Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка інтерактивного сайту «KinoTale» для перегляду і сортування фільмів з використанням Vue.js, Node.js та mongoDB

Виконала: студентка IV курсу, групи СНс-42
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

(підпис) Івашенюк І.О. (прізвище та ініціали)

Керівник (підпис) Липак Г.І. (прізвище та ініціали)

Нормоконтроль (підпис) Шимчук Г.В. (прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О. (прізвище та ініціали)

Рецензент (підпис) Оробчук О.Р. (прізвище та ініціали)

Тернопіль
2022

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Івашенюк Ірини Олександрівни
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка інтерактивного сайту «KinoTale» для перегляду і сортування фільмів з використанням Vue.js, Node.js та mongoDB.

Керівник роботи Липак Галина Ігорівна, к. соц. комун., доц. каф. КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 16 » березня 2022 року № 4/7-161

2. Термін подання студентом завершеної роботи 22.06. 2022 р.

3. Вихідні дані до роботи Літературні та Інтернет-джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Аналіз предметної області та постановка задач проєктування інтерактивного сайту «KinoTale» для перегляду і сортування фільмів. 1.1. Аналіз еволюції кінотеатрів. 1.2. Постановка завдання розробки інтерактивного сайту «KinoTale». 1.3. Формування вимог до інтерактивного сайту для перегляду і сортування фільмів. 1.4. Проєктування сайту для перегляду і сортування фільмів. 1.5. Прототипування сайту. 1.6. Висновки до першого розділу. 2. Розробка інтерактивного сайту «KinoTale» для перегляду і сортування фільмів . 2.1. Технології для створення інтерактивного сайту «KinoTale» 2.1.1. Фреймворк Vue.js для перегляду та сортування фільмів. 2.1.2 Платформа Node.js для написання серверної частини сайту «KinoTale». 2.1.3 База даних MongoDB. 2.2. Практична реалізація кваліфікаційної роботи. 2.2.1. Серверна частина. 2.2.2. Частина опису фронтенду кваліфікаційної роботи. 2.3. Оцінка одержаних результатів кваліфікаційної роботи «KinoTale».2.3. Висновки до другого розділу. 3. Безпека життєдіяльності, основи охорони праці. 3.1. Вплив діяльності людини на довкілля. 3.2. Характеристика приміщення при роботі інтерактивного сайту «KinoTale» для перегляду та сортування фільмів щодо небезпеки ураження електричним струмом, пожежній небезпеці, вибухонебезпеці. Висновки. Перелік джерел. Додатки.

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Гурик О.Я., доцент кафедри МТ		

7. Дата видачі завдання 24 січня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	24.01.2022	*Виконано*
2.	Підбір джерел про побудову та функціонал сайту	04.01.2022-30.01.2022	*Виконано*
3.	Переклад та опрацювання джерел про можливе розширення сайту	31.01.2022-06.02.2022	*Виконано*
4.	Виконання дослідження щодо розробки запланованого функціоналу	07.02.2022-13.02.2022	*Виконано*
	Розроблення фунціоналу сайту		
5.	Оформлення розділу « Аналіз предметної області та	14.02.2022-06.03.2022	*Виконано*
	Постановка задач проєктування інтерактивного сайту «KinoTale» для перегляду та сортування фільмів»		
6.	Оформлення розділу « Розробка інтерактивного сайту	07.03.2022-03.04.2022	*Виконано*
	«KinoTale» для перегляду і сортування фільмів»		
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	04.04.2022-17.04.2022	*Виконано*
8.	Виконання завдання до підрозділу «Основи охорони	18.04.2022-01.05.2022	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	02.05.2022-15.05.2022	*Виконано*
10.	Нормоконтроль	16.05.2022-22.05.2022	*Виконано*
11.	Перевірка на плагіат	08.06.2022	*Виконано*
12.	Попередній захист кваліфікаційної роботи	09.06.2022	*Виконано*
13.	Захист кваліфікаційної роботи	22.06.2022	

Студент _____ Івашенюк І.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Липак Г.І.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка інтерактивного сайту «KinoTale» для перегляду і сортування фільмів з використанням Vue.js, Node.js та mongoDB // Кваліфікаційна робота освітнього рівня «Бакалавр» // Івашенюк Ірина Олександрівна // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНс-42 // Тернопіль, 2022 // С. 58, рис. – 15, табл. – 0, кресл. – 0, додат. – 3, бібліогр. – 24.

Ключові слова: vue.js, javascript, база даних, веб-сайт, кінострічка, онлайн кінотеатр, користувач, інтерфейс.

Кваліфікаційна робота присвячена побудові зручного інтерфейсу та зручного функціоналу для перегляду фільмів.

Мета роботи: розробка інтерактивного сайту для перегляду фільмів, з можливістю сортування кінострічок по жанрах та акторах, з подальшим розвитком сайту для збільшення клієнтів.

В першому розділі кваліфікаційної роботи розглянуто вимоги до кваліфікаційної роботи, здійснено постановку завдань до веб-сайту, проєктування та прототипування сайту, побудову діаграм та ключових елементів до проєктування кваліфікаційної роботи.

В другому розділі кваліфікаційної роботи описано вибір середовища розробки та технологій для програмування сайту, розроблений функціонал та створення і наповнення бази даних кінострічками.

ANNOTATION

Development of an interactive site «KinoTale» for movies viewing and sorting using Vue.js, Node.js and mongoDB // Qualification work of the educational level "Bachelor" // Ivashenyuk Irina Alexandrovna // Ivan Puliuj Ternopil National Technical University, Faculty of Computer Information Systems and Software Engineering, Computer Science Department, group SNc-42 // Ternopil, 2022 // P.58, pic. -15, table. -0, drawings. -0, add. -3, bibliogr. -24.

Keywords: vue.js, javascript, database, website, film, online cinema, user, interface.

Qualification work is devoted to building a user-friendly interface and user-friendly functionality for watching movies.

Purpose: development of an interactive site for watching movies, with the ability to sort movies by genres and actors, with the further development of the site to increase customers.

In the first section of the qualification work the requirements for the qualification work are considered, tasks for the website are set, the site is designed and prototyped, diagrams and key elements for the diploma design are built.

The second section of the qualification work describes the choice of development environment and technologies for site programming, developed functionality and the creation and filling of the database with films.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПК – персональний комп’ютер.

CSS – cascading style sheets (каскадні таблиці стилів).

HTML – hypertext markup language (мова розмітки гіпертекстових документів).

JS (javascript) – це невибаглива до ресурсів мова програмування з функціями першого класу, код якої інтерпретується та компілюється під час виконання.

БД – база даних.

DB (database) – це організована структура, яка призначена для зберігання, зміни та обробки взаємозалежної інформації, переважно великих обсягів.

SQL – (tructured query language) мова структурованих запитів, декларативна мова програмування для взаємодії користувача з базами даних.

DFD – (Data Flow Diagram) модель проектування, графічне представлення «потоків» даних в інформаційній системі.

DSD – (Direct Stream Digital) файловий формат.

Er-діаграма – (Entity-relationship model або entity-relationship diagram) модель даних, яка дозволяє описувати концептуальні схеми за допомогою узагальнених конструкцій блоків.

CLI – Command Line Interface.

NPM – (Node Package Manager) це менеджер пакунків для мови програмування JavaScript.

SPA – (Single Page Application) односторінковий застосунок.

HTTP – (HyperText Transfer Protocol) протокол передачі гіпертекстових документів.

SEO – (search engine optimization) пошукова оптимізація сайту.

ЗМІСТ

ВСТУП

Актуальність теми. Під терміном «глобалізація» сьогодні називають великий різновид речей та подій. До цього терміну також відноситься інтернет, за допомогою нього відбуваються кардинальні дії та нові можливості для всього світу та його користувачів.

Безліч процесів, які раніше потребували фізичної присутності чи роботи у довготривалій перспективі, тепер стали легшими і простішими внаслідок появи інтернету. Також з цим відкриттям почали з’являтися все нові і нові професії, одна я яких – програміст. Програмісти це такі люди, які за допомогою інтернету та мов програмування осучаснюють та розвивають процеси життєдіяльності для полегшення людського життя або ж просто розробляються різні застосунки для розваги чи відпочинку.

Перегляд кінострічок – це заняття, яке завжди приносило задоволення та допомагало відпочити після важкого робочого дня, також перегляд фільмів являється одним із глобальних процесів.

Переглянути улюблений фільм можна в кінотеатрі, для цього потрібно зробити врахування багатьох моментів, таких як: наявність вільних місць у залі, час проходження сеансу, час на черги як до каси за білетами так і час на дорогу, і ще можливі труднощі. Майже всі кінотеатри мають власні сайти, де можна уникнути всіх цих незручностей, переглянути сеанси, вільні місця і купити квиток онлайн, але прогрес пішов ще далі.

Актуальності набирає перегляд фільмів онлайн із власного комп’ютера, це можна зробити будь-коли і будь-де. За даними дослідження все більше людей обирають перегляд кінострічок онлайн і не витрачаючи при цьому багато часу на кінотеатр, це підтверджує світовий ринок стрімінгових сервісів, який зріс на 37% за останні два роки [1].

Цей ринок є конкурентоспроможним, що в свою чергу надає можливість шукати кращих варіантів для власного кіно-сайту, модернізація якого

привернула би більшу кількість користувачів. В даній кваліфікаційній роботі розроблятиметься інтерактивний сайт «KinoTale».

Мета і задачі дослідження. Метою даної кваліфікаційної роботи є розробка інтерактивного сайту для перегляду фільмів, з можливістю сортування кінострічок по жанрах та акторах, з подальшим розвитком сайту для збільшення клієнтів.

Для досягнення поставленої мети було сформовано наступний список завдань:

– провести аналіз ринку для детального розуміння ніші онлайн кінотеатрів;

– здійснити проектування дизайну та логіки сайту;

– розробити основний функціонал сайту;

– побудувати базу даних з фільмами;

– створити вікна реєстрації для клієнтів сайту.

Об’єкт дослідження: Методи, засоби та технології розробки інтерактивних сайтів.

Предмет дослідження: Розробка інтерактивного сайту для перегляду та сортування фільмів «KinoTale» засобами Vue.js 3.0, Node.js 17.0.0, MongoDB 4.3.1 HTML5 та CSS3.

Практичне значення одержаних результатів. Полягає у можливості подальшого розвитку та модернізації сайту, збільшення бази даних з фільмами, підтримки готового функціоналу та розробці нового; покращення інтерфейсу сайту для зручності та для залучення нових клієнтів.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ ПРОЄКТУВАННЯ ІНТЕРАКТИВНОГО САЙТУ «KINOTALE» ДЛЯ ПЕРЕГЛЯДУ І СОРТУВАННЯ ФІЛЬМІВ

1.1 Аналіз еволюції кінотеатрів

Культура та мистецтво завжди відігравали важливу роль у житті людини. Фільми також являються різновидом мистецтва, але як такий фільм нам не був відомий із самого початку. Спочатку це були книжки і театри, аж допоки із розвитком технологій не було створено кінематограф. І в подальшому майбутньому разом із людством розвивалися і технології, і кіно, і спосіб його перегляду. Спочатку світ зустрів звичайний театр, де потрібно було приходити в певне місце в певний час, з довгим очікуванням і великими паузами, адже грали у ньому живі люди яким був потрібен час на переодягання, зміну локацій тощо. І такі прості речі, як зупинити перегляд і відкласти його на потім, чи перемотати вперед чи назад акторів було неможливо, а це сильно би зекономило багато часу. Але 1895 став початком вирішення цієї проблеми. Так офіційно день 28 грудня 1895 року вважається днем народження такого явища як кіно. Ще близько ста років кіно буде проходити неймовірний розвиток. Так як перші кадри це були пришвидшені фото, з часом це були німі чорно-білі фільми і навіть пройшовши такий шлях до початку онлайн кінотеатрів ще досить далеко. Процес створення фільму був схожий на театр тільки з поправкою на запис процесу на плівку. І після цього було достатньо цей запис відзнятого фільму прокручувати стільки разів скільки було потрібно, і навіть якщо це потребувало транслювання фільму одночасно в декількох кінотеатрах. З часом на заміну плівці прийшли електронно-цифрові носії що значно полегшило роботу.

Для перегляду фільму в кінотеатрі, дуже часто потрібно моніторити вільні години, вистоювати довгі черги за для покупки квитка, а в дні прем’єри фільму не завжди вдається купити білет із зручним місцем. Також одною із

класифікацією кінотеатрів є приміщення і кількість людей яку можна в собі вмістити. Доволі часто власники задля заробітку хочуть вмістити якомога більше людей і від того страждає якість стільців. Ще одним мінусом може бути те, що від великої кількості людей особливо у дні прем'єри, важко сконцентруватися на перегляді фільму через навколишній шум. Також, через ті чи інші причини ми не завжди можемо дозволити собі похід в кінотеатр чи покупку диска, але із появою і розвитком всесвітньої павутини нам став доступний ще один варіант перегляду фільмів. Тепер є можливим дивитися фільми онлайн абсолютно безкоштовно, хоча існують і платні версії та преміум підписки.

Прогрес не заставив себе довго чекати і усвідомлення того, що перегляд фільмів можна зробити ще більш приємним і ще більш зручним, призвів до появи онлайн кінотеатрів.

Одною з переваг дивитися фільми онлайн є те, що вам не потрібно зберігати кінострічку і тим самим засмічувати дисковий простір на своєму персональному комп'ютері. Для того щоб подивитися фільм більше не потрібно шукати вільні години витрачати час на дорого та черги, потрібно лише увімкнути ПК зайти на сайт найзручніший для власного використання та знайти потрібний фільм. Також для ще зручнішого перегляду можна вибрати якість фільму та озвучку мовою країни якою ви бажаєте продовжити перегляд. Зручність також полягає і у тому що фільм можна пришвидшити, зупинити перегляд і повернутися до нього згодом або ж просто зробити голосніше чи тихіше і все це не витрачаючи лишній час та гроші [2].

Після перегляду фільму на сайті де був здійснений перегляд дуже часто є можливість поставити коментар та оцінку картині, щоб люди які зайдуть на сайт після вас також змогли оцінити чи варто витрачати свій час на перегляд саме цієї кінострічки. Невеличким бонусом є і те що сайт який вам сподобався можна поділитись з друзями. Ще однією відмінністю кінотеатру від онлайн кінотеатру є те, що в першому випадку ви переміщаєтесь для перегляду, а в другому онлайн кінотеатр переміщається за вами.

1.2 Постановка завдання розробки інтерактивного сайту «Kinotale»

Ціллю даної кваліфікаційної роботи являється створення web-сайту для перегляду фільмів в онлайн режимі.

Повне найменування кваліфікаційної роботи – «Розробка інтерактивного сайту «Kinotale» для перегляду і сортування фільмів на основі технології vue.js, node.js та mongoDB».

Стисле найменування для web-сайту – «Kinotale».

Даний сайт призначається для людей, які хочуть відпочити за переглядом чудового фільму на свій вибір, не витрачаючи при цьому багато часу на похід до кінотеатру. Програмний продукт також є безкоштовним, що дозволяє зекономити не лише час, а й гроші.

Для постановки задач, які повинен містити у собі web-сайт потрібно спочатку переглянути і проаналізувати вимоги, які є поставлені для такого роду проєктів.

Виготовлений сайт повинен виконувати такі задачі:

– реєстрація;
– вхід на сайт;
– пошук фільмів;
– навігація по сайту;
– бд з фільмами;
– можливість перегляду фільму в хорошій якості з бажаною озвучкою;
– можливість залишити відгук про переглянутий фільм;
– можливість вибрати фільм, який сподобався користувачу у вкладку «вибране»;

Для того, щоб на завершенні отримати якісний web-сайт, спочатку необхідно правильно поставити задачу і спроєктувати всі необхідні дії для подальшого проєктування. Це одна з важливих умов для створення сайту у

майбутньому, проєктування мусить містити в собі зрозумілу структуру та зручну навігацію, але навіть при точні постановці задачі для написання сайту ціль може бути не досягнута. Але якщо не дотримуватися запланованих дій, то можливість створення нікому не зрозумілого продукту дуже велика.

План дій написання продукту потрібно розробляти докладніше, для того щоб під час створення програми не спостерігалися «стрибки» між різними етапами: від написання коду – до розробки архітектури, від розробки архітектури – до опису концепції, від опису концепції – до написання коду. В результаті таких дій з’являється великий ризик втрати логіки сайту під час цих стрибків [3].

Окрім цього, при постійних переходах між різними етапами можна з легкісттю значно погіршити якість результату роботи. Такі «стрибки» призведуть до того, що розробники будуть змушенні переробляти одну і ту саму роботу по декілька разів. А це в свою чергу негативно впливатиме на термін виготовлення проєкту та на його вартість.

Для того, щоб уникнути такої проблеми, потрібно заздалегідь поставити правильно задачу і спроєктувати всі дії та чітко виконувати поставлені планом задачі. Сьогодні існує безліч стандартних моделей проєктування, які дозволяють поетапно, крок за кроком реалізувати будь-який проєкт від ідеї до її втілення.

Після проходження всіх етапів описаних вище, було проаналізовано поставленні задачі та для їх успішного виконання вибрано мову програмування javascript та node.js із фреймворком vue.js, та mongoDB для роботи з базаю даних.

1.3 Формулювання вимог до інтерактивного сайту для перегляду і сортування фільмів

Вимоги до програмного забезпечення – це такий набір вимог на рахунок якості, властивостей та функцій програмного забезпечення, яке має бути розроблене або вже знаходитися на стадії розробки.

Які саме повинні бути вимоги визначається в процесі аналізу вимог, а також фіксуються в специфікації вимог, діаграмах прецедентів та інших артефактах процесу аналізу та розробки вимог (див.рис.1.1).

Розробку вимог для ПЗ можна розділити на декілька етапів:

– знаходження вимог;

– аналіз вимог;

– специфікація ;

– тестування вимог.

Також існує три рівні вимог:

– Бізнес-вимоги – саме призначення програмного продукту, хто і чому буде користуватися цим продуктом, його собівартість.

– Вимоги користувача – визначається які завдання може виконувати користувач на сайті, з якими проблемами він може зіткнутися і їх сценарії вирішення. Такі вимоги можуть бути у вигляді тверджень, варіантів використання, історії користувача та сценаріїв взаємодії.

– Функціональні вимоги – яку саме функцію повинен виконувати програмний продукт.

Аналіз вимог включає в себе три види діяльності:

– Виявлення вимог: задача комунікації з користувачами для визначення їх вимог. Збір інформації про цільову аудиторію для того щоб краще розуміти що саме потребують користувачі. Також це називають збором вимог.

– Аналіз вимог: після виявлення вимог потрібно їх проаналізувати для виявлення недоліків (неточностей, неповноти, неоднозначностей чи суперечностей) і їх виправлення.

– Запис вимог: Вимоги можуть документуватись в різних формах, таких як опис звичайною мовою, прецедентами, користувацькими історіями, чи специфікаціями процесу [4].

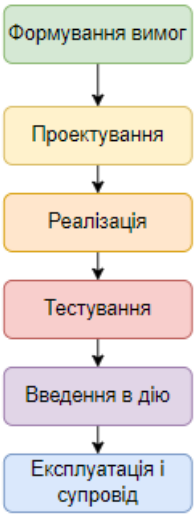

Рисунок 1.1 – Модель життєвого циклу [5]

Аналізування вимог – це довгий та вкрай важкий процес що вимагає застосування тонких психологічних навичок. Нові системи змінюють середовище і відношення між людьми, тому важливо розпізнати всі зацікавлені сторони, взяти до уваги всі їхні потреби, і переконатись що вони розуміють наслідки які приносить нова система [6].

Таким чином було сформовано наступні вимоги для кіно-сайту. При заходженні на сайт користувач чітко повинен бачити основні функції програмного продукту, простий і чіткий інтерфейс і зручність у користуванні. Також на сайті повинна бути зручна пошукова система по базі даних з найкращими фільмами. Ще одною із вимог повинно бути при відвідуванню сайту, можливість зареєструватися для того щоб отримувати сповіщення

виходу нового фільму чи зберігання його у папку вибране, з подальшою можливістю заходити на сайт під своїм логіном на паролем.

1.4 Проєктування сайту для перегляду і сортування фільмів

Проєктування програмного забезпечення – це робота у межах життєвого циклу, де достеменно проходить аналіз вимог для того щоб створити внутрішню структуру програмного забезпечення за допомогою детально створеного опису, що являється основою для побудови ПЗ як такого. Також це процес під час якого визначається тип архітектури, компоненти, побудова інтерфейсу, створення алгоритмічного процесу, структури даних, системи чи компонентів ПЗ та всіх можливих атрибутів які знадобляться для майбутнього програмного забезпечення.

Проєктуванню підлягають:

– інтерфейс користувача;
– компоненти ПЗ;
– архітектура ПЗ.

Для проєктування програмного продукту використовують різного роду моделі, а саме їх основні різновидності – блок-схеми, ER-діаграми, DFD та інші.

Блок-схема – це побудова алгоритму у графічному вигляді, де зображено алгоритм виконання та послідовність дій у програмі за допомогою блоків, з'єднаних між собою стрілками, в середині блоків також може бути записаний їх короткий опис [7].

DFD – це графічна презентація «потоків» даних в інформаційній системі, також ця діаграма може бути застосована для візуального представлення процесу обробки даних [8].

ER-діаграма – це найпоширеніша модель для проєктування та розробки програм та бази даних, різних систем та досліджень. Діаграма має вигляд блок-схеми та використовує геометричні символи для побудови, відображає

взаємозв’язки об’єктів, відносин та їх атрибутів. Цей вид діаграми тісно пов’язаний з структурою даних DSD, фокус полягає на відносини елементів всередині сутностей. Для відображення інформації процесів чи систем схеми ER-діаграми часто працюють разом із потоками даних DFD [9].

Після детального ознайомлення з різновидностями діаграм та їх можливостями для проєктування сайту було обрано модель ER-діаграми (див.рис.1.2).

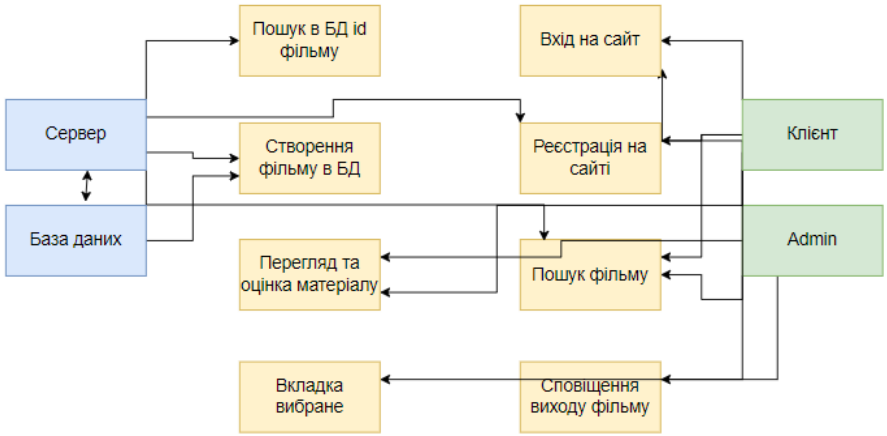

Рисунок 1.2 – Побудова можливостей роботи сайту та взаємозв’язки між ними

У цьому програмному продукті важливу роль відіграє робота з базою даних, сервером на користувачем (див.рис.1.3).

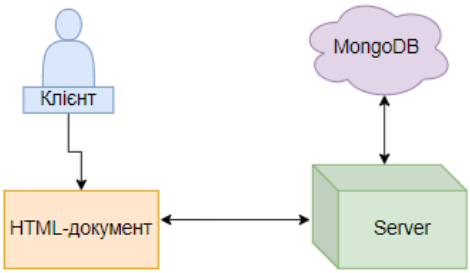

Рисунок 1.3 – Схема відправки та отримання даних між користувачем, сервером та базою даних

Веб-сервер – це одна з основних частин всесвітньої павутини, він приймає http-запит від клієнта, і видає назад http-відповіді, зазвичай це HTML-сторінка, файл, медіа, зображення чи будь-які інші дані [10].

Веб-сервером називають як програмне забезпечення, що виконує всі функції веб-сервера, так і комп'ютер, на якому це програмне забезпечення працює.

Клієнти дістаються веб-сервера за URL-адресою потрібної їм веб-сторінки або іншого ресурсу.

Сервер складається з таких частин як:

– фізичний сервер;

– операційна система сервера;

– програмне забезпечення для полегшення зв’язку http [11].

Клієнт – це, як правило, браузер, який передає серверу запити на отримання різного роду інформації позначених URL-адресами.

Зазвичай база даних показана у вигляді таблиць, що складаються із стовпців і рядків, як це зроблено у MySQL. Та для роботи над кваліфікаційною роботою було обрано mongoDB. Ця база даних не потребує опису схеми таблиць. Система є документно-орієнтованою та містить в собі відкритий вихідний код, а також підтримує зберігання документів в JSON-подібному форматі.

Суть такої системи полягає в тому, що коли клієнт відправляє запит на сервер, там він проходить процес обробки, і готовий результат відправляється клієнтові. Сервер також може обслуговувати кілька клієнтів одночасно. Часто так стається що одночасно приходить більше одного запиту, у такому випадку вони встановлюються в чергу і виконуються сервером послідовно. Іноді запити можуть мати пріоритети. Запити з більш високими пріоритетами повинні виконуватися раніше.

Сервер баз даних виконує обслуговування та управління базою даних та відповідає за цілісність та збереження даних, а також забезпечує операції введення-виведення при доступі клієнта до інформації [12].

Архітектура клієнт-сервер складається з клієнтів та серверів. Основна ідея полягає в тому, щоб розміщувати сервери на потужних машинах, а додаткам, що використовують мовні компоненти СКБД, забезпечити доступ до них з менш потужних машин-клієнтів за допомогою зовнішніх інтерфейсів.

Функції, які реалізуються на сервері:

– зберігання, доступ, захист і резервне копіювання даних;
– обробка клієнтського запиту;
– відправлення відповіді клієнту.

Функції, які реалізуються на стороні клієнта:

– надання користувальницького інтерфейсу;
– формулювання запиту до сервера і його відправка;
– отримання відповіді, відправка додаткових команд таких як запити на додавання, оновлення або видалення даних [13].

1.5 Прототипування сайту

Створення прототипу для програмних продуктів, додатків на телефон чи будь-яких інших інтернет-сервісів – це основний етап розробки яким не потрібно нехтувати, в майбутньому це допоможе значно з економити свій час.

Вигляд того, як саме повинен виглядати кінцевий продукт повинен чітко пройти всі етапи проєктування, адже уява того що має вийти в кінцевому етапі, в процесі розробки може значно змінитися, а також програміст може відхилитися від плану, якщо його не чітко поставити, тому прототип дозволяє чітко візуально бачити весь інтерфейс та поставлене технічне завдання і якщо знадобиться то на ранній стадії вносити зміни до продукту.

Прототипування важливе також для того щоб краще зрозуміти побажання замовника. На самих перших стадіях задача представлена у письмовому вигляді чи навіть усному. Зі створеним прототипом можливо наочно все продумати і оформити замовлення у цифровий продукт, що допоможе покращити усі не точності. Ще є плюсом те, що переробити прототип значно легше ніж готове

ПЗ. Іншими словами, що будь-яке цифрове рішення легко відобразити не тільки у вигляді програмного коду або виконуваних файлів, а й в графічному поданні – у вигляді прототипу продукту. Останній являє собою набір наочних блок-схем, діаграм переходів, логічних елементів інтерфейсу і макетів дизайну.

Проєктування програмного забезпечення має подібну аналогію до проєктування макетів будинків, це дає можливість чітко візуально подивитися на готову архітектуру програмного продукту, внаслідок чого програмістів, які проєктували програмне забезпечення, часом називають програмними або системними архітекторами.

Після чіткого розуміння кінцевого продукту і всіх його технічних процесів, які воно має виконувати, розробникам стає простіше зробити вибір на користь вибору платформи та технології які будуть потрібні для роботи. Якщо чітко бачити прототип програми, то це дає можливість вирішення багатьох економічних питань, часові рамки на виконання розробки, та кількість працівників яких потрібно залучити до проєкту і їх завантаження.

1.6 Висновок до першого розділу

У першому розділі кваліфікаційної роботи бакалавра розглянуто еволюцію кінотеатрів, від перших театрів до онлайн кінотеатрів та їх аналіз.

Проаналізовано предметну область та висунуто вимоги, яким має задовольняти інтерактивний сайт та внаслідок цього поставлено задачі для його виконання.

Для кращого розуміння побудови веб-сайту пройдено та висвітлено в розділі етапи проєктування та прототипування.

РОЗДІЛ 2. РОЗРОБКА ІНТЕРАКТИВНОГО САЙТУ «KINOTALE» ДЛЯ ПЕРЕГЛЯДУ І СОРТУВАННЯ ФІЛЬМІВ

2.1 Технології для створення інтерактивного сайту «Kinotale»

Інженер Тім Бернс-Лі у 1991 році для того, щоб показати світу перший сайт, створив просту сторінку наповнену інформацією за допомогою мови розмітки HTML чим дав певний старт для розвитку технологій [14].

За ці роки кількість сайтів вже неможливо підрахувати, зараз вони мають різну складність і тематику. Також з’явилася велика кількість мов програмування і технологій за допомогою яких можна створити будь який програмний застосунок.

Так як з’являлись різновиди технологій так і люди відкрили для себе нові різновиди професій. Більшість населення використовує всесвітню мережу Internet як пошукову систему чи спосіб спілкування, але є і ті, які відкрили для себе більші можливості – програмування.

Ця робота потребує постійного розвитку для того, аби найти спосіб вирішити проблеми людей і щоразу все новішими і новішими методами. В наслідок цього мови програмування постійно і швидко оновлюються, зараз мало не що дня з’являється щось нове, нові технології, нові методи вирішення проблем. І для того, щоб програмний засіб залишався швидкісним і зручним у використанні завжди потрібно слідкувати за оновленнями та новинками.

2.1.1 Фреймворк Vue.js для перегляду та сортування фільмів

Для того, щоб код був унікальний і легкий для перегляду і розуміння іншими розробниками, програмісти почали полегшувати собі роботу фреймворками. Але часто при довготривалій роботі кодова база сильно збільшується у розмірі в деяких фреймворках і вони починають вносити труднощі у проект. Тому в наслідок цього перед розробкою потрібно врахувати

два фактори: складність програми та складність фреймворка. І для того, аби написати інтерактивний сайт для перегляду фільмів ідеально підходить vue, він є простим у використанні і код буде точний і зрозумілий.

Варто сказати, що vue це фреймворк для java-script. Java-script – це мова програмування, за допомогою якої можна створити веб-додаток, а також ця мова дозволяє зробити сторінку інтерактивною чим дозволяє реагувати на дії користувача.

Самого java-script замало для написання кіно-сайту чи будь якого іншого сучасного сайту, він потребує додаткових фреймворків. Найбільш популярні це React, Angular та Vue. Мій вибір зупинився на останньому Vue.

Vue – це сучасний фреймворк для побудови користувацького інтерфейсу. Його створено придатним для поступового впровадження на відміну від фреймворків монолітів. Ядро vue насамперед вирішує задачі рівня представлення, що значно полегшує інтеграцію з іншими бібліотеками та існуючим програмним забезпеченням. Водночас цей фреймворк також повністю підходить для програмування складних одно-сторінкових програм такого типу як SPA, Single-Page Application, якщо використати його із додатковими бібліотеками та новітніми інструментами [15].

Підключити vue можна декількома способами, один з яких використати cdn-скрипт, але цей варіант підійде не для всього, лише для простих проектів. Для маштабного проекту у вигляді кіно-сайту використано npm та інструмент командного рядка cli. Ці технології найкраще підходять для роботи, при встановлені шаблона cli лише за декілька хвилин можна отримати працюючу конфігурацію з гарячим перезавантаженням модулів, лінтингом коду при збереженні та настроєною конфігурацією production-складання.

2.1.2 Платформа Node.js для написання серверної частини сайту «KinoTale»

Node.js – платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript. Засновником платформи є Раян Дал. Раніше в Javascript для того щоб обробити дані потрібно було використовувати браузер користувача, то node.js взяв на себе серверну роботу, він виконує javascript-скрипти на сервері і після опрацювання вертає результат користувачеві його запиту. Платформа Node.js перетворила JavaScript на мову загального використання з великою спільнотою розробників.

Node.js має наступні властивості:

– асинхронна одно-нитєва модель виконання запитів;
– неблокуючий ввід/вивід;
– система модулів CommonJS;
– рушій JavaScrirt Google V8.

Node.js був написаний Раяном Далом у 2009 році. А вже у січні 2010 року для середовища Node.js був введений пакетний менеджер під назвою npm. Менеджер пакетів полегшує програмістам публікацію та обмін сирцевим кодом бібліотек Node.js та застосовується для простішого встановлення, оновлення та видалення бібліотек [16].

Платформа Node.js застосовується для вокористання високопродуктивних мережевих застосунків, написаних мовою програмування javascript. Платформа чудово виконує не лише серверні скрипти для веб-запитів, а й застосовується для створення клієнт-серверних програм.

Для того щоб працювати з платформою її спочатку потрібно встановити на комп’ютер і вже після цього можна додавати всі потрібні бібліотеки та скрипти для роботи.

2.1.3 База даних MongoDB

MongoDB – документо-орієнтована система керування базами даних (СКБД) з відкритим вихідним кодом, для якої не потрібно опис схем таблиць. MongoDB займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СКБД, функціональними і зручними у формуванні запитів.

MongoDB підтримує зберігання документів в JSON-подібному форматі, має досить гнучку мову для формування запитів, може створювати індекси для різних збережених атрибутів, ефективно забезпечує зберігання великих бінарних об'єктів, підтримує журналювання операцій зі зміни і додавання даних в БД, може працювати відповідно до парадигми Map/Reduce, підтримує реплікацію і побудову відмовостійких конфігурацій. У MongoDB є вбудовані засоби із забезпечення шардінгу (розподіл набору даних по серверах на основі певного ключа), комбінуючи який з реплікацією даних можна побудувати горизонтально масштабований кластер зберігання, в якому відсутня єдина точка відмови (збій будь-якого вузла не позначається на роботі БД), підтримується автоматичне відновлення після збою і перенесення навантаження з вузла, який вийшов з ладу. Розширення кластера або перетворення одного сервера на кластер проводиться без зупинки роботи БД простим додаванням нових машин.

Основні можливості MongoDB:

– документо-орієнтоване сховище (проста та потужна JSON-подібна схема даних);
– досить гнучка мова для формування запитів;
– динамічні запити;
– повна підтримка індексів;
– профілювання запитів;
– швидкі оновлення «на місці»;

– ефективне зберігання бінарних даних великих обсягів, наприклад, фото та відео;

– журналювання операцій, що модифікують дані в БД;

– підтримка відмовостійкості і масштабованості асинхронна реплікація, набір реплік і шардінг;

– може працювати відповідно до парадигми MapReduce.

На рисунку 2.1 зображено один з фільмів який міститься в базі даних MongoDB Compass.

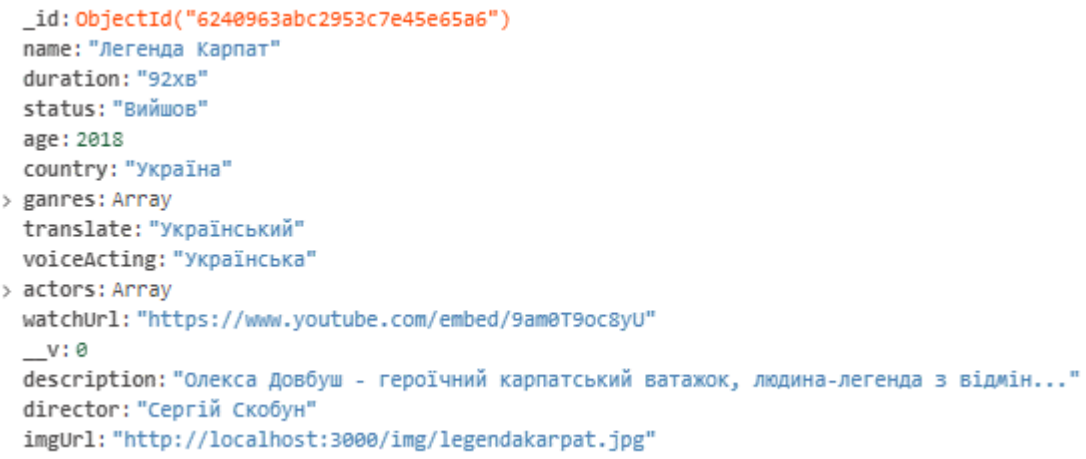

Рисунок 2.1 – Фільм, що міститься в базі даних

На рисунку 2.2 зображено список зареєстрованих користувачів в базі даних з їх закодованими паролями.

Рисунок 2.2 – Список зареєстрованих користувачів

Compass – це інтерактивний інструмент для запитів, оптимізації та аналізу даних MongoDB. Саме цей інструмент використовується для зберігання всіх фільмів, інформацію про них та користувачів [17].

2.2 Практична реалізація кваліфікаційної роботи

Для того, щоб реалізувати програмне забезпечення «KinoTale» потрібно обрати програмне середовище для розробки, і після вибору всіх технологій та огляду їхніх можливостей було обрано інтегроване середовище розробки яке найкраще для цього підходить – Visual Studio Code.

Visual Studio Code – програмне середовище, яке дозволяє створювати, редагувати на налагоджувати сучасні веб-програми та програми для хмарних систем. Засіб являється абсолютно безкоштовним, ним може скористуватися будь-яка людина, також він доступний для операційних систем Windows, Linux та OSX.

Розробником Visual Studio Code є компанія Microsoft, яка презентувала середовище у 2015 році. Середовище розробки з класу Visual Studio стало першим кросплатформних.

За основу для Visual Studio Code використовуються напрацювання вільного проєкту Atom, що розвивається компанією GitHub. Зокрема, Visual Studio Code є надбудовою над Atom Shell, що використовує браузерний рушій Chromium і Node.js [18].

Редактор містить у собі вбудовану систему налагодження програм, інструменти для роботи з Git та процес редагування програмного коду і його навігація, автодоповнення типових конструкцій і контекстної підказки. Середовище також підтримує розробку для платформ ASP.NET і Node.js, дозволяє обійтися без повного інтегрованого середовища розробки. Visual Code підтримує безліч мов програмування серед яких є і JavaScript.

2.2.1 Серверна частина

На рисунку 2.3 зображено перелік файлів та папок серверної частини кваліфікаційної роботи.

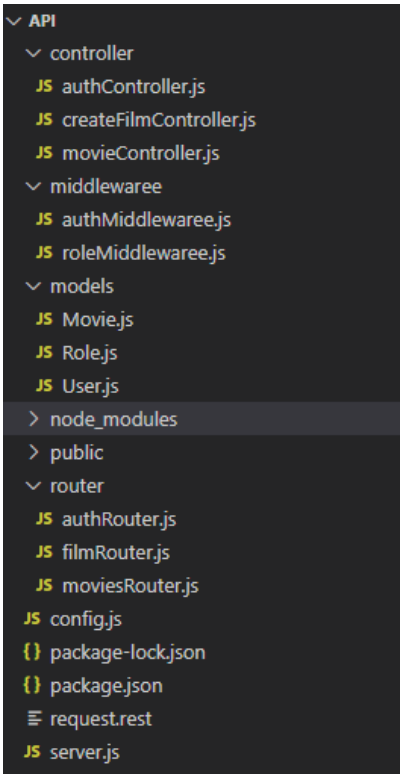

Рисунок 2.3 – Перелік папок та файлів API

Для написання серверної частини сайту було створено окремий документ API для частини бекенду.

Файл server.js – файл, у якому було створено сервер, на якому відбуваються всі операції між користувачем та сервером. Сервер стартує на localhost:3000. А також підключення до бази даних MongoDB за допомогою mongoose на стандартному хості localhost:27017.

Файл package.json – створюється автоматично та містить у собі залежності, команди, версії та інформацію про створений проект. Також за допомогою командного рядка можна встановити і інші залежності які потрібні для роботи.

Файл config.js модульний експортер секретного ключа по якому розшифровується токен.

Для функціоналу реєстрації і подальшого входу на сайт було створено папки controller, middleware, models та router. В цих папках також містяться файли для доступу до фільмів і їх запити.

Файл authRouter.js – це файл у якому визначаються маршрути по яким пізніше ці маршрути будуть визначатися. Також тут прослуховуються запити такі яка get, post та є можливим і прослуховування інших http-запитів. У файлі відбуваються запити саме на реєстрацію та вхід.

Файл authController.js – містить у собі весь ключовий функціонал реєстрації та входу на сайт, взаємодію з користувачем.

User.js та Role.js схеми того як виглядає сутність користувача та ролі цього користувача в базі даних.

Файл authMiddlewaree.js дає доступ до тих чи інших функцій тільки зареєстрованим користувачам.

Файл roleMiddlewaree.js дає доступ до можливостей лише адміністратору.

Файл filmRouter.js – це файл у якому визначається маршрут на створення фільмів.

Файл moviesRouter – файл у якому визначається маршрут на отримання фільмів з бази даних.

Файл Movie.js – схема того як виглядає сутність фільмів, їхній тип та властивості.

Файл movieController – описує функціонал отримання фільмів з бази даних.

Файл createFilmController.js – отримавши інформацію з http-запиту схеми, записує фільм в базу даних.

Папка public містить в собі папку img з вмістом картинок до фільмів.

2.2.2 Частина опису фронтенду проєкту

На рисунку 2.4 зображено перелік файлів та папок частини фронтенду кваліфікаційної роботи.

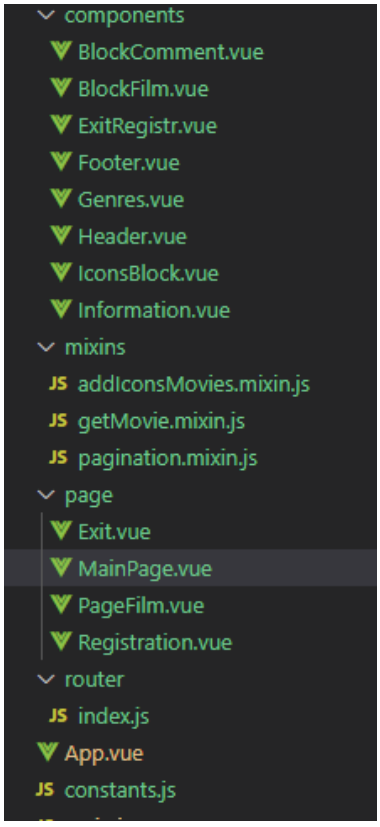

Рисунок 2.4 – Перелік папок та файлів «KinoTale»

У документі KinoTale описується весь фронтенд кваліфікаційної роботи.

У папці page містяться основні сторінки:

MainPage.vue – це сама основна сторінка, яка містить усі компоненти програми.

PageFilm.vue – це сторінка фільму при його виборі та завантаженні, на ній міститься сам фільм і інформація про нього.

Registration.vue – сторінка реєстрації на сайті.

Exit.vue – сторінка входу на сайт.

У папці router у файлі index.js розписані всі переходи між сторінками та їхні адреси хостів.

Файл constans.js експортує константне значення хоста сервера.

Файл main.js імпортує роути та vue.

У папці components містяться усі компоненти ключових сторінок:

На рисунку 2.5 зображено шапку сайту

Рисунок 2.5 – Компонент Header.vue

На рисунку 2.6 зображено іконки фільмів

Рисунок 2.6 – Компонент IconsBlock.vue

На рисунку 2.7 зображено жанри.

Рисунок 2.7 – Компонент Ganres.vue

На рисунку 2.8 зображено інформацію про сайт

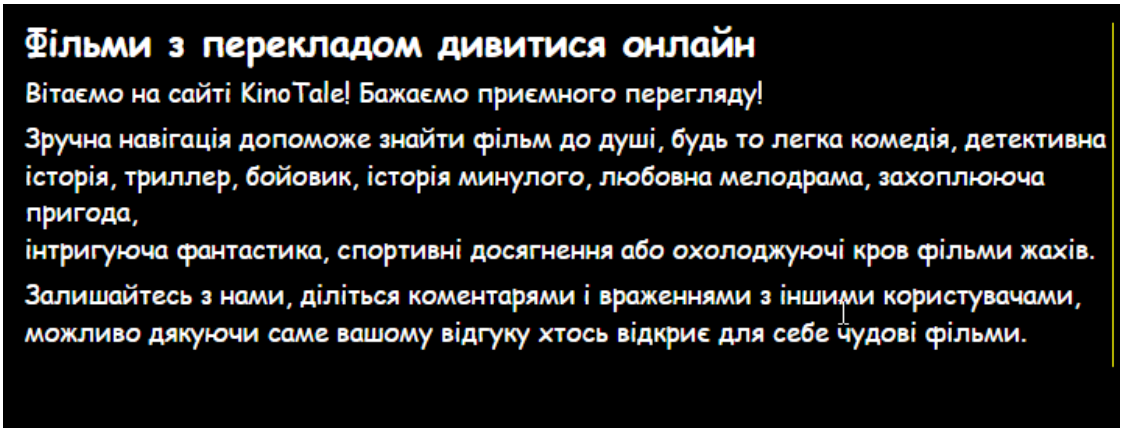

Рисунок 2.8 – Компонент Information.vue

На рисунку 2.9 зображено блок з клавішами за допомогою яких можна здійснити реєстрацію або якщо користувач вже зареєстрований то здійснити вхід на сайт. А також посилання на вибрані фільми чи прем’єри фільмів.

Рисунок 2.9 – Компонент ExitRegistr.vue

На рисунку 2.10 зображено один із блоків з фільмами

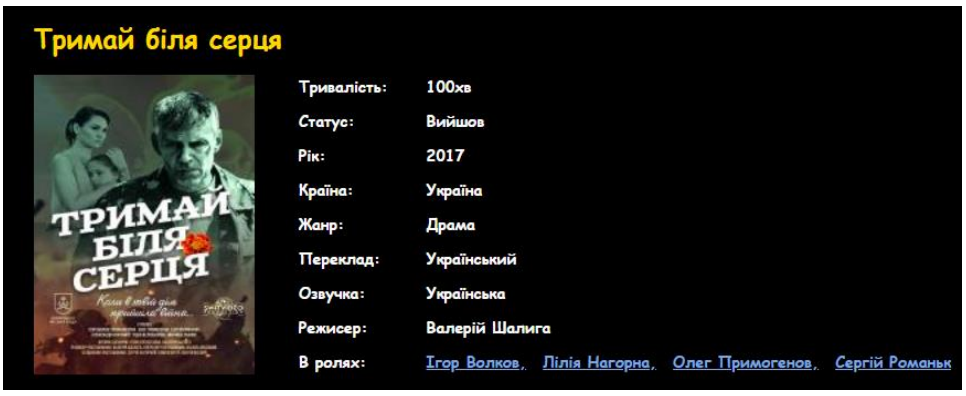

Рисунок 2.10 – Компонент BlockFilm.vue

На рисунку 2.11 зображено вікно коментарів

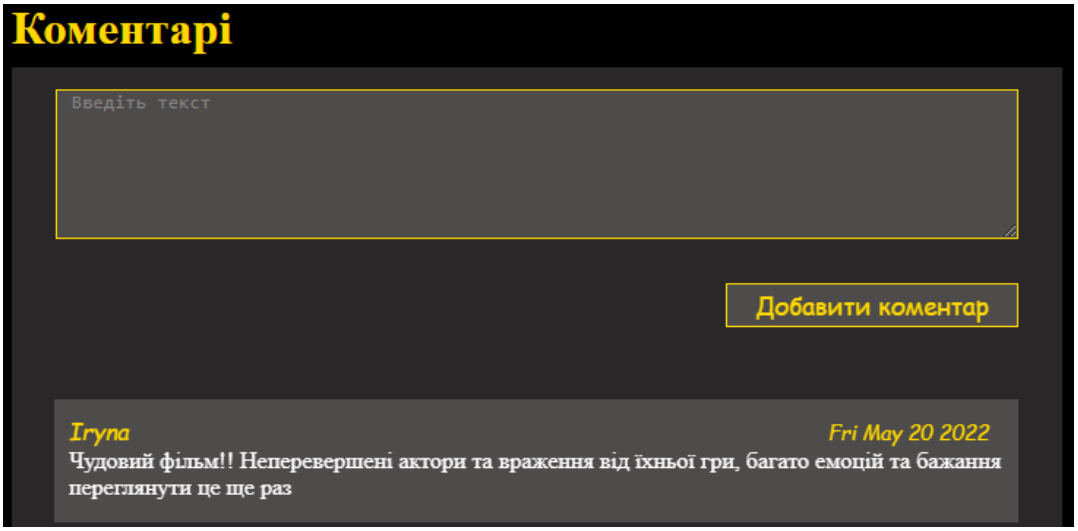

Рисунок 2.11 – Компонент BlockComment.vue

На рисунку 2.12 зображено навігацію сторінок та низ сайту

Рисунок 2.12 – Компонент Footer.vue

У папці mixins знаходяться міксини для полегшення роботи і для оригінальності коду без повторень. Файл addIconsMovies.mixin.js перевіряє url-адресу картинки фільму та добавляє його до блоку IconsBlock. Файл pagination.mixin.js потрібен для пагінації сторінок, зручний перехід між сторінками з фільмами. У папці assets знаходяться іконки до сайту.

2.3 Оцінка одержаних результатів кваліфікаційної роботи «KinoTale»

Під час написання кваліфікаційної роботи «Розробка інтерактивного сайту «Kinotale» для перегляду і сортування фільмів» було досягнуто поставлених задач.

Відвідавши сайт користувач має змогу знайти фільм до душі у цьому йому допоможе пошукова система у шапці сайту. Зручна навігація дозволить йому обрати жанр до вподоби для перегляду. Також з поставлених задач було зробити вкладку «список обраного» та вкладку «прем'єри» з інформацією про прем'єри кінострічок.

На сайті можна здійснити реєстрацію для кращого подальшого користування, наприклад для того щоб зберегти фільми у вкладку «вибране» потрібно спочатку зареєструватися на сайті.

Після перегляду кінострічки користувач, якщо він є зареєстрований, може залишити оцінку фільму та залишити відгук про нього, що дасть змогу іншим користувачам краще розуміти чи зацікавить картина при перегляді.

2.4 Висновки до другого розділу

В ході роботи другого розділу внаслідок висунутих раніше вимог вибрано середовище розробки та технології, які використовувалися при виконанні завдань проєктуванні кваліфікаційної роботи .

У наступних параграфах описуються технології та мови програмування, також йде опис дерева папок у коді.

Під час роботи створено базу даних за допомогою MongoDB та Node.js., яка містить у собі всю інформацію про фільм, користувачів, їхні коментарі та уподобання.

Також створено фільтрацію фільмів по акторах та жанрах з ширшим функціоналом для зареєстрованих користувачів.

РОЗДІЛ 3. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1 Вплив діяльності людини на довкілля

Господарська діяльність людини зумовила пошкодження і вичерпування природних ресурсів, що призводить до деформації сформованих протягом багатьох мільйонів років природного кругообігу речовин та енергетичних потоків на планеті. Внаслідок цього почалося прогресуюче руйнування біосфери Землі, що може набути характеру незворотних процесів і навколишнє середовище може стати непридатним для існування.

В XXI столітті навколишнє середовище нашої планети неухильно погіршується внаслідок антропогенного впливу. Люди вже не спроможні адаптуватися до цих швидких глобальних змін. Крім того, постала проблема демографічного вибуху і обмеженості природних ресурсів та життєвого простору земної кулі [19].

В умовах науково-технічного прогресу значно ускладнились взаємовідносини суспільства з природою. Людина отримала можливість впливати на хід природних процесів, підкорила сили природи, почала опановувати майже всі доступні відновні і невідновні природні ресурси, але разом з тим забруднювати і руйнувати довкілля.

. Ноосфера - сучасна стадія розвитку біосфери, пов'язана з появою в ній людства. Частина планети і навколопланетного простору зі слідами діяльності людини. Відповідно до оригінальної теорії Вернадського, ноосфера є третьою у послідовності таких основних фаз розвитку Землі як утворення геосфери(неживої природи) та біосфери (живої природи).

При добуванні корисних копалин людина проникає у надра землі і тим змінює довкілля, втручається у природні процеси, що відбуваються на землі. Сучасна техніка зробила людину такою могутньою, що природа в багатьох

випадках поступається людині у здатності до зміни ландшафту і рельєфу поверхні. Кар'єри довжиною до 10 км і глибиною до 1000 м, «гори» породних відвалів, терикони біля шахт, басейни-сховища дрібних відходів площею в декілька квадратних кілометрів - усе це результат людської діяльності. Внаслідок гірничотехнічної діяльності в світі порушено не менше 15-20 млн га земель, з них 59 % площі використано під різні гірничі виробки, 38 % - під відвали пустої породи або відходи збагачення, 3 % - місця осідання, провалів, порушень поверхні, пов'язаних з підземними розробками. Обсяг відвалів порід, що утворилися, і виробничих відходів становить понад 2000 км³. Для отримання мінеральної сировини і палива людство вимушене використовувати дедалі глибші шари земної кори (золоторудні шахти Південно-Африканської Республіки, досягли позначок 3-4 км нижче земної поверхні; амплітуда висот між дном найглибших кар'єрів і поверхнею найвищих відвалів перевищує 1100 м) [20]. Внаслідок переміщення великих обсягів гірничої маси погіршується режим ґрунтових і підземних вод, змінюються поверхневий водостік і структура ґрунту, інтенсифікується ерозійна робота води і вітру, що в деяких випадках спричиняє зміну клімату в районі ведення гірничих робіт. Гірничодобувні роботи супроводжуються штучним водозниженням. Скидання стічних вод, що відкачуються при проведенні гірничих робіт, веде до забруднення поверхневих водних об'єктів різними солями, нафтопродуктами та важкими металами. Зсуви гірських порід на територіях, що підробляються, осідання поверхні, розсіювання породи з відвалів негайтивно впливають на стан земельних ресурсів. Значні надходження забруднюючих речовин відбуваються в зонах комунікацій і транспортних вузлів (90 т пилу на 1 км залізничного полотна на рік). При експлуатації нафтопроводів та продуктопроводів найбільшої шкоди довкіллю завдають аварійні витоки нафти, суспензій.

Одна з головних екологічних проблем пов'язана з погіршенням стану земельних ресурсів. За історичний час внаслідок прискореної ерозії, дефляції та інших негативних процесів людство втратило майже 2 млрд га продуктивних

земель. До утворення пустель схильна площа в 4,5 млрд га, на якій проживає близько 850 млн чол. Пустелі швидко розвиваються (до 5-7 млн га на рік) у тропічних районах Африки, Азії і Америки, а також в субтропіках Мексики.

Швидкість зникнення лісів становить 6-20 млн га на рік [21]. Важлива для людства проблема - охорона геологічного середовища верхньої частини літосфери, яка розглядається як багатокомпонентна динамічна система, що перебуває під впливом інженерно-господарської діяльності людини. Найголовніший компонент геологічного середовища - гірські породи, що містять поряд з твердими мінеральними і органічними компонентами гази, підземні води. Особливо негативно впливають на довкілля техногенні катастрофи, найбільші з яких у останні десятиліття сталися на Чорнобильській атомній електростанції в Україні та аварія на Першій Фукусімській АЕС в Японії.

Знаючи інформацію про вплив діяльності, робимо висновок людського впливу на довкілля, а саме його негативну сторону, а також у підсумку підставивши всі факти використання інтерактивного сайту «KinoTale» для перегляду і сортування фільмів не призведе до таких негативних наслідків для навколишнього середовища.

3.2 Характеристика приміщення при роботі інтерактивного сайту «KinoTale» для перегляду і сортування фільмів щодо небезпеки ураження електричним струмом, пожежній небезпеці, вибухонебезпеці

За ступенем небезпеки ураження електричним струмом усі приміщення поділяються на три категорії: приміщення без підвищеної небезпеки; приміщення з підвищеною небезпекою; особливо небезпечні приміщення.

Приміщення з підвищеною небезпекою характеризуються наявністю в них однієї з таких умов, що створюють підвищену небезпеку: висока відносна вологість повітря (перевищує 75 % протягом тривалого часу); висока температура (перевищує 35 °С протягом тривалого часу); струмопровідний пил;

струмопровідна підлога (металева, земляна, залізобетонна, цегляна); можливість одночасного доторкання до металевих елементів технологічного устаткування чи металоконструкцій будівлі, що з'єднані із землею, та металевих частин електроустаткування, які можуть опинитись під напругою [22].

Особливо небезпечні приміщення характеризуються наявністю однієї з умов, що створюють особливу небезпеку: дуже високої відносної вологості повітря (близько 100 %), хімічно активного середовища; або одночасною наявністю двох чи більше умов, що створюють підвищену небезпеку. Оскільки наявність небезпечних умов впливає на наслідки випадкового доторкання до струмопровідних частин електроустаткування, то для ручних переносних світильників, місцевого освітлення виробничого устаткування та електрифікованого ручного інструменту в приміщеннях з підвищеною небезпекою допускається напруга живлення до 42 В, а в особливо небезпечних приміщеннях - до 12 В.

Безпечна експлуатація електроустановок забезпечується: конструкцією електроустановок; технічними способами та засобами захисту; організаційними і технічними заходами. Ізоляція струмовідних частин забезпечується шляхом покриття їх шаром діелектрика для захисту людини від випадкового доторкання до частин електроустановок, через які проходить струм [23].

Вимоги щодо конструктивних та планувальних рішень промислових об'єктів, а також інших питань забезпечення їхньої пожежо- та вибухобезпеки значною мірою визначаються категорією приміщень та будівель за вибухопожежною та пожежною небезпекою. Відповідно до ДСТУ БВ.1.1.-36:2016 приміщення за вибухопожежною та пожежною небезпекою поділяють на п'ять категорій: А, Б, В, Г і Д. Категорії А і Б класифікуються як вибухопожежонебезпечні, а категорія В як пожежонебезпечна [24].

Категорія А. – (вибухопожежонебезпечна) горючі гази, легкозаймисті рідини з температурою спалаху не більше 28 °С в такій кількості, що можуть утворюватися вибухонебезпечні парогазоповітряні суміші, при спалахуванні котрих розвивається розрахунковий надлишковий тиск вибуху в приміщенні,

що перевищує 5 кПа. Речовини та матеріали, здатні вибухати та горіти при взаємодії з водою, киснем повітря або одне з одним в такій кількості, що розрахунковий надлишковий тиск вибуху в приміщенні перевищує 5 кПа.

Категорія Б. – (вибухопожежонебезпечна) горючий пил або волокна, легкозаймисті рідини з температурою спалаху більше 28 °С та горючі рідини в такій кількості, що можуть утворюватися вибухонебезпечні пилоповітряні або пароповітряні суміші, при спалахуванні котрих розвивається розрахунковий надлишковий тиск вибуху в приміщенні, що перевищує 5 кПа.

Категорія В. – (пожежонебезпечна) горючі та важкогорючі рідини, тверді горючі та важкогорючі речовини і матеріали, речовини та матеріали, здатні при взаємодії з водою, киснем повітря або одне з одним лише горіти за умови, що приміщення, в яких вони знаходяться, або використовуються, не відносяться до категорій А та Б.

Категорія Г. – (помірнопожежонебезпечна) негорючі речовини та матеріали в гарячому, розжареному або розплавленому стані, процес обробки яких супроводжується виділенням променистого тепла, іскор, полум'я; горючі гази, рідини, тверді речовини, які спалюються або утилізуються як паливо.

Категорія Д. – (зниженопожежонебезпечна) негорючі речовини та матеріали в холодному стані. В основу розрахункового методу визначення категорій вибухопожежної та пожежної небезпеки виробничих приміщень покладено енергетичний підхід, що полягає в оцінці розрахункового надлишкового тиску вибуху в порівнянні з допустимим.

Підсумовуючи характеристику приміщення при роботі інтерактивного сайту «KinoTale» для перегляду і сортування фільмів щодо небезпеки ураження електричним струмом, пожежній небезпеці та вибухонебезпеці враховуємо висновок, щоб не стався нещасний випадок у компанії, працівники зобов’язані дотриматися всіх настанов щодо безпеки електротехніки та пожежної безпеки на наявність швидко-займистих речовин та матеріалів.

ВИСНОВКИ

Підсумовуючи результат кваліфікаційної роботи слід зазначити, що для того, щоб сайт працював у повноцінному режимі, варто за допомогою SEO-спеціаліста оптимізувати пошукову систему та провести рекламну інтеграцію, щоб більше людей дізналися про сайт. І зі збільшенням користувачів додатково закупити потужніші сервера.

В першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр»:

– подано постановку завдань для розробки інтерактивного сайту;
– розглянуто прототипування веб-сайту;
– висвітлено формування вимог до сайту;
– проаналізовано вимоги до онлайн кінотеатрів.

В другому розділі кваліфікаційної роботи:

– Розроблено базу даних з фільмами, пошукову систему та фільтрацію фільмів за жанрами та акторами.
– Подано теоретичну частину про технології та платформи, а також середовище розробки сайту та бази даних.

У розділі «Безпека життєдіяльності, основи охорони праці» висвітлено вплив людини на навколишнє довкілля та характеристику приміщення для того щоб убезпечити себе та команду розробки від ураження електричним струмом, пожежній небезпеці та вибухонебезпеці.

ПЕРЕЛІК ДЖЕРЕЛ

1. Інноваційні зміни онлайн-кінотеатрів 2020 року [Електронний ресурс] – Режим доступу до ресурсу: https://mind.ua/openmind/20225105-divis-vdoma-yak-innovaciyi-zminili-onlajn-kinoteatri-2020-roku

2. Переваги онлайн-кінотеатрів [Електронний ресурс] – Режим доступу до ресурсу: https://www.0382.ua/list/232183

3. Постановка задачі [Електронний ресурс] – Режим доступу до ресурсу: https://studfile.net/preview/5679245/

4. Вимоги до програмного забезпечення [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Вимоги_до_програмного_забезпечення

5. Онлайн діаграми [Електронний ресурс] – Режим доступу до ресурсу: https://app.diagrams.net/

6. Аналіз вимог [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Аналіз_вимог

7. Блок-схеми алгоритму [Електронний ресурс] – Режим доступу до ресурсу: https://yevshan.com.ua/info/006/content/content3.html

8. Програма DFD [Електронний ресурс] – Режим доступу до ресурсу: https://www.lucidchart.com/pages/ru/диаграмма-dfd

9. ER-діаграми DFD [Електронний ресурс] – Режим доступу до ресурсу: http://hi-news.pp.ua/kompyuteri/14668-er-dagrama-ce-opis-vidi-pravila-pobudovi.html

10. Веб-сервер [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Вебсервер

11. Веб-сервер [Електронний ресурс] – Режим доступу до ресурсу: https://uk.theastrologypage.com/web-server

12. Korotkevitch D. Expert SQL Server Transactions and Locking: Concurrency Internals for SQL Server Practitioners / Dmitri Korotkevitch., 2018. – 340 с. – (Apress). – (978-1484239568)

13. Клієнт-серверна архітектура [Електронний ресурс] – Режим доступу до ресурсу: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/

14. Створення HTML [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/ Тім_Бернерс-Лі

15. Hanchett E. Vue.js in Action / E. Hanchett, B. Listwon., 2018. – 375 с. – (1st Edition). – (1617294624).

16. Платформа Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Node.js

17. База даних MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/MongoDB

18. Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Visual_Studio_Code

19. Вплив діяльності людини на довкілля [Електронний ресурс] – Режим доступу до ресурсу: https://pidru4niki.com/17190512/ekologiya/vpliv_diyalnosti_lyudini_dovkillya

20. Вплив людини на довкілля [Електронний ресурс] – Режим доступу до ресурсу: https://nubip.edu.ua/node/53384

21. Дія людини на природу [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/ Дія_людини_на_природу

22. Методи та засоби забезпечення безпеки у приміщенні [Електронний ресурс] – Режим доступу до ресурсу: http://opcb.kpi.ua/wp-content/uploads/2014/08/14_РТФ_ІТС_Практ_Пож_Безп_Категорії_прим_будівель.pdf

23. Класифікація приміщень за ступенем небезпеки ураження електричним струмом [Електронний ресурс] – Режим доступу до ресурсу: https://pidru4niki.com/16330826/bzhd/klasifikatsiya_primischen_stupenem_nebezpeki_urazhennya_elektrichnim_strumom

24. ДСТУ Б.В.1.1-36:2016 „Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою”.

ДОДАТКИ

А

Лістинг файлу authController.js папки controller

```
const User = require('../models/User')
const Role = require('../models/Role')
const bcrypt = require('bcrypt')
const jwt = require('jsonwebtoken')
const {validationResult} = require('express-validator')
const {secret} = require('../config.js')

const generateAccessToken = (id, roles) => {
    const payload = {
        id,
        roles
    }
    return jwt.sign(payload, secret, {expiresIn: "24h"})
}


class authController{
    async registration(req, res){
        try {
            const errors = validationResult(req)
            if(!errors.isEmpty()){
                console.log(errors)
                return res.status(400).json({message: "Помилка при
реєстрації", errors, status: 'fail'})

            }

            const {username, password} = req.body
            const candidate = await User.findOne({username})
            if(candidate){
                return res.status(400).json({message: "Користувач з таким
ім'ям вже існує", status: 'fail'})
            }
            const hashPassword = bcrypt.hashSync(password, 7)
            const userRole = await Role.findOne({value: "USER"})
            const user = new User({username, password: hashPassword,
roles: [userRole.value]})
            await user.save()
            delete user.password
            return res.json({message: "Користувач успішно
зареєстрований", status: 'success', user})
        } catch (error) {
            console.log(error)
            res.status(400).json({message: 'Registration error'})
        }
```

```
    }
    async login(req, res){
        try {
            const {username, password} = req.body
            const user = await User.findOne({username})
            if(!user){
                return res.status(400).json({message: `Користувач
${username} не найденний`, status: 'fail'})
            }
            const validPassword = bcrypt.compareSync(password,
user.password)
            if(!validPassword){
                return res.status(400).json({message: `Введенний не
правильний пароль`, status: 'fail'})
            }
            const token = generateAccessToken(user._id, user.roles)
            delete user.password
            return res.json({token, status: 'success', user})
        } catch (error) {
            console.log(error)
            res.status(400).json({message: 'Login error'})
        }
    }
    async getUsers(req, res){
        try {
            const users = await User.find()
            res.json(users)
        } catch (error) {

        }
    }
}

module.exports = new authController()
```

Лістинг файлу createFilmController.js папки controller

```
const Movie = require('../models/Movie')

class CreateFilm{
    async createfilm(req, res){
        try {
            const createdfilm = new Movie({
                name:req.body.name,
                duration:req.body.duration,
                status:req.body.status,
                age:req.body.age,
                country:req.body.country,
```

```
                ganres:req.body.ganres,
                translate:req.body.translate,
                voiceActing:req.body.voiceActing,
                director:req.body.director,
                actors:req.body.actors,
                watchUrl:req.body.watchUrl,
                description:req.body.description,
                imgUrl:req.body.imgUrl,
            })
            await createdfilm.save()
            res.json('film create')
        } catch (error) {
            console.log(error)
            res.status(400).json({message: 'Error film'})
        }
    }
}

module.exports = new CreateFilm()
```

Лістинг файлу movieController.js папки controller

```
const Movies = require('../models/Movie')
const bcrypt = require('bcrypt')
const jwt = require('jsonwebtoken')
const {validationResult} = require('express-validator')
const {secret} = require('../config.js')


class MovieController{
    async getMovies(req, res){
        try {
            const movies = new Movies()
            await movies.save()
            res.json('server work')
        } catch (error) {
            console.log(error)
        }
    }
}

module.exports = new MovieController()
```

Б

Лістинг файлу authMiddlewaree.js папки middlewaree

```
const jwt = require('jsonwebtoken')
const {secret} = require('../config.js')
module.exports = function(req, res, next){
    if(req.method === 'OPTIONS'){
        next()
    }
    try{
        const token = req.headers.authorization.split(' ')[1]
        if(!token){
        return res.status(403).json({massage: "Користувач не
авторизований"})
        }
        const decodeData = jwt.verify(token, secret)
        req.user = decodeData
        next()
    }catch(error){
        console.log(error)
        return res.status(403).json({massage: "Користувач не
авторизований"})
    }
}
```

Лістинг файлу roleMiddlewaree.js папки middlewaree

```
const jwt = require('jsonwebtoken')
const { secret } = require('../config')
module.exports = function(roles){
    return function(req, res, next){
        if(req.method === 'OPTIONS'){
            next()
        }
        try{
            const token = req.headers.authorization.split(' ')[1]
            if(!token){
            return res.status(403).json({massage: "Користувач не
авторизований"})
            }
            const {roles: userRoles} = jwt.verify(token, secret)
            let hasRole = false
            userRoles.forEach(role => {
                if(roles.includes(role)){
                    hasRole = true
                }
            });
            if(!hasRole){
```

```
            return res.status(403).json({massage: "У вас не має
доступу"})
            }
            next()
        }catch(error){
            console.log(error)
            return res.status(403).json({massage: "Користувач не
авторизований"})
        }
    }
}
```

В

Лістинг файлу MainPage.vue папки page

```
<template>
<div>

    <Header @search="searchMovieName"/>
    <div class="icon">
        <div v-for="movie in currentPageMovies" :key="movie._id">
            <IconsBlock :movie="movie" />
        </div>
    </div>

    <Genres  @filter="searchGanresMovies"/>
    <div class="container">
        <div class="wrap-container">
            <div class="block">
                <Information />
                <ExitRegistr />
            </div>
            <div class="movi" v-for="movie in currentPageMovies"
:key="movie._id+'g'">
                <BlockFilm :movie="movie" @filter="searchActorinMovies"/>
            </div>

        </div>
    </div>
    <div class="pagination-block">
        <Paginate
        v-model="page"
        :page-count="pageCount" class="number"
        :click-handler="pageChangeHandler"
        :prev-text="'Назад'"
        :next-text="'Вперед'"
        :container-class="'pagination'" />
    </div>

    <Footer />
</div>
</template>

<script>
import Header from '../components/Header.vue'
import IconsBlock from '../components/IconsBlock.vue'
import Footer from '../components/Footer.vue'
import Genres from '../components/Genres.vue'
import ExitRegistr from '../components/ExitRegistr.vue'
import BlockFilm from '../components/BlockFilm.vue'
```

В2

```
import Information from '../components/Information.vue'
import paginationMixin from '@/mixins/pagination.mixin'

export default {
    name: 'MainPage',
    components: {
        Header,
        IconsBlock,
        Footer,
        Genres,
        ExitRegistr,
        BlockFilm,
        Information
    },
    mixins: [paginationMixin],
    data() {
        return {
            movies: [],
            searchMovies: '',
            clonMovies: [],
            genresLists: []
        }
    },
    async beforeMount() {
        await this.getMovies()
        if (this.$route.query.filter){
            this.searchActorinMovies(this.$route.query.filter)
        }
        if (this.$route.query.search){
            this.searchMovieName(this.$route.query.search)
        }
        this.setupPagination(this.movies)
    },
    methods: {
        async getMovies() {
            let response = await fetch('http://localhost:3000/film')
            this.movies = await response.json()
            this.clonMovies = JSON.parse(JSON.stringify(this.movies ||
[]))
        },
        searchMovieName(searchMovieQuery) {
            this.movies = this.clonMovies.filter(movie =>
movie.name.toLowerCase().includes(searchMovieQuery.toLowerCase()));
            this.setupPagination(this.movies)
        },
        searchGanresMovies(searchGanresQuery){
            if(searchGanresQuery !== 'Всі жанри' && searchGanresQuery !==
'Всі актори'){
```

```
                this.movies = this.clonMovies.filter(movie =>
movie.ganres.includes(searchGanresQuery))
            }else{
                this.movies = this.clonMovies
            }
            this.setupPagination(this.movies)
        },
        searchActorinMovies(searchActorsQuery){
            this.movies = this.clonMovies.filter(movie =>
movie.actors.includes(searchActorsQuery));
            this.setupPagination(this.movies)
        }
    },
}
</script>
```

Лістинг файлу PageFilm.vue папки page

```
<template>
<div>
    <Header @search="searchMovies"/>
    <div class="container">
        <div class="wrap-container">
            <div class="block">
                <div class="block-film-name">
                    <h1>{{movie.name}}</h1>
                    <div class="flex-films">
                        <div class="img-film">
                            <img :src="getImg(movie.imgUrl)" alt="">
                        </div>
                        <div class="info-film">
                            <div class="table">
                                <p class="duration color">Тривалість:</p>
                                <p class="duration color">Статус:</p>
                                <p class="duration color">Рік:</p>
                                <p class="duration color">Країна:</p>
                                <p class="duration color">Жанр:</p>
                                <p class="duration color">Переклад:</p>
                                <p class="duration color">Озвучка:</p>
                                <p class="duration color">Режисер:</p>
                                <p class="duration color">В ролях:</p>
                            </div>
```

```
                            <div class="info">
                                <p>{{movie.duration}}</p>
                                <p>{{movie.status}}</p>
                                <p>{{movie.age}}</p>
                                <p>{{movie.country}}</p>
                                <div class="actors-block" v-for="(ganres,
i) in movie.ganres" :key="i">
                                    <p>{{ganres}}</p>
                                </div>
                                <p>{{movie.translate}}</p>
                                <p>{{movie.voiceActing}}</p>
                                <p>{{movie.director}}</p>
                                <div class="actors-block left">
                                    <router-link class="actors-href"
tag="button"
                                    v-for="(actor, i) in movie.actors"
:key="i"
                                     :to="{path: '/', query: {filter:
actor}}">
                                    {{actor}}</router-link>
                                </div>
                            </div>
                        </div>
                    </div>
                    <div class="text">
                        <p class="description-movie">Коротко про
фільм</p>
                        <p>{{movie.description}} </p>
                    </div>
                    <div class="video-film">
                        <!-- <video  class="video"
                        controls
                        ></video> -->
                        <iframe width="700" height="400"
:src="this.movie.watchUrl" frameborder="0" allow="accelerometer;
                        autoplay; clipboard-write; encrypted-media;
gyroscope; picture-in-picture" allowfullscreen></iframe>
                    </div>
                    <div class="block-like">
                            <button @click="patchLikes('counterUp')"
class="like-btn-img"><p
class="counter">{{this.movie.likes.counterUp.length}}<p/></button>
                            <button @click="patchLikes('counterDown')"
class="like-btn-img bad"><p
class="counter">{{this.movie.likes.counterDown.length}}<p/></button>
                            <button @click="patchLikes('counterLove')"
class="emoji love"><p class="counter-
emoji">{{this.movie.likes.counterLove.length}}<p/></button>
```

B5

```
                            <button @click="patchLikes('counterAngry')"
class="emoji angry"><p class="counter-
emoji">{{this.movie.likes.counterAngry.length}}<p/></button>
                            <button @click="patchLikes('counterCryning')"
class="emoji cryning"><p class="counter-
emoji">{{this.movie.likes.counterCryning.length}}<p/></button>
                            <button @click="patchLikes('counterGlasses')"
class="emoji glasses"><p class="counter-
emoji">{{this.movie.likes.counterGlasses.length}}<p/></button>
                            <button @click="patchLikes('counterHappy')"
class="emoji happy"><p class="counter-
emoji">{{this.movie.likes.counterHappy.length}}<p/></button>
                            <button @click="patchLikes('counterNeutral')"
class="emoji neutral"><p class="counter-
emoji">{{this.movie.likes.counterNeutral.length}}<p/></button>
                            <button @click="patchLikes('counterSad')"
class="emoji sad"><p class="counter-
emoji">{{this.movie.likes.counterSad.length}}<p/></button>
                            <button @click="patchLikes('counterSmail')"
class="emoji smail"><p class="counter-
emoji">{{this.movie.likes.counterSmail.length}}<p/></button>
                            <button
@click="patchLikes('counterVomiting')" class="emoji vomiting"><p
class="counter-
emoji">{{this.movie.likes.counterVomiting.length}}<p/></button>
                        </div>
                    </div>
                    <ExitRegistr />
                </div>
            </div>
        </div>
        <Footer />
</div>
</template>

<script>
import Header from './../components/Header.vue'
import Footer from './../components/Footer.vue'
import ExitRegistr from './../components/ExitRegistr.vue'
import addIcons from '@/mixins/addIconsMovies.mixin'
import {
    VUE_APP_API_HOST
} from './../constants.js'
// import axios from 'axios'
// import VueAxios from 'vue-axios'

export default {
    name: 'PageFilm',
    mixins: [addIcons],
```

B6

```
    components: {
        Header,
        ExitRegistr,
        Footer
    },
    data(){
        return {
            movie: {},
            clonMovies: [],
            VUE_APP_API_HOST
        }
    },
    beforeMount() {
        const movieId = this.$route.params.id;
        this.getFilm(movieId)
    },
    mounted() {
        this.moveUp()
    },
    methods: {
        moveUp() {
            window.scrollTo(0, 0);
        },
        async getFilm(movieId) {
            const movieToWatchResponse = await
fetch('http://localhost:3000/createfilm/' + movieId, {
                method: 'GET',
            })
            this.movie = await movieToWatchResponse.json()
        },
        searchActor(actor){
            this.$emit('filter', actor)
        },
        searchMovies(searchValue){
            this.$router.push({path: '/', query: {search: searchValue}})
        },
        async patchLikes(likeName){
            this.user = JSON.parse(localStorage.getItem('user'))
            const body = {
                movieId: this.movie._id,
                likeName: likeName,
                userId: this.user._id

            }
            console.log(body)
            if(body.userId){
```

```
                const patches = await
this.$axios.patch('http://localhost:3000/likeMovie', body)
            }else{
                alert('Користувач не зареєстрований')
            }
        }
    },
}
</script>
```

Лістинг файлу Registration.vue папки page

```
<template>
<div class="go-exit">
    <div class="container">
        <div class="header-block">
            <div class="color-header-img" @click="$router.push('/')">
                <h1 class="logo-kinotale">Kino</h1>
                <h1 class="logo-kinotale tale">Tale</h1>
                <!-- <img @click="$router.push('/')" class="logo-img"
src="~@/assets/img/lo.png" alt="logo"> -->
            </div>
            <div class="major-block">
                <div class="site-registr">
                    <p>Тут ви можете зареєструватися на сайті</p>
                </div>
                <div class="login">
                    <div class="block-flex">
                        <p class="login-p">Логін</p>
                        <p class="cursiv">(мінімум 3 символи англ. Без
крапок)</p>
                    </div>
                    <input v-model="username" id="userName"
placeholder="Login" type="text" class="input-log">
                </div>
                <div class="login">
                    <div class="block-flex">
                        <p class="login-p">E-mail</p>
                        <p class="cursiv">(введіть діючий email)</p>
                    </div>
                    <input v-model="email" id="email" placeholder="e-
mail" type="email" class="input-log">
                </div>
                <div class="password">
                    <div class="block-flex">
                        <p class="login-p">Пароль</p>
                        <p class="cursiv">(мінімум 6 символів)</p>
```

```
                    </div>
                    <input v-model="password" id="password"
placeholder="Пароль" type="password" class="input-log">
                    <div class="toconfirm">
                        <p class="login-p">Підтвердити пароль</p>
                        <input v-model="repeatPassword"
placeholder="Підтвердити пароль" type="password" class="input-log">
                    </div>
                    <div class="btn-exit">
                        <button class="btn" @click="registration()">
Реєстрація</button>
                    </div>
                    <div class="registration-link">
                        <a @click="$router.push('/exit')"
href="">Війти</a>
                        <a @click="$router.push('/')" class="back-to-
home" href="">Повернутися на головну</a>
                    </div>
                </div>
            </div>

        </div>
    </div>
</div>
</template>

<script>

export default {
    name: 'Registration',
    data() {
        return {
            username: '',
            email: '',
            password: '',
            repeatPassword: ''
        }
    },
    methods: {
        async registration() {
            const body = {
                username: this.username,
                email: this.email,
                password: this.password,
                repeatPassword: this.repeatPassword,
            }
            const goToregistration = await
fetch('http://localhost:3000/auth/registration', {
```

```
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify(body)
            })
            this.$router.push('/')
            const data = await goToregistration.json()
            console.log(data)
            localStorage.setItem('user' ,JSON.stringify(data.user));
            this.$router.go(0);
        },
    },
}
</script>
```

Лістинг файлу Exit.vue папки page

```
<template>
<div class="go-exit">
    <div class="container">
        <div class="header-block">
            <div class="color-header-img" @click="$router.push('/')">
                <h1 class="logo-kinotale">Kino</h1>
                <h1 class="logo-kinotale tale">Tale</h1>
                <!-- <img @click="$router.push('/')" class="logo-img"
src="~@/assets/img/lo.png" alt="logo"> -->
            </div>
            <div class="major-block">
                <div class="login">
                    <p>Логін</p>
                    <input v-model="username" type="text"
placeholder="Login" class="input-log">
                </div>
                <div class="password">
                    <p>Пароль</p>
                    <input v-model="password" type="password"
placeholder="Пароль" class="input-log">
                    <div class="remember-me">
                        <input type="checkbox" name="" id="">
                        <p>Запам'ятати мене</p>
                    </div>
                    <div class="btn-exit">
                        <button @click="login()"  class="btn">
Вхід</button>
                    </div>
                    <div class="registration-link">
```

```
                        <a @click="$router.push('/registration')"
href="">Реєстрація</a>
                        <a @click="$router.push('/')" class="back-to-
home" href="">Повернутися на головну</a>
                    </div>
                </div>
            </div>

        </div>
    </div>
</div>
</template>

<script>
export default {
    name: 'Exit',
    data() {
        return {
            username: '',
            password: '',
        }
    },
    methods: {
        async login() {
            const body = {
                username: this.username,
                password: this.password
            }
            const goToExit = await
fetch('http://localhost:3000/auth/login', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify(body)
            })
            const data = await goToExit.json()
            if(data.status === 'success'){
                this.$router.push('/')
                console.log("Ви успішно увішли!")
            }else{
                console.log("Ви не увішли!")
            }
            console.log(data)
            localStorage.setItem('user' ,JSON.stringify(data.user));
        }
    }
}
</script>
```