.
12. Affandi, Z., & Orlando, E. (2011). Web Design Applications For Helmet Marketing Sales In Helmet Store With Php, Mysql, And Macromedia Dreamweaver 8.0. Jurnal Sistem Informasi STMIK Jakarta STI&K, 212569.
13. Duckett, J. (2011). HTML & CSS: design and build websites (Vol. 15). Indianapolis, IN: Wiley.

14. Priambodo, M., & Noer, F. (2022). Implementasi Framework Laravel Untuk Pengembangan Website Kamar. Co. Id Menggunakan Laravel 8.

15. Pitt, C. (2012). Pro PHP 8 MVC.

16. Мартин, Р. (2013). Чистый код: Создание, анализ и рефакторинг. библиотека программиста. Издательский дом «Питер».

17. Зандстра, М. (2011). PHP: объекты, шаблоны и методы программирования, 3-е изд. С. 79-86. ИД Вильямс.

18. Дунаев, В. (2012). Сценарии для Web-сайта. PHP и JavaScript. С. 28-36. БХВ-Петербург.

19. Файзрахманов, А. (2020). Архитектура сложных веб-приложений. С примерами на Laravel.

20. Ульман, Л. (2022). Основы программирования на PHP. 63 с. Litres.

21. Веллинг, Л., & Томсон, Л. (2011). Разработка веб-приложений с помощью PHP и MySQL. С. 158-168.

22. Laravel documentation. Laravel - The PHP Framework For Web Artisans. URL: https://laravel.com/ (дата звернення 11.05.2022).

23. Bootstrap documentation. Bootstrap · The most popular HTML, CSS, and JS library in the world. URL: https://getbootstrap.com/docs/ (дата звернення 11.05.2022).

24. PHP: Руководство по PHP - Manual. PHP: Hypertext Preprocessor. URL: https://www.php.net/manual/ru/ (дата звернення 11.05.2022).