НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Наталія АУШЕВА
«___»_____2022 р.

Дипломна робота

на здобуття ступеня бакалавра
спеціальності 122 «Комп'ютерні науки»
освітня програма «Комп'ютерний моніторинг та геометричне моделювання процесів і систем»

на тему: «Клієнт-серверна система обробки та аналізу вхідних даних користувача для оптимізації роботи веб-порталу»

Виконав:
студент IV курсу, групи ТР-82
Єременко Назарій Максимович _____

Керівник:
Старший викладач
Гурін Артем Леонідович _____

Рецензент:
_____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2022

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший

Спеціальності 122 «Комп’ютерні науки»

Освітня програма «Комп’ютерний моніторинг та геометричне моделювання процесів і систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Наталія АУШЕВА
(підпис)
”___”_____2022р.

ЗАВДАННЯ

на дипломну роботу студенту

Єременка Назарія Максимовича

(прізвище, ім’я, по батькові)

1. Тема роботи: Клієнт-серверна система обробки та аналізу вхідних даних користувача для оптимізації роботи веб-порталу

керівник роботи ст. викладач Гурін Артем Леонідович

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”27” травня 2022р. №

2. Строк подання студентом роботи 10.06.2022р.

3. Вихідні дані до роботи: Технологія node.js, середовище розробки WebStorm. Основні програмні модулі: прикладний програмний інтерфейс, обробник запитів, база даних, сервер. Всі програмні модулі об’єднані клієнт-серверною архітектурою.

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

Ознайомитись з наявними аналогічними програмними системами. Розробити алгоритм програмної системи. Розробити графічний інтерфейс. Розробити головний модуль. Побудувати клієнт-серверний зв'язок.

5. Перелік ілюстративного матеріалу

Наявні програмні модулі, архітектура програмного продукту, відображення роботи бази даних, інтерфейс роботи користувача, демонстрація клієнт-серверної логіки.

7. Дата видачі завдання "02" травня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	*Затвердження теми роботи*	02.05.2022	
2.	*Вивчення та аналіз задачі*	02.05.2022- 08.05.2022	
3.	*Розробка архітектури та загальної структури системи*	09.05.2022- 15.05.2022	
4.	*Розробка структур окремих підсистем*	16.05.2022- 22.05.2022	
5.	*Програмна реалізація системи*	23.05. 2022- 29.05. 2022	
6.	*Оформлення пояснювальної записки*	02.06.2022	
7.	*Захист програмного продукту*	06.06. 2022	
8.	*Передзахист*	06.06. 2022- 09.06. 2022	
9.	*Захист*	20.06. 2022 - 30.06. 2022	

Студент _____ _____Єременко Н.М._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____Гурін А.Л. _____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Записка містить 55 сторінок, 33 рисунків, 1 додаток та 18 посилання.

Метою роботи є створення веб-порталу каталогу фільмів із впровадженням клієнт-серверної архітектури, для аналізу та обробки вхідних даних користувача.

Було обрано за основу мову програмування JavaScript та платформу Node.js для роботи, оскільки JavaScript — потужний інструмент для веб-орієнтованої розробки, а Node.js має усі необхідні інструменти та компоненти для обробки даних користувача в браузері, утворення клієнт-серверної архітектури.

Розроблено систему з користувацьким інтерфейсом у вигляді веб-порталу, що надає можливість оброблювати вхідні дані, заносити і зчитувати інформацію з бази даних у режимі реального часу, створювати, оновлювати та видаляти окремі об’єкти та їх складові, тим самим дозволяючи користувачу вести каталог кінострічок.

Ключові слова: клієнт-сервер, веб-орієнтовано, веб-портал, JavaScript, Node.js.

ABSTRACT

The note contains 55 pages, 33 figures, 1 appendix and 18 links.

The aim of the work is to create a web portal of the movie catalog with the implementation of client-server architecture for analysis and processing of user input data.

The JavaScript programming language and the Node.js platform were chosen as the basis for work, because JavaScript is a powerful tool for web-based development, and Node.js has all the necessary tools and components for processing user data in the browser, creating a client-server architecture.

The system was developed with the user interface in form of the web portal, allows to process input data, enter and read information from the database in real time, create, update and delete individual objects and their components, that allows user to maintain the catalog of films.

Keywords: client-server, web-oriented, web portal, JavaScript, Node.js.

ЗМІСТ

Перелік умовних позначень, скорочень і термінів

HTML (HyperText Markup Language)— стандартизована мова розмітки гіпертексту документів для перегляду браузерних веб-сторінок.

CSS (Cascading Style Sheets) —Каскадні таблиці стилів, простіше - мова стилю сторінок для опису дизайну, зовнішнього стилю.

XML (Extensible Markup Language) — розширювана мова розмітки.

IDE (Integrated Development Environment) — Інтегроване середовище розробки

ЕОМ — електронна обчислювальна машина.

англ. — англійською.

юзер – (від англ. user) – користувач.

валідація – (від англ. validation) – затвердження, ратифікація.

рефа́кторинг (англ. refactoring), або перепроектування коду, переробка коду, рівносильне перетворення алгоритмів — процес зміни внутрішньої структури програми, що не зачіпає її зовнішньої поведінки та поліпшує її роботу.

кеш (англ. cache, від фр. cacher - «ховати») - проміжний буфер зі швидким доступом до нього, що містить інформацію, яка може бути запитана з найбільшою ймовірністю.

ВСТУП

Сьогодні інформацію розглядають як один із основних ресурсів розвитку суспільства, а інформаційні системи та технології як засіб підвищення продуктивності та ефективності роботи людей.

Створення сучасних електронних обчислювальних машин дозволило автоматизувати обробку даних у багатьох сферах людської діяльності, зокрема на підприємствах. Без сучасних систем обробки даних важко уявити сьогодні передові виробничі технології, управління економікою всіх її рівнях, наукові дослідження, освіту, видавничу справу, функціонування засобів і т.д.

Комп'ютери, програми та системи з часом ростуть і розвиваються. Щоб адаптуватися до цих змін, компанії повинні постійно оновлювати програмне та апаратне забезпечення. Якщо б підприємства не використовували сервер, то повинні були б змінювати систему окремо, на кожному клієнті. Наприклад, антивірусні рішення необхідно завантажувати розрізнено – тоді це може призвести до проблем на кожній робочій станції. Та при наявності антивірусних серверів, можна завантажити одну версію оновлених визначень вірусів з Інтернету, а потім поширити ці оновлення клієнтам, за яких цей сервер відповідає. Ці системи також можуть визначити, чи було успішно оновлено клієнт і коли. Це суттєво поліпшує існування та обслуговування практично будь-якого продукту.

Інтегральні системи, призначені для роботи з базами даних, засновані на двох взаємодіючих компонентах - клієнті, що відповідає за організацію діалогу з користувачем і несе на собі бізнес-логіку, і сервері, що забезпечує розраховану на багато користувачів роботу з даними та їх цілісність. Рівень залежності бізнесу від інформаційних систем є дуже високим. Розробники займаються завданнями реалізації адекватної технічним вимогам функціональності та інтерфейсу користувача, оптимізацією обміну даними між різними компонентами системи. Корпоративні системи мають високий рівень складності, тому вони повинні бути надійними і легкокерованими. В результаті таких вимог виникає необхідність

виділення з клієнтської та серверної частини системи компонентів, що несуть суворо певну службову функціональність.

Як і великі, так і малі організації зазвичай зменшують витрати, впроваджуючи систему клієнт-сервер. Сервер може бути створений, наприклад, для зберігання документів компанії, дозволяючи окремим клієнтам використовувати менші жорсткі диски. Як додатковий бонус, дані кожного можна архівувати, просто створивши резервні копії файлів на сервері, замість копіювання окремих файлів кожного клієнта. Системи телефонії на основі IP дозволяють всім телефонам у середовищі використовувати одного постачальника послуг міжміського зв'язку замість окремих облікових записів міжміського сполучення. Економія часто служить визначальним аргументом для замовників і мотивацією для постачальників серверних систем.

Не винятком є сфера ведення каталогів фільмів, а саме веб-сайти з вільно редагованою базою даних про кінематограф. Такий портал потребує цю базу даних для збереження інформації про назви фільмів, їх рейтинги, акторські склади та загалом будь-які ідентифікуючі ту чи іншу кінострічку дані. Не менш важливим є те, що такі дані повинні бути чітко структуровані, аби можна було зручно і швидко працювати з базою даних, як і для оновлення, так і швидкого відображення результату занесення тої чи іншої інформації. Так як користувачу важливо мати позитивний досвід з користування порталом, окрім логічної і лаконічної бази даних, система повинна мати зручний прикладний програмний інтерфейс, доступний та інтуїтивно зрозумілий для юзера, та бек-енд частину – обробник даних, що буде регулювати весь процес валідації (перевірки) вхідних запитів від користувача та відповідати за навігацію між сторінками, вікнами, модулями системи. Всі вищевказані структури становитимуть цілісну клієнт-серверну архітектуру, що і даватиме змогу системі виконувати бажані функції та відповідати вимогам користувача.

Враховуючи усі вище перераховані пункти, виникає необхідність розробки ефективного інструменту ведення кінотеки, який буде доступним та зрозумілим

для користувача будь-якого рівня, та повністю відповідатиме як меті порталу, так і сучасним канонам клієнт-серверної архітектури.

Тож, метою дипломної роботи є створення клієнт-серверної системи обробки та аналізу вхідних даних користувача для оптимізації роботи веб-порталу, покривши запити зручності та швидкості.

Для розробки програмного продукту було обрано гнучку, динамічну, об’єктно-орієнтовану мову програмування JavaScript, яка до того ж є платформонезалежною (може запускатися з пристроїв на різних операційних системамх) та має безліч бібліотек, що спрощують роботу та надають готові рішення. Розробка проводилась у інтегрованому середовищі розробки JetBrains WebStorm, так як воно наділене інструментами, що добре опрацьовують структуру проектів і поліпшують роботу з різними аспектами написання коду, у числі автодоповнення коду, безпечного рефакторингу, постійного пошуку потенційних проблем та підказок щодо їх виправлення.

Програмний продукт має ряд функцій та базових можливостей, метою яких є вирішення поставленої задачі. Робота умовно поділена на етапи: додавання, оновлення або видалення об’єкту «фільм», редагування акторського складу, сортування списку за наявними параметрами.

Даний програмний продукт може бути використаний будь-яким користувачем, що бажає вести власну теку кінострічок.

1 ОПИС ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

Складно сперечатися з приводу того, що сучасні темпи розвитку обчислювальної техніки, програмного забезпечення та комп'ютерних технологій надали поштовх розвитку систем, основаних на клієнт-серверній архітектурі. Більшість баз даних, у тому числі фільмографії, мають на меті збільшення швидкості процесів обробки та вичитки інформації, зручності та зрозумілості виконання робіт користувачем програмного продукту.

Метою дипломної роботи є створення клієнт-серверної системи обробки та аналізу вхідних даних користувача для оптимізації роботи веб-порталу, покривши запити зручності та швидкості обробки всіх запитів між веб-інтерфейсом, обробником запитів та базою даних.

Зміст: Створення лаконічного та простого у використанні прикладного програмного інтерфейсу веб-порталу каталогу кінострічок, що взаємодіє із обробником запитів, утворення клієнт-серверного зв'язку задля коректної роботи сайту

Задачі, що вирішуються:

• Простота інтерфейсу;

• Коректне відображення елементів керування;

• Забезпечення логіки обробки та запису вхідних даних користувача у базу даних;

• Можливість додавати, редагувати та видаляти у базу даних об'єктів та їх складових

Вхідні дані: Шаблони сторінок і переходів між ними, прототип клієнт-серверної системи.

Вихідні дані: Користувацький інтерфейс з можливістю перегляду списку фільмів, їх сортування по даті випуску, рейтингу, алфавітному порядку назв; редагування даних по конкретному фільму; переглядання та правка акторського складу.

Для збереження напрацювань з можливістю повторного відображення каталогу та подальшого редагування програмний продукт повинен містити можливість збереження даних у вигляді бази даних.

Створений інтерфейс має весь потрібний функціонал для формування та обробки каталогу кінострічок, є зрозумілим для кінцевого користувача, що забезпечує комфорт під час роботи з продуктом та швидкість у веденні теки.

2 ОГЛЯД ІСНУЮЧИХ КЛІЄНТ-СЕРВЕРНИХ СИСТЕМ

Мережа «клієнт-сервер» відноситься до налаштування мережі, яка використовує клієнтські апаратні пристрої та сервери. Модель мережі клієнт-сервер може використовуватися через локальну мережу або Інтернет. Поширені приклади мереж клієнт-сервер включають DNS (системи доменних імен), веб-браузери та веб-сервери, а також клієнти FTP (протоколу передачі файлів).

Більшість людей щодня стикаються з мережею клієнт-сервер за допомогою комп'ютерів, смартфонів і планшетів, підключених до Інтернету. Їхні пристрої — це клієнти, які запитують доступ до вмісту веб-сторінки, який потім доставляється їм серверами веб-сайту. [1]

Отже, сторона клієнта і сторона сервера – це терміни веб-розробки, які описують, де виконується код програми. Веб-розробники також будуть називати цю відмінність інтерфейсом і бек-ендом, хоча на стороні клієнта/сервера і на стороні інтерфейсу/бек-енда це не зовсім те саме. У безсерверній архітектурі постачальник без сервера розміщує та призначає ресурси всім процесам на стороні сервера, а процеси збільшуються в міру збільшення використання програми. [4]

2.1 DNS

DNS - (система доменних імен) перетворює доменні імена, зручні для людського сприйняття (наприклад, www.amazon.com), в IP-адресу, зрозумілі для машини (наприклад, 192.0.2.44).

Система DNS в Інтернеті дуже схожа на телефонну книгу, яка встановлює прив'язку імен абонентів до їх номерів. Сервери DNS перетворюють запити за іменами на IP-адреси, забезпечуючи з'єднання кінцевого користувача з певним сервером при введенні доменного імені у веб-браузер користувача. Такі повідомлення називаються запитами.

Основою DNS є представлення про ієрархічну структуру імені та зони. Кожен сервер, що відповідає за ім'я, може нести відповідальність за подальшу частину домену іншого серверу (з адміністративною точкою — іншій організації чи людині), що дозволяє нести відповідальність за актуальність інформації на серверах різних організацій (людей), що відповідають лише за «свою» частину доменного імені.

Починаючи з 2010 року в DNS впроваджуються засоби перевірки цілісності системних даних, які називаються розширеннями безпеки DNS (DNSSEC). Дані, що передаються, не шифруються, але їх достовірність перевіряється криптографічними способами. Впроваджений стандарт DANE забезпечує передачу засобів DNS достовірної криптографічної інформації (сертифікатів), використовуваних для встановлення безпечних і захисних з'єднань транспортного та прикладного рівня. [5]

DNS має такі характеристики:

• Розподіл адміністрування. Відповідальність за різні частини ієрархічної структури несуть різні люди чи організації.

• Розподіл зберігання інформації. Кожен вузол мережі в обов'язковому порядку повинен зберігати ті дані, які входять до його зони відповідальності, і (можливо) адреси кореневих DNS-серверів.

• Кешування інформації. Вузол може зберігати деяку кількість даних не зі своєї зони відповідальності зменшення навантаження на мережу. Ієрархічна структура, в якій всі вузли об'єднані в дерево, і кожен вузол може самостійно визначати роботу нижчих вузлів, або делегувати (передавати) їх іншим вузлам.

• Резервування. За зберігання та обслуговування своїх вузлів (зон) відповідають (зазвичай) кілька серверів, розділені як фізично, так і логічно, що забезпечує збереження даних та продовження роботи навіть у разі збою одного з вузлів.

2.2 FTP

FTP (англ. File Transfer Protocol) - протокол передачі файлів по мережі, FTP є найпоширенішим способом надсилання та отримання файлів між двома комп’ютерами. Прикладом використання FTP є веб-розробники, які підключаються до свого веб-сервера за допомогою FTP-клієнта або програми FTP (наприклад, FileZilla), щоб надіслати (завантажити) оновлені версії веб-сторінки.

Протокол побудований на архітектурі «клієнт-сервер» і використовує різні мережеві з'єднання для передачі команд і даних між клієнтом і сервером. Користувачі FTP можуть пройти аутентифікацію, передати логін і пароль відкритим текстом, або ж, якщо це дозволено на сервері, вони можуть підключатися анонімно. Можна використовувати протокол SSH для безпечної передачі, що скриває (шифрує) логін і пароль, а також шифрує вміст. [6]

Поширеними прикладами використання FTP-клієнта є:

• Публікація сторінок сайту на інтернет-сервері Веб-розробником

• Завантаження музики, програм та будь-яких інших файлів даних звичайним користувачем інтернету.

Дані приклади часто навіть не усвідомлюється багатьма користувачами FTP-клієнта та протоколу, так як багато публічних серверів не вимагають додаткових даних для аутентифікації користувачів, а Інтернет-браузери (також є FTP-клієнтами) здійснюють скачування файлів без додаткових питань.

У найпростішому для користувача (але при цьому найбільш комплексному) випадку FTP-клієнт є емулятором файлової системи, яка просто знаходиться на іншому комп'ютері. З цією файловою системою можна робити всі звичні користувачеві дії: копіювати файли з сервера та на сервер, видаляти файли, створювати нові файли. В окремих випадках можливе також відкриття файлів для перегляду, запуску програм, редагування. Необхідно враховувати лише, що відкриття файлу передбачає його попереднє завантаження на комп'ютер користувача. Прикладами таких програм можуть бути:

• Інтернет-браузери (часто працюють у режимі "тільки читання", тобто не дозволяють додавати файли на сервер)

• Багато файлових менеджерів, наприклад: Windows Explorer (Провідник), WinSCP, Total Commander, FAR, Midnight Commander, Krusader

• Спеціалізовані програми, наприклад: FileZilla

• Онлайн клієнти, робота з якими здійснюється за допомогою будь-якого інтернет-браузера

Завдяки поширеності протоколу FTP, прості (з точки зору реалізації) FTP-клієнти є практично в кожній операційній системі. Однак, використання цих клієнтів вимагає навичок використання консолі, а також знання команд протоколу для спілкування з сервером. Так у Windows такою утилітою є ftp.exe. У багатьох збірках Linux також є утиліта ftp.

Файлова система на віддаленому сервері зазвичай має налаштування прав доступу для різних користувачів. Так, наприклад, анонімним користувачам можуть бути доступні лише деякі файли, про існування інших користувачів не знатимуть. Іншій групі користувачів можуть бути доступні інші файли або, наприклад, на додачу до прав на читання файлів, можуть бути також дані права на запис нових або оновлення наявних файлів.

Діапазон варіантів прав доступу залежить від операційної системи та програмного забезпечення кожного конкретного сервера FTP. Як правило, поділяють права на перегляд вмісту папки (тобто можливість отримати список файлів, що містяться в ній), на читання файлу(ів), на запис (створення, видалення, оновлення) файлу(ів).

2.3 IMDb TV

IMDb –(абревіатура від Internet Movie Database)— це онлайн-база даних інформації, пов’язаної з фільмами, телевізійними серіалами, домашнім відео,

відеоіграми та потоковим контентом в Інтернеті, включаючи акторський склад, виробничу групу та особисті біографії, короткий зміст сюжету, дрібниці, оцінки, а також відгуки шанувальників і критичні відгуки. IMDb розпочався як база даних фільмів, керованої фанатами, у групі Usenet «rec.arts.movies» у 1990 році. Поступово база росла і технічна удосконалювалась. У 1993 році з'явилася база у Всесвітній павутині на сервері Кардіффського університету в Уельсі. [7]

Вже тоді база була дуже популярна в Інтернеті. У січні 1996 року була створена компанія «Internet Movie Database, Ltd», власники якої стали розробниками бази. Незважаючи на велику популярність, база продовжувала залишатися захопленням, хоббі — більшість розробників бази мали інше постійне місце роботи.

Скоро база вичерпала ресурси університетського сервера та переїхала на виділені сервери в США та Великобританії. Через рік виділених серверів було вже 7, але і цих потужностей виявилося недостатньо. Підтримувати базу стало складніше, для розвитку сайту не вистачало ресурсів. Декілька компаній запропонували купити IMDb, а базу ентузіастів не довіряли їм — ці компанії бачили в IMDb не єдиний проект для любителів кіно, маркетингову платформу. Багато розділів бази до сих пір у значній мірі наповнюються і виправляються добровольцями. IMDb використовується як базове джерело інформації для кінематографічних станів, однак не визнається достатньо достовірною для її використання як основного або єдиного джерела на тему. Сайт бази IMDb щомісячно відвідує мільйони людей, це один із сорока найпопулярніших сайтів Всесвітньої павутини.

IMDb побудований на архітектурі «клієнт-сервер». Більшість даних можна завантажити у вигляді стиснених текстових файлів, а інформацію можна отримати за допомогою наданих інструментів інтерфейсу командного рядка. Доступна також програма графічного інтерфейсу користувача (GUI) на основі JavaScript, яка здатна обробляти стислі текстові файли, що дозволяє здійснювати пошук та відображати інформацію. Ця програма з графічним інтерфейсом

підтримує різні мови, але дані, пов’язані з фільмом, надаються англійською мовою, доступними на IMDb. Пакет JavaScript під назвою IMDbJs також можна використовувати для обробки стиснених текстових файлів у низку різних баз даних SQL, що забезпечує легший доступ до всього набору даних для пошуку або аналізу даних.

Отже, IMDb виступає чудовим прикладом веб-порталу каталогу фільмів. На даному ресурсі є велика база даних кінострічок, її список можна відсортувати, наприклад, за рейтингом [8]:

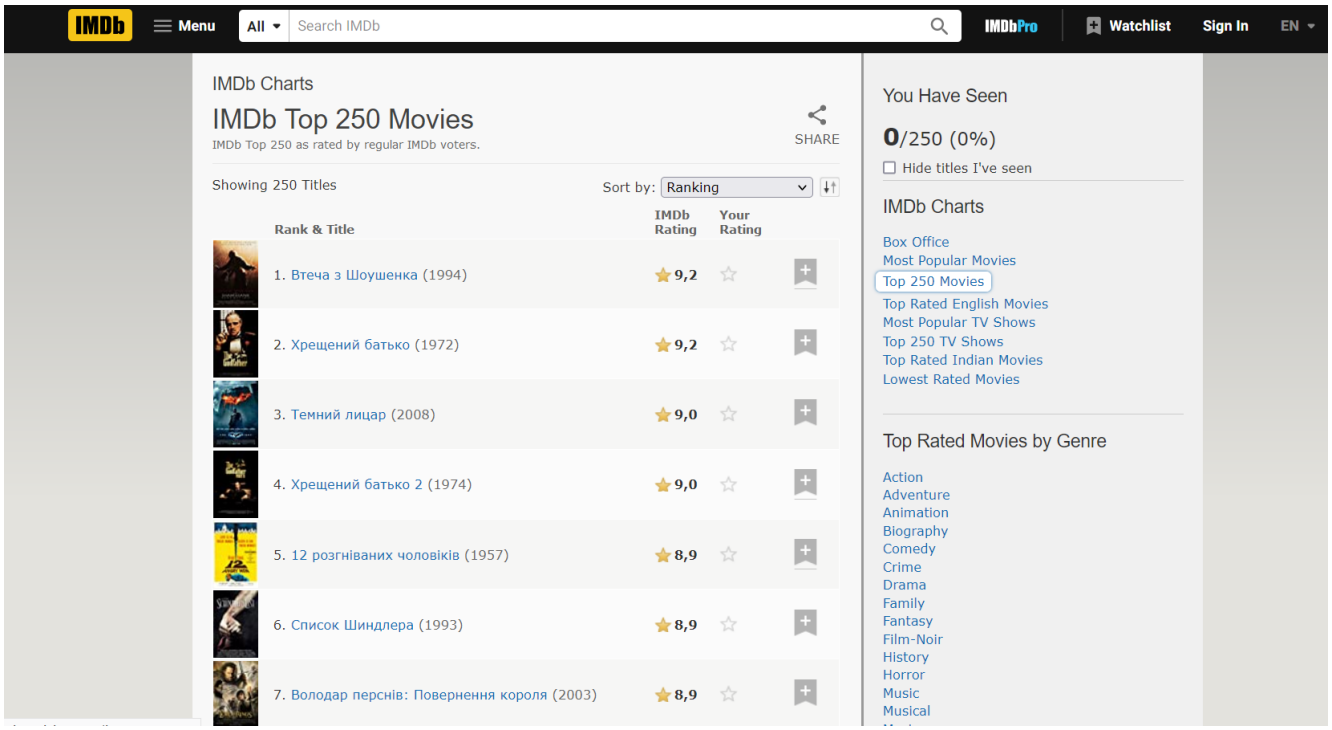

Рисунок 2.3.1 Список фільмів IMDb – сортування за рейтингом

Примітною особливістю IMDb є різноманіття можливостей для сортування фільмів, наприклад за різними видами рейтингу, датою випуску стрічки [9]:

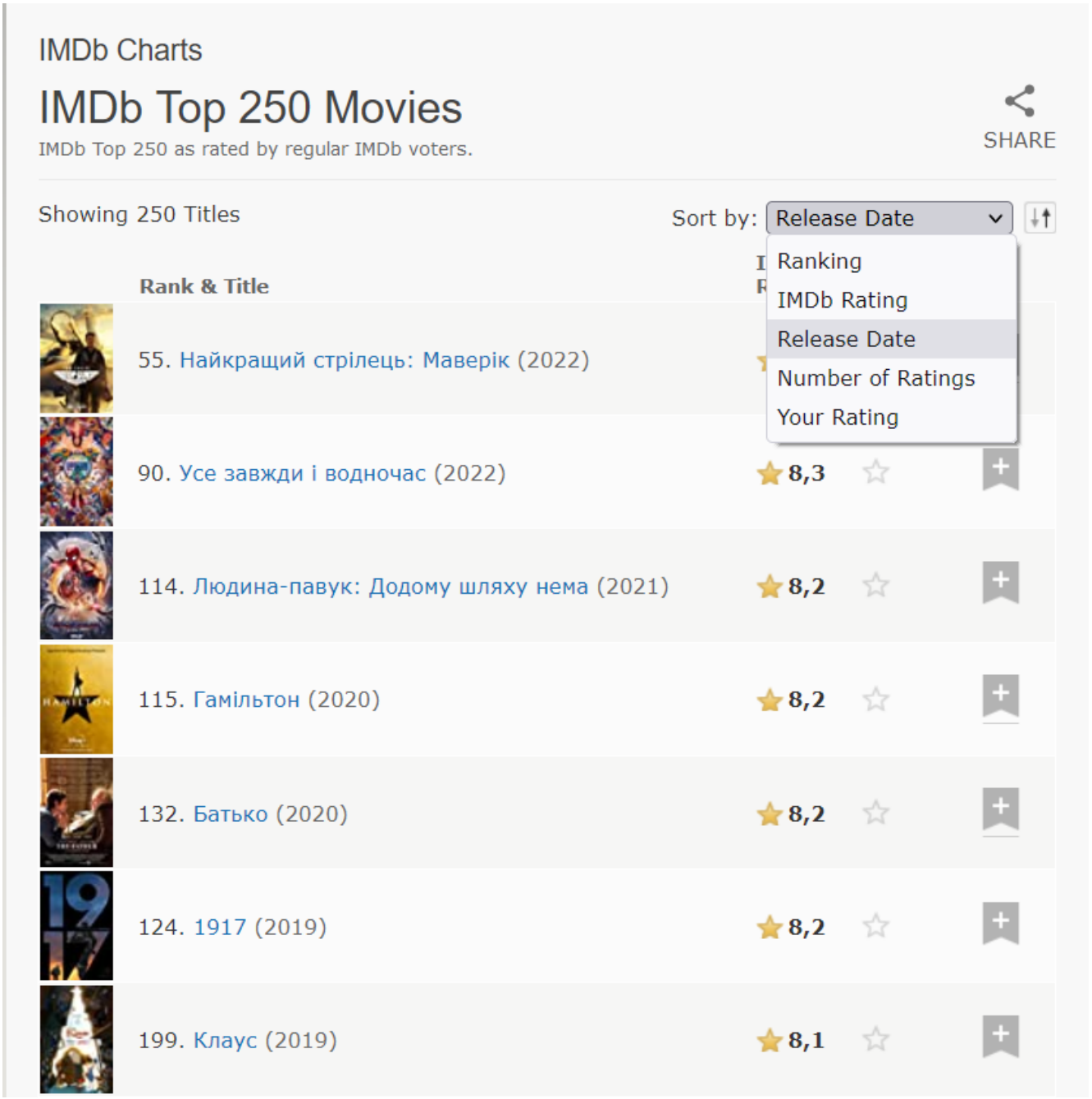

Рисунок 2.3.2 Список фільмів IMDb – сортування за датою випуску (найновіші)

До того ж, для кожного фільма редакторами порталу створюється і редагується акторський склад, що включає в себе фотографії, імена акторів, посилання на їх біографії [10]:

Рисунок 2.3.3 Список акторів фільму «Найкращий стрілець: Маверік» IMDb

Аналізуючи усі перераховані вище приклади клієнт-серверних систем можна чітко визначитись із логікою утворення даної архітектури між програмними модулями. Подібна система забезпечить розуміння усіх необхідні аспектів при подальшій роботі із обраною технологією.

Загострившши увагу на вищенаведеному порталі кінострічок, можна усвідомити принципи побудови зручної та лаконічної системи як з боку користувача, так і з боку розробника, урахувавши всі нюанси та підмітивши особливості, які можна видозмінити, щоб вдосконалити функціонал.

Можна підвести підсумок, що існує широкий вибір систем побудованих на клієнт-серверній архітектурі. В свою чергу, продукти що містять таку можливість є широко спеціалізованими, що ускладнює процес їх освоєння та вимагає більше часу на освоєння.

Необхідно створити програмний продукт, який буде спеціалізований на веденні каталогу кінофільмів, задовольняти всі пов’язані із цим потреби та буде максимально простим і зрозумілим у використанні.

3 ЗАСОБИ РОЗРОБКИ

При створенні інтерфейсу потрібно завжди думати про досвід кінцевого користувача під час використання продукту, тому потрібно уявити та продумати кожну сторінку, логіку взаємодії, обробку та відтворення вхідних та вихідних даних.

Основним інструментом для роботи клієнт-серверної системи було обрано Програмне забезпечення JetBrains WebStorm. Для написання безпосередньо сторінок було використано стандартизована мова розмітки документів для перегляду веб-сторінок у браузері HTML, формальна мова опису зовнішнього вигляду веб-сторінки CSS, та багатофункціональний набір інструментів інтерфейсу Bootstrap . Для обробника запитів та створення серверної частини було використно мову сценаріїв JavaScript. База даних була створена за домогою JSON (JavaScript Object Notation).

3.1 Середовище розробки WebStorm

Середовище розробки JetBrains WebStorm була використана при розробці програмного продукту, адже вона дозволяє зручно працювати з кожним із програмних модулей окремо, та об’єднати їх бажаною клієнт-серверною архітектурою.

Програмне забезпечення JetBrains WebStorm являє собою інструмент для розробки веб-сайтів і редагування кодів HTML, CSS і JavaScript. Рішення забезпечує швидку навігацію між файлами та генерує інформацію про виникаючі проблеми в коді в режимі реального часу. JetBrains WebStorm дозволяє додавати розмітку HTML-документів або елементи SQL безпосередньо в JavaScript. Цей інструмент здійснює розгортання та синхронизацію проектів через протокол FTP.

Використовуючи можливості коду HTML/XHTML та XML, WebStorm забезпечує автоматичне завершення стилів, посилань, атрибутів та інших елементів коду. При роботі з CSS забезпечується завершення коду класів, HTML-номерів, ключових слів і т.д. Середовище розробки пропонує автоматичне рішення таких проблем, як вибір формату, властивостей, класів, шляхів до файлів та інших атрибутів CSS. Рішення дозволяє використовувати потужність інструменту кодування Zen для верстки HTML, відображає дії тега на веб-страниці. [13]

Продукт WebStorm здійснює завершення коду JavaScript для ключових слів, лейблів, змінних, параметрів і функцій DOM і підтримує особливості популярних браузерів. Реалізовані в вирішенні функції рефакторингу JavaScript дозволяють покращити структуру коду, файлів і .js. WebStorm забезпечує відновлення коду JavaScript і надає широкий діапазон можливостей: налаштування точки зупинки в HTML і JavaScript, налаштування параметрів точки зупинки, тестування синтаксису коду в режимі реального часу і т.д.

Продукт підтримує платформи JQuery, YUI, Prototype, DoJo, MooTools, Qooxdoo і Bindows. WebStorm передбачає інтегровану перевірку тексту на теги, послідовність коду, помилки в написанні та ін. д. Дане середовище розробки дозволяє редагувати файли та автоматично синхронизувати їх за вимогами при видаленій роботі або зберіганні.

Продукт підтримує функцію контролю версії та альтернативних варіантів коду та фіксує всі вироблені дії та зміни. Завдяки створенню історії в WebStorm можна відновити кодові вирази, блоки та навіть цілі файли. WebStorm підтримує відладку додатків у node.js. Також підтримується повний набір функцій редагування додатків на javascript — як для виконання на сервері, так і в браузері: автодоповнення, навігація за кодом, рефакторинг і перевірка помилок. Для node.js підтримується також виведення повідомлень node.js на окремому вкладку в IDE.

3.2 Мова гіпертекстової розмітки HTML

Мова гіпертекстової розмітки HTML була використана при розробці програмного продукту для формування прикладного програмного модуля, візуальної складової майбутнього сайту, формування лаконічного інтерфейсу з усіма необхідними елементами управління.

HTML, або мова гіпертекстової розмітки, дозволяє веб-користувачам створювати та структурувати розділи, абзаци та посилання за допомогою елементів, тегів та атрибутів. Однак варто зазначити, що HTML не вважається мовою програмування, оскільки він не може створювати динамічні функції.

HTML має багато варіантів використання, а саме:

• Веб-розробка. Розробники використовують HTML-код, щоб розробити, як браузер відображає елементи веб-сторінки, такі як текст, гіперпосилання та медіафайли.

• Інтернет навігація. Користувачі можуть легко переміщатися та вставляти посилання між пов’язаними сторінками та веб-сайтами, оскільки HTML активно використовується для вбудовування гіперпосилань.

• Веб-документація. HTML дозволяє впорядковувати та форматувати документи, подібно до Microsoft Word.

HTML надає засоби для створення заголовків, абзаців, списків, посилань, цитат та інших елементів. Елементи HTML виділяються тегами, записаними з використанням кутових дужок. У простому, загальному випадку, елемент розмітки вказується парою тегів: «початковий тег» і «кінцевий тег» . Текстовий вміст елемента, якщо такий є, розміщується між цими тегами. Теги також можуть містити додаткову розмітку тегів між початком і кінцем, включаючи суміш тегів і тексту. Це вказує на подальші (вкладені) елементи як дочірні елементи батьківського елемента [2].

Початковий тег також може включати атрибути елемента всередині тегу. Вони вказують іншу інформацію, таку як ідентифікатори розділів у документі,

ідентифікатори, які використовуються для прив’язки інформації про стиль до презентації документа, а для деяких тегів, таких як <img>, що використовуються для вбудовування зображень, посилання на ресурс зображення в такий формат: <img src="example.com/example.jpg">. деякі елементи, такі як розрив рядка
 або <br />, не дозволяють вбудовувати будь-який вміст, текст або інші теги. Для них потрібен лише один порожній тег (як початковий тег) і не використовується кінцевий тег. Багато тегів, зокрема закриваючий кінцевий тег для дуже часто використовуваного елемента абзацу <p>, є необов'язковими. Браузер HTML або інший агент може зробити висновок про закриття кінця елемента з контексту та структурних правил, визначених стандартом HTML. Ці правила складні та не зрозумілі більшість програмістів HTML.

Таким чином, загальна форма елемента HTML: <tag attribute1="value1" attribute2="value2">"content"</tag>. Деякі елементи HTML визначаються як порожні елементи і мають форму <tag attribute1="value1" attribute2="value2">. Порожні елементи можуть не містити вмісту, наприклад, тег
 або вбудований тег <img>. Ім'я елемента HTML - це ім'я, яке використовується в тегах. Варто звернути увагу, що назві кінцевого тегу передує символ косої риски /, і що в порожніх елементах кінцевий тег не є ані обов’язковим, ані дозволеним. Якщо атрибути не згадуються, у кожному випадку використовуються значення за замовчуванням. Такі теги, як <img/> та <input/>, безпосередньо вводять контент на сторінку. Інші теги, такі як <p>, оточують і оформляють текст у собі і можуть включати інші теги як от піделементів. Браузери не відображають HTML-теги, але використовують їх для інтерпретації вмісту сторінки.

Щоб додати зовнішній елемент на сторінку (зображення, відео, сценарій тощо), він не вбудовується безпосередньо в код сторінки, а зберігає посилання на розташування зазначеного елемента, доступ здійснюється за допомогою тексту. Таким чином, веб-сторінка містить лише текст, а на веб-браузер покладається завдання об’єднання всіх елементів і відображення кінцевої сторінки.

У HTML можна вбудувати програмний код мовою програмування JavaScript, керувати поведінкою та змістом веб-сторінок. Також включення CSS в HTML описує зовнішній вигляд та макет сторінки.

3.3 Каскадні таблиці стилів CSS

Каскадні таблиці стилів CSS були використана при розробці програмного продукту для візуального оформлення прикладного програмного інтерфейсу, окультурення зовнішнього вигляду сторінки майбутнього сайту для позитивного досвіду користувача, пов’язаним з використанням програмної системи.

Каскадні таблиці стилів, або ж CSS, — це проста мова дизайну, призначена для спрощення процесу створення презентаційних веб-сторінок.

CSS обробляє зовнішній вигляд і відчуття частини веб-сторінки. Використовуючи CSS, можна керувати кольором тексту, стилем шрифтів, інтервалом між абзацами, розміром і розташуванням стовпців, які фонові зображення або кольори використовуються, дизайном макета, варіантами відображення для різних пристроїв і розмірів екрана. а також рядом інших ефектів. CSS легко вивчити та зрозуміти, він забезпечує потужний контроль над поданням HTML-документа. Найчастіше CSS поєднується з мовами розмітки HTML або XHTML. [14]

Розділення форматування та вмісту також дає можливість представити одну й ту саму сторінку розмітки в різних стилях для різних методів відтворення, наприклад, на екрані, у друкованому вигляді, голосом (через мовленнєвий браузер або програму зчитування з екрана) та на основі Брайля. тактильного пристроя. CSS також має правила альтернативного форматування, якщо доступ до вмісту здійснюється на мобільному пристрої.

Каскадність назви походить від заданої схеми пріоритету, щоб визначити, яке правило стилю застосовується, якщо більше одного правила відповідає певному елементу. Ця каскадна схема пріоритетів є передбачуваною.

Переваги CSS:

• CSS економить час — можна написати CSS стиль один раз, а потім повторно використовувати той код на кількох сторінках HTML. Можна окремо визначити стиль для кожного елемента HTML і застосувати його до будь-якої кількості веб-сторінок.

• Сторінки завантажуються швидше — якщо користуватись CSS, то зникає необхідність щоразу писати атрибути тегу HTML. Існує опція написати одне правило CSS для тегу та застосувати його до всіх входів цього тегу. Таким чином, чим менше коду буде використано у програмі, тим швидше буде її завантаження.

• Просте обслуговування — щоб внести глобальні зміни, можна просто змінити стиль, і всі елементи на всіх веб-сторінках оновляться автоматично.

• Кращі стилі ніж поодиноке використання HTML – CSS має набагато ширший набір атрибутів, ніж HTML, тому можна надати набагато кращий вигляд HTML-сторінці, поєднавши її зі стилями, порівняно зі стандартними атрибутами HTML.

• Сумісність кількох пристроїв — таблиці стилів дозволяють оптимізувати вміст для більш ніж одного типу пристроїв. Використовуючи той самий документ HTML, можна представити різні версії веб-сайту для портативних пристроїв, таких як КПК і мобільні телефони, або для друку.

• Глобальні веб-стандарти — на сьогоднішній день атрибути HTML застаріли, і загальноприйнятим є використання саме CSS. Тому інструмент CSS використано на всіх сторінках HTML програмного продукту, щоб зробити їх сумісними із сучасними та майбутніми браузерами. Специфічність - це спосіб, за допомогою якого браузери визначають, які значення властивостей CSS найбільше відповідають елементу і, отже, будуть застосовані. Специфіка базується на правилах відповідності, що складаються з селекторів CSS

різних типів. Специфічність - вага, що надається конкретному правилу CSS. Вага правила визначається кількістю кожного з типів селекторів у цьому правилі. Якщо в кількох правил специфічність однакова, то елемент застосовується останнє по порядку правило CSS. Специфічність має значення лише тому випадку, якщо один елемент відповідає кільком правилам. Згідно специфікації CSS, правило безпосередньо відповідного елемента завжди матиме більший пріоритет, ніж правила, успадковані від предка. [15]

У CSS селектори оголошують, до якої частини розмітки застосовується стиль, зіставляючи теги та атрибути в самій розмітці.

Селектори можуть застосовуватися до:

• всіх елементів певного типу, як от, наприклад, до заголовків секцій другого рівня h 2 (heading 2)

• елементів, визначені атрибутами, зокрема:

• id - унікальний ідентифікатор у документі, ідентифікований за допомогою хеш-префікса, наприклад. #ідентифікатор

• клас: ідентифікатор, який може анотувати декілька елементів у документі, ідентифікований префіксом крапки, наприклад. .classname

• елементів залежно від того, як вони розміщені відносно інших у дереві документа.

Класи та ідентифікатори чутливі до регістру, починаються з літер і можуть включати буквено-цифрові символи, дефіси та підкреслення. Клас може застосовуватися до будь-якої кількості екземплярів будь-яких елементів. Ідентифікатор можна застосувати лише до одного елемента.

Псевдокласи використовуються в селекторах CSS, щоб дозволити форматування на основі інформації, яка не міститься в дереві документа. Одним із прикладів широко використовуваного псевдокласу є :hover, який ідентифікує вміст лише тоді, коли користувач «вказує на» видимий елемент, зазвичай утримуючи на ньому курсор миші. Він додається до селектора, як у a:hover або #elementid:hover. Псевдоклас класифікує елементи документа, такі як :link або :visited, тоді як

псевдоелемент робить вибірку, яка може складатися з часткових елементів, таких як ::first-line або ::first-letter.

Селектори можна комбінувати багатьма способами для досягнення великої специфічності та гнучкості. Декілька селекторів можна об’єднати в список з інтервалами, щоб вказати елементи за розташуванням, типом елемента, ідентифікатором, класом або будь-якою їх комбінацією. Важливим є порядок селекторів. Наприклад, div .myClass {color: red;} застосовується до всіх елементів класу myClass, що знаходяться всередині елементів div, тоді як .myClass div {color: red;} застосовується до всіх елементів div, які знаходяться всередині елементів класу myClass. Це не слід плутати з конкатенованими ідентифікаторами, такими як div.myClass {color: red;}, який застосовується до елементів div класу myClass.

3.4 Мова програмування JavaScript

Мова JavaScript була використана при розробці програмного продукту для створення програмного модуля обробника запитів, а також для написання серверу з метою хостингу майбутнього сайту і створення логіки зв’язку клієнт-сервер.

JavaScript - це динамічна мова комп'ютерного програмування. Він легкий і найчастіше використовується як частина веб-сторінок, чиї реалізації дозволяють клієнтському сценарію взаємодіяти з користувачем і створювати динамічні сторінки. Це інтерпретована мова програмування з об'єктно-орієнтованими можливостями.

Спочатку JavaScript був відомий як LiveScript, але Netscape змінив назву на JavaScript, можливо, через хвилювання, викликане Java. Вперше JavaScript з’явився в Netscape 2.0 у 1995 році під назвою LiveScript. Основне ядро мови загального призначення було вбудовано в Netscape, Internet Explorer та інші веб-браузери.

Специфікація ECMA-262 визначає стандартну версію основної мови JavaScript, надаючи наступних характеристик:

• JavaScript — це легка інтерпретована мова програмування.

• Мова призначена для створення мережевих програм.

• Доповнює та інтегрується з Java.

• Доповнює та інтегрується з HTML.

• Відкрита та кросплатформна [3]

Клієнтський JavaScript є найпоширенішою формою мови. Сценарій повинен бути включений в HTML-документ або посилання на нього, щоб код інтерпретував браузер. Це означає, що веб-сторінка не обов'язково має бути статичним HTML, але може включати програми, які взаємодіють з користувачем, керують браузером і динамічно створюють вміст HTML.

Механізм на стороні клієнта JavaScript надає багато переваг перед традиційними серверними сценаріями CGI. Наприклад, можна використовувати JavaScript, щоб перевірити, чи ввів користувач дійсну адресу електронної пошти в поле форми. Код JavaScript виконується, коли користувач надсилає форму, і лише якщо всі записи дійсні, вони будуть передані на веб-сервер.

JavaScript можна використовувати для захоплення ініційованих користувачем подій, таких як натискання кнопок, навігація за посиланнями та інші дії, які користувач ініціює явно чи неявно.

Перевагами використання JavaScript є:

• Менше взаємодії з сервером —можна перевірити введення користувача перед відправкою сторінки на сервер. Це економить трафік сервера, що означає менше навантаження на сервер.

• Негайний зворотній зв'язок із відвідувачами – їм не потрібно чекати перезавантаження сторінки, щоб побачити, чи не забули вони щось ввести.

• Підвищена інтерактивність — JavaScript надає змогу створювати інтерфейси, які реагують, коли користувач наводить на них курсор миші або активує їх за допомогою клавіатури.

• Розширені інтерфейси – можна використовувати JavaScript, щоб включити такі елементи, як компоненти перетягування й повзунки, щоб надати відвідувачам сайту збагачений функціоналом інтерфейс.

Не можна розглядати JavaScript як повноцінну самостійну мову програмування. У ньому відсутні наступні важливі функції:

• JavaScript на стороні клієнта не дозволяє читати або записувати файли. Це було збережено з міркувань безпеки.

• JavaScript не можна використовувати для мережевих програм, оскільки така підтримка недоступна.

• JavaScript не має жодних багатопоточних чи багатопроцесорних можливостей.

Знову ж таки, JavaScript — це легка, інтерпретована мова програмування, яка дозволяє вбудувати інтерактивність у статичні сторінки HTML. Однією з основних переваг JavaScript є те, що він не вимагає дорогих інструментів розробки. Можна почати з простого текстового редактора, такого як Блокнот. Оскільки це інтерпретована мова в контексті веб-браузера, не потрібно купувати компілятор. Одним із інструментів роботи з JavaScript є npm.

3.5 npm

Менеджер npm був використаний при розробці програмного продукту з метою підтримки актуальних пакетів, бібліотек, найновіших інструментів мови JavaScript. За допомогою npm проект був зібраний у єдину програмну систему.

npm (спочатку скорочення від Node Package Manager)[4] — це менеджер пакетів для мови програмування JavaScript, що підтримується npm, Inc. npm — менеджер пакетів за замовчуванням для середовища виконання JavaScript Node.js. Він складається з клієнта командного рядка, який також називається npm, і онлайнової бази даних загальнодоступних і платних приватних пакетів, що

називається реєстром npm. Доступ до реєстру здійснюється через клієнт, а доступні пакети можна переглядати та шукати на веб-сайті npm. Менеджером пакетів і реєстром керує npm, Inc. [16]

npm — це дві речі: перш за все, це онлайн-сховище для публікації проектів Node.js з відкритим кодом; по-друге, це утиліта командного рядка для взаємодії із зазначеним репозиторієм, яка допомагає інсталювати пакети, керувати версіями та керувати залежностями. Безліч бібліотек і програм Node.js публікується на npm, і багато інших додаються щодня. Коли існує пакет, який користувач хоче інсталювати, його можна встановити за допомогою однієї команди командного рядка.

npm включено як рекомендовану функцію в інсталятор Node.js. npm складається з клієнта командного рядка, який взаємодіє з віддаленим реєстром. Це дозволяє користувачам використовувати та поширювати модулі JavaScript, які доступні в реєстрі. Пакети в реєстрі мають формат CommonJS і містять файл метаданих у форматі JSON. У головному реєстрі npm доступно понад 1,3 мільйона пакетів. Реєстр не має жодного процесу перевірки для подання, а це означає, що знайдені там пакети потенційно можуть бути низької якості, небезпечними або шкідливими. Замість цього npm покладається на звіти користувачів, щоб видалити пакунки, якщо вони порушують політику через низьку якість, незахищеність або шкідливість. npm надає статистичні дані, включаючи кількість завантажень і кількість залежних пакетів, щоб допомогти розробникам оцінити якість пакунків.

npm може керувати пакетами, які є локальними залежностями конкретного проекту, а також глобально встановленими інструментами JavaScript. При використанні npm як менеджер залежностей для локального проекту, npm може за одну команду встановити всі залежності проекту через файл package.json. У файлі package.json кожна залежність може вказати діапазон дійсних версій за допомогою семантичної схеми керування версіями, що дозволяє розробникам автоматично оновлювати свої пакунки, водночас уникаючи небажаних змін, що порушують роботу. npm також надає інструменти визначення версії для

розробників, щоб позначити свої пакунки певною версією. npm також надає файл package-lock.json, який містить точну версію, яку використовує проект після оцінки семантичного керування версіями в package.json. [17]

3.6 Node.js

Середовище Node.js було використане при розробці програмного продукту для запуску, швидкої та коректної роботи програмного продукту, створення сценаріїв на стороні сервера задля хостингу сайту, що становитиме клієнт-серверну логіку.

Node.js — це міжплатформне середовище виконання JavaScript із відкритим вихідним кодом, яке працює на «двигуні» V8 і виконує код JavaScript за межами веб-браузера. Node.js дозволяє розробникам використовувати JavaScript для написання інструментів командного рядка та для сценаріїв на стороні сервера — запуск сценаріїв на стороні сервера для створення динамічного вмісту веб-сторінки, перш ніж сторінка буде відправлена у веб-браузер користувача.

Отже, Node.js являє собою парадигму «JavaScript всюди», що об’єднує розробку веб-додатків навколо однієї мови програмування, а не різних мов для сценаріїв на стороні сервера та клієнта. Node.js має архітектуру, керовану подіями, з можливістю асинхронного введення-виведення. Ці варіанти дизайну спрямовані на оптимізацію пропускної здатності та масштабованості у веб-додатках з багатьма операціями введення/виводу, а також для веб-додатків у режимі реального часу (наприклад, комунікаційні програми в режимі реального часу та браузерні ігри).

Node.js дозволяє створювати веб-сервери та мережеві інструменти за допомогою JavaScript і набору «модулів», які обробляють різні основні функції. Надаються модулі для вводу-виводу файлової системи, мереж (DNS, HTTP, TCP, TLS/SSL або UDP), двійкових даних (буферів), функцій криптографії, потоків

даних та інших основних функцій. Модулі Node.js використовують API, призначений для зменшення складності написання серверних додатків.

JavaScript є єдиною мовою, яку Node.js підтримує початково, але є багато мов компіляції в JS. В результаті програми Node.js можна писати на CoffeeScript, Dart, TypeScript, ClojureScript та інших. Node.js в основному використовується для створення мережевих програм, таких як веб-сервери. Найбільш істотна відмінність між Node.js і PHP полягає в тому, що більшість функцій у PHP блокують до завершення (команди виконуються лише після завершення попередніх команд), тоді як функції Node.js не блокують (команди виконуються одночасно або навіть паралельно, і використовують зворотні виклики, щоб повідомити про завершення або помилку).

Node.js офіційно підтримується в Linux, macOS і Microsoft Windows 8.1 і Server 2012 (і пізніших версіях), з підтримкою рівня 2 для SmartOS і IBM AIX і експериментальною підтримкою FreeBSD. OpenBSD також працює, і версії LTS доступні для IBM i (AS/400). Наданий вихідний код також може бути побудований на операційних системах, подібних до тих, що офіційно підтримуються, або модифікований третіми сторонами для підтримки інших, таких як NonStop OS і сервера Unix.

Node.js надає веб-серверам програмування на основі подій, що дозволяє розробляти швидкі веб-сервери на JavaScript. Розробники можуть створювати масштабовані сервери без імплементації потоків, використовуючи спрощену модель програмування на основі подій, які спричинять зворотні виклики для сигналу про завершення завдання. Node.js поєднує простоту мови сценаріїв (JavaScript) з потужністю мережевого програмування Unix.

Node.js був побудований на основі «двигуна» Google V8 JavaScript, оскільки він був відкритим під ліцензією BSD. Він володіє основами Інтернету, такими як HTTP, DNS і TCP. JavaScript також був добре відомою мовою, що робило Node.js доступним для спільноти веб-розробників.

3.7 Фреймворк Bootstrap

Фреймворк Bootstrap було використано при розробці програмного продукту з метою оформлення дизайну основних навігаційних компонентів керування веб-інтерфейсу програмної системи.

Bootstrap (також відомий як Twitter Bootstrap) — вільний набір інструментів для створення сайтів та веб-застосунків. Включає HTML- і CSS-шаблони оформлення для типографіки, веб-форм, кнопок, міток, блоків навігації та інших компонентів веб-інтерфейсу, включаючи JavaScript-розширення. Bootstrap використовує сучасні напрацювання в області CSS та HTML, тому необхідно бути уважним за підтримки старих браузерів. Він містить шаблони дизайну на основі HTML, CSS і JavaScript для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу.

Основні інструменти Bootstrap:

• Сітки — заздалегідь задані розміри колонок, які можна відразу використовувати, наприклад, ширина колонки 140 px відноситься до класу .span2 (.col-md-2 в третій версії фреймворку), який можна використовувати в CSS-описанні документа.

• Шаблони – фіксований або адаптивний шаблон документа.

• Типографіка - опис шрифтів, визначення деяких класів для шрифтів, таких як код, цитати тощо.

• Медіа — надає певне керування зображеннями та відео. Таблиці — засоби оформлення таблиць, до додавання функціональності сортування.

• Форми - класи для оформлення форм та деяких подій, що відбуваються з ними.

• Навігація – класи оформлення для панелей, вкладок, переходу по сторінках, меню та панелі інструментів.

• Алерти - оформлення діалогових вікон, підказок та спливаючих вікон.

Bootstrap — це бібліотека HTML, CSS та JS, яка зосереджена на спрощенні розробки інформаційних веб-сторінок (на відміну від веб-програм). Основна мета його додавання до веб-проекту — застосувати вибір кольору, розміру, шрифту та макета Bootstrap до цього проекту. Таким чином, основним фактором є те, чи знайдуть відповідальні розробники ці варіанти до смаку. [18]

Після додавання до проекту Bootstrap надає базові визначення стилів для всіх елементів HTML. Результатом є уніфікований вигляд таблиць і елементів форм у всіх веб-браузерах. Крім того, розробники можуть скористатися перевагами класів CSS, визначених у Bootstrap, щоб додатково налаштувати зовнішній вигляд свого вмісту. Наприклад, у Bootstrap передбачені таблиці світлих і темних кольорів, заголовки сторінок, більш помітні лапки та текст із виділенням.

Bootstrap також постачається з кількома компонентами JavaScript у вигляді плагінів jQuery. Вони забезпечують додаткові елементи інтерфейсу користувача, такі як діалогові вікна, підказки. Кожен компонент Bootstrap складається з HTML-структури, декларацій CSS і в деяких випадках супровідного коду JavaScript. Вони також розширюють функціональні можливості деяких існуючих елементів інтерфейсу, включаючи, наприклад, функцію автозаповнення для полів введення.

Найпомітнішими компонентами Bootstrap є його компоненти макета, оскільки вони впливають на всю веб-сторінку. Основний компонент макета називається «Контейнер», оскільки в ньому розміщуються всі інші елементи сторінки. Розробники можуть вибирати між контейнером фіксованої ширини та контейнером рідинної ширини. У той час як останній завжди заповнює ширину веб-сторінки, перший використовує одну з п’яти попередньо визначених фіксованих ширин, залежно від розміру екрана, на якому відображається сторінка:

- Менше 576 пікселів
- 576–768 пікселів
- 768–992 пікселів
- 992–1200 пікселів
- Більше 1200 пікселів

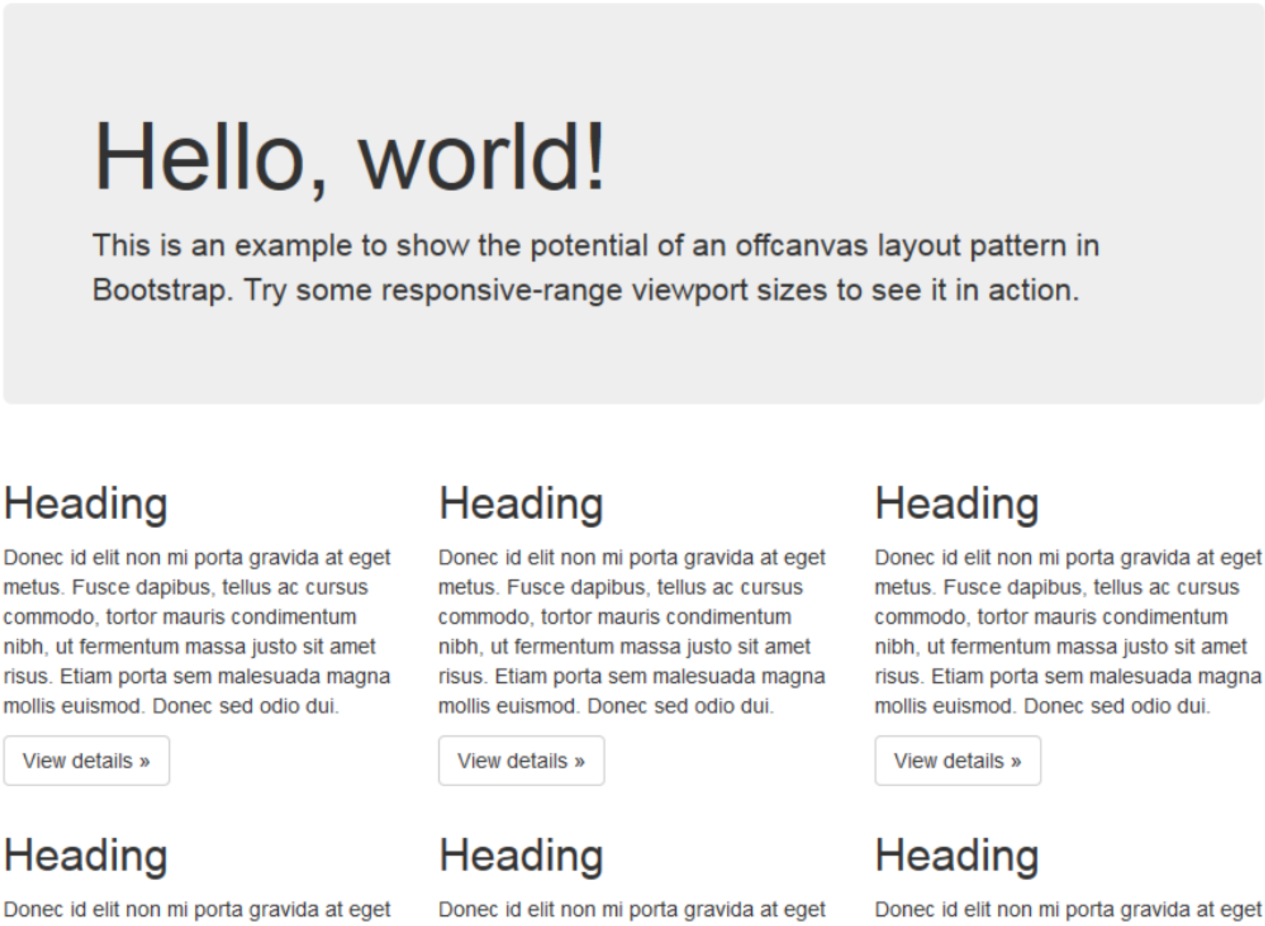

Рисунок 3.7 Приклад сайту, при дизайні якого використали фреймворк BootStrap

Після встановлення контейнера інші компоненти макета Bootstrap реалізують макет CSS Flexbox за допомогою визначення рядків і стовпців. Попередньо скомпільована версія Bootstrap доступна у вигляді одного файлу CSS і трьох файлів JavaScript, які можна легко додати до будь-якого проекту. Однак необроблена форма Bootstrap дозволяє розробникам впроваджувати подальші налаштування та оптимізацію розміру. Ця необроблена форма є модульною, що означає, що розробник може видалити непотрібні компоненти, застосувати тему та змінити нескомпільовані файли Sass.

Підсумовуючи, можна зробити висновок, що обрані для розробки програмного продукту інструменти відповідають усім вимогам і потребам.

Використання вищевказаних мов є обґрунтованим, а середовище розробки та супутні їй технології поліпшать реалізацію системи, дозволять у повній мірі впровадити клієнт-серверну архітектуру.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Розробка програмного продукту передбачає наступні кроки:

• Створення прикладного програмного інтерфейсу

• Створення обробника запитів

• Створення бази даних

• Побудова клієнт-серверної архітектури.

4.1 Опис архітектури

Система складається з наступних компонентів:

• Frontend – користувацький веб-інтерфейс;

• Backend – обробник запитів;

• База даних;

• Сервер для взаємодії компонентів системи.

Опис архітектури додатку показаний на блок-схемі системи (рисунок 4.1).

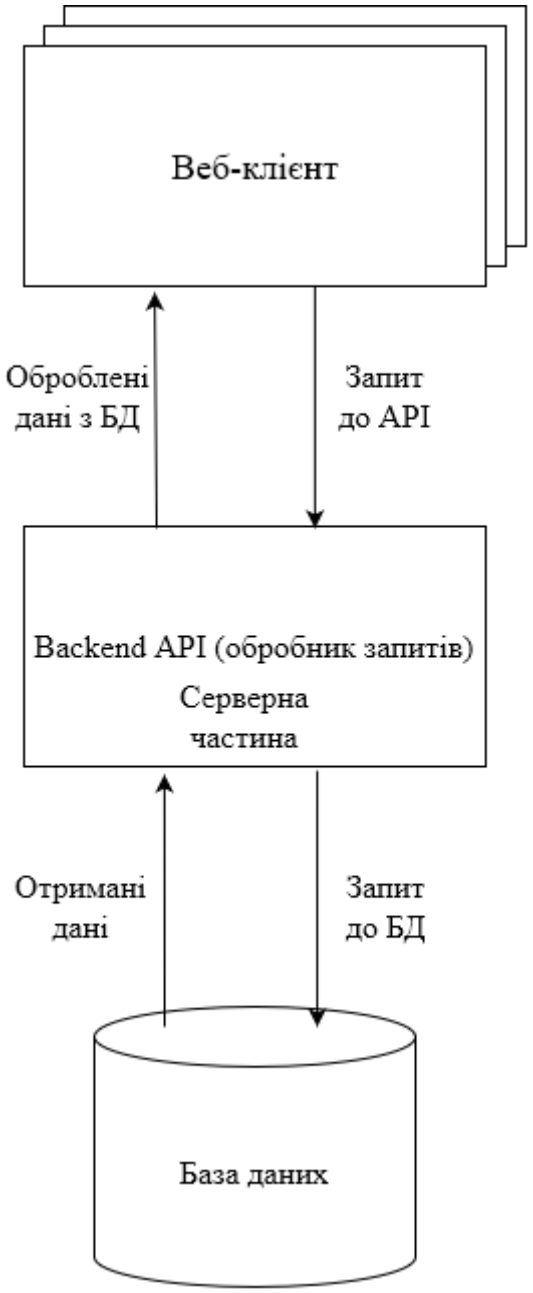

Рисунок 4.1 – Блок-схема веб-системи

Користувацький веб-інтерфейс створений за допомогою розмітки HTML, дизайнів CSS і фреймворка Bootstrap.

Обробник запитів написаний мовою JavaScript на технології node.js, і складається з наступних структурних частин:

1. Перевірка правильності введених даних;
2. Операції над даними з БД

4.1.1 Перевірка правильності введених даних

Дана частина програмного продукту відповідає за валідацію полів перед внесенням відповідних атрибутів до бази даних, з метою попередження не коректного типу даних або пустого атрибуту, що може призвести до помилки на стороні БД. Наприклад, поле «Id» для об’єкту «фільм», атрибут якого стане первинним ключем в базі даних, повинно бути обов’язково заповнено. Власні імена, дати та url-посилання мають свій особливий формат – кожен з яких необхідно перевірити, перш ніж давати запит на занесення у БД, адже дані повинні відповідати закладеному формату, для коректної роботи системи.

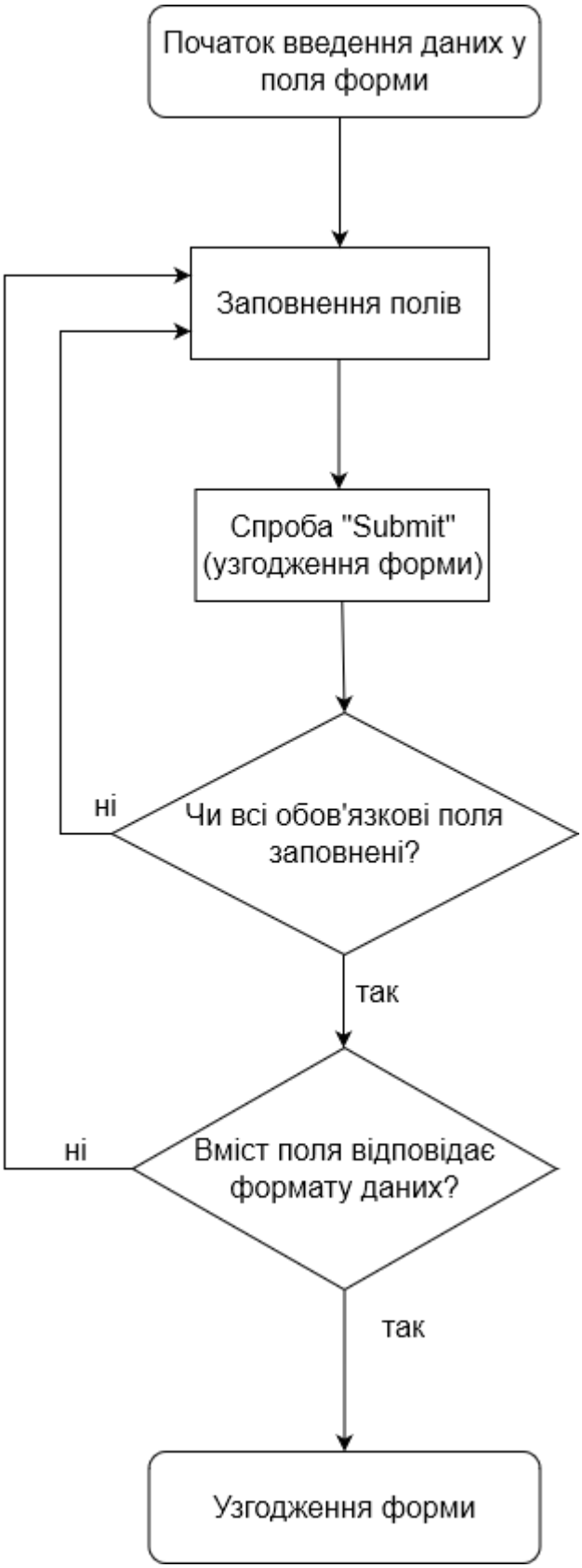

Рисунок 4.1.1 – Блок-схема алгоритма валідації полів на формі

4.1.2 Динамічна вичитка даних з БД

При побудові модуля обробки даних БД, було використано асинхронний підхід AJAX із включенням необхідних бібліотек та використанням необхідних вбудованих функцій для реалізації методології. Для кожного процесу додавання, оновлення або видалення використовується логіку виклику AJAX для обробки.

AJAX (Asynchronous JavaScript And XML)— підхід до побудови користувацьких інтерфейсів веб-застосунків, за яких веб-сторінка, не перезавантажуючись, у фоновому режимі надсилає запити на сервер і сама звідти довантажує потрібні користувачу дані. AJAX— один з компонентів концепції DHTML. Використання цих підходів дозволяє створювати набагато зручніші веб-інтерфейси користувача на тих сторінках сайтів, де необхідна активна взаємодія з користувачем. AJAX— асинхронний, тому користувач може переглядати далі контент сайту, поки сервер все ще обробляє запит. Браузер не перезавантажує web-сторінку і дані посилаються на сервер без візуального підтвердження (крім випадків, коли ми самі захочемо показати процес з'єднання з сервером) [11]

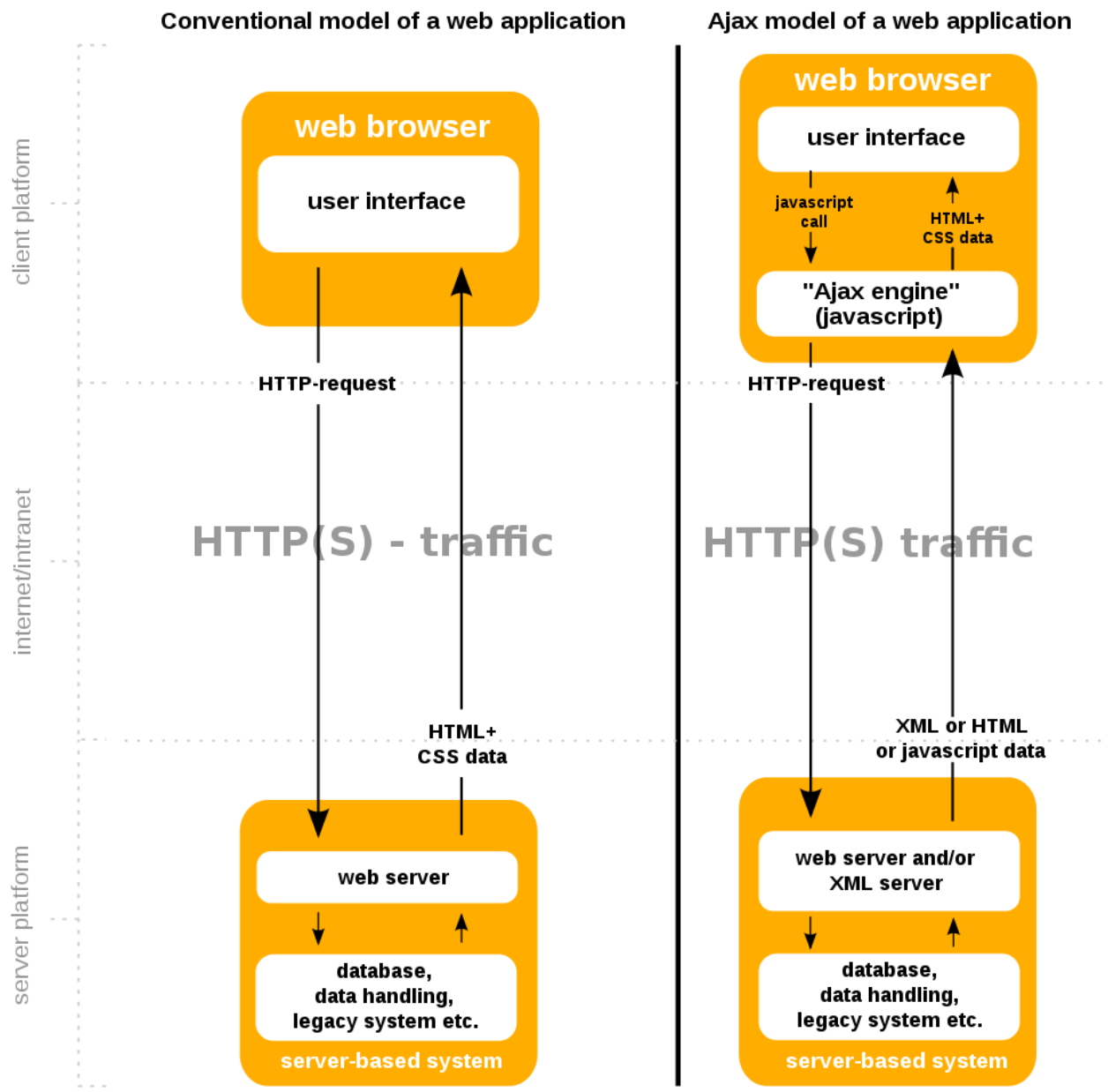

Рисунок 4.1.2 – Модель класичних застосунків для мережі (зліва) в прямому порівнянні із застосуванням AJAX (праворуч) [12]

4.2 Опис алгоритму програмної системи

1. Головна сторінка – на яку потрапляє користувач коли заходить на сайт, на ній відображено поточний каталог фільмів.

2. Сторінка додавання фільмів – на яку попадає користувач, коли хоче додати новий фільм. Він заповнює всі необхідні дані для нової кінострічки, дані валідуються обробником запитів
3. Сторінка оновлення даних про фільм – сюди користувач попадає, коли хоче поновити фільм. Валідуються вхідні дані, оновлюються лише введені поля, тобто змінюються лише бажані користувачем атрибути кінострічки.
4. Спливаюче вікно для редагування акторського складу – відображається поверх головної сторінки, користувач має змогу додати дані про нового актора, які також валідуються, або видалити наявного.
5. Інформація заноситься у базу даних .json, в якій в режимі реального часу оброблюється і поновлюється.
6. Користувач автоматично попадає на головну сторінку, де вже відображаються зміни, підтягнуті з сервера
7. Користувач має змогу перевірити базу даних у вигляді атрибутів JSON, за рахунок хосту бази сервером.

4.3 USE-CASE діаграма

У системі є один актор: користувач.

Користувач має наступні опції:

- Перегляд каталогу фільм,
- Сортування фільмів за параметрами,
- Додавання фільмів,
- Редагування фільмів,
- Створення нових акторів у обраному фільмі,
- Видалення актора,
- Видалення фільму.

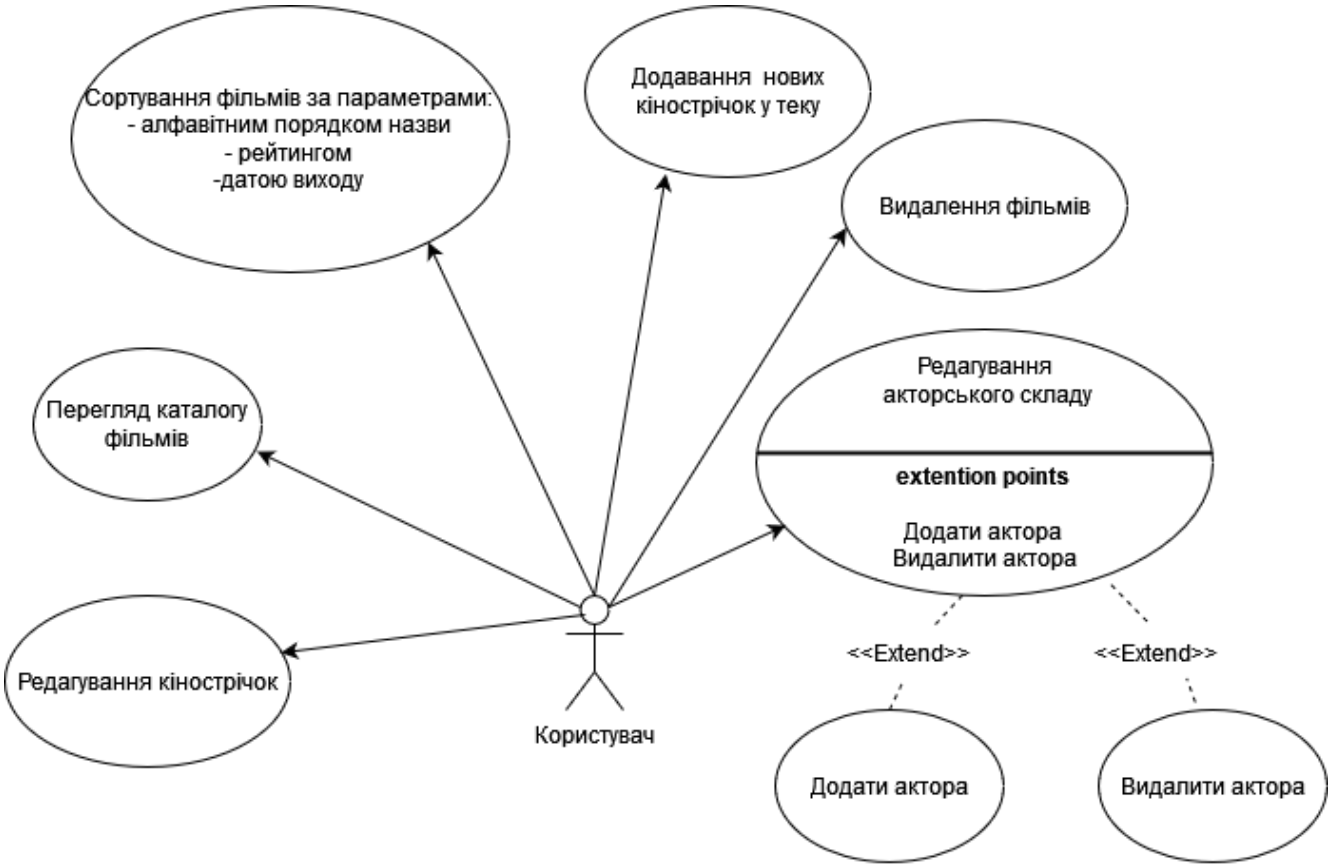

Рисунок 4.3 Use-case діаграма

4.4 Опис схем даних

В базі даних було створено 4 таблиці, кожна з яких містить свій набір даних. Нижче представлена модель бази даних веб-порталу (рисунок 4.4).

Таблиця “Film” містить основну інформацію про фільм, а саме його назву, обкладинку, режисера, рейтинг, наявність серій, дату випуску.

Таблиця “ActorList” являє собою набір акторів.

Таблиця “Actor” містить основну інформацію про актора: картинка, ім’я, посилання на сайт із додатковою інформацією.

Таблиця “SeriesList” зберігає кількість епізодів.

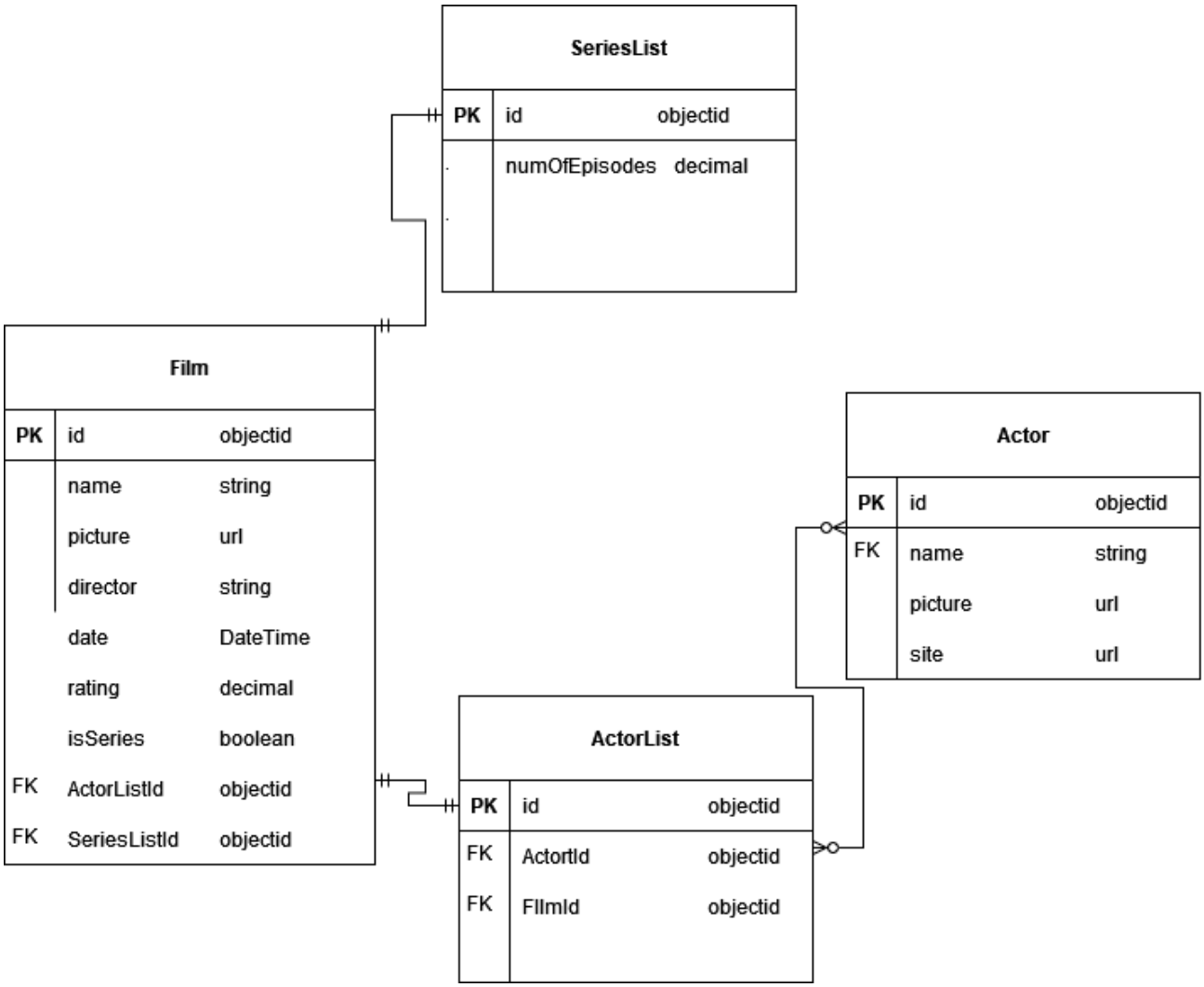

Рисунок 4.4 Модель бази даних

Отже, можна зробити висновок, що система проаналізована, має чіткий план для розробки, втілення його в життя.

5 РОБОТА КОРИСТУВАЧА З ІНТЕРФЕЙСОМ

5.1 Системні вимоги

Для коректної роботи з кінцевим продуктом необхідне виконання наступних мінімальних системних вимог:

1. Об’єм оперативної пам’яті (RAM) – 1024 МБ для Windows 7, 4098 МБ для Windows 11.
2. Операційна система Windows 7/10/11, Linux, MacOS.
3. Процесор Intel Pentium 4 або більш пізньої версії з підтримкою SSE3.

На комп’ютері повинне бути встановлене наступне програмне забезпечення:

Інтернет браузер: Google Chrome/Opera/Mozilla Firefox/Safari.

5.2 Сценарій взаємодії користувача з інтерфейсом

5.2.1 Перший вхід на сайт

Користувач переходить за посиланням http://localhost:3001/list, яке виводиться в терміналі середовища розробки при успішному запуску програмного продукту. Це буде адресою URL хостинга сайту веб-порталу, таким чином, користувач опиниться на головній сторінці.

Головна сторінка веб-порталу включає в себе:

• Шапку з умовною назвою сайту

• Опис вмісту стовпчиків

• Рядки з інформацією по конкретному фільму: ID, назва, обкладинка, рейтинг, дата виходу, кнопки керування медіа.

• Для кожного фільму кнопки операцій: «Видалити», «Оновити», «Додати актора», «Список акторів»

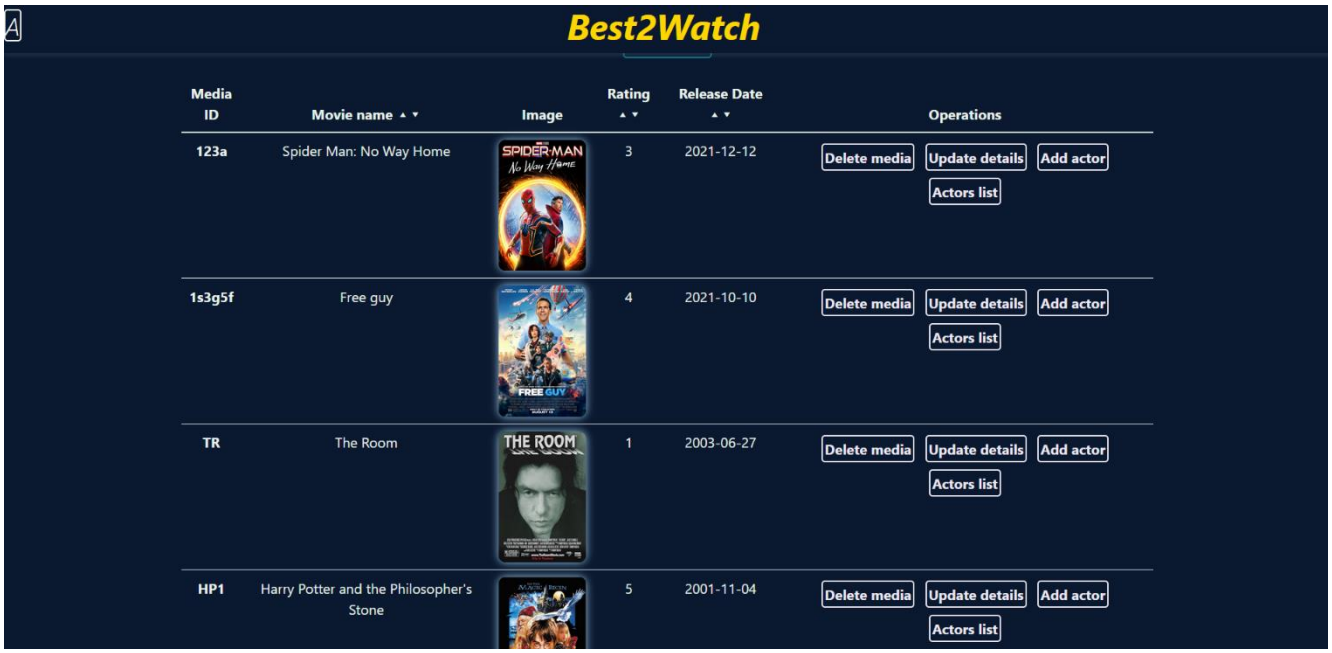

Рисунок 5.2.1 Головна сторінка сайту (за замовчуванням)

5.2.2 Сортування за датою

Найменування стовпчику “Rating” є кнопкою для сортування фільмів. За замовчуванням, фільми відсортовані по даті виходу, від найновіших до найстаріших.

При натисканні цієї кнопки фільмі відсортуються навпаки – від найстаріших до найновіших. При повторному натисканні – повернуться у стан «від найновіших до найстаріших».

Тобто, при кожному наступному натисканні – стан змінюватиметься на зворотно-протилежний, і так для кожного типу сортування. На рисунку 5.2.2 бачимо результат сортування:

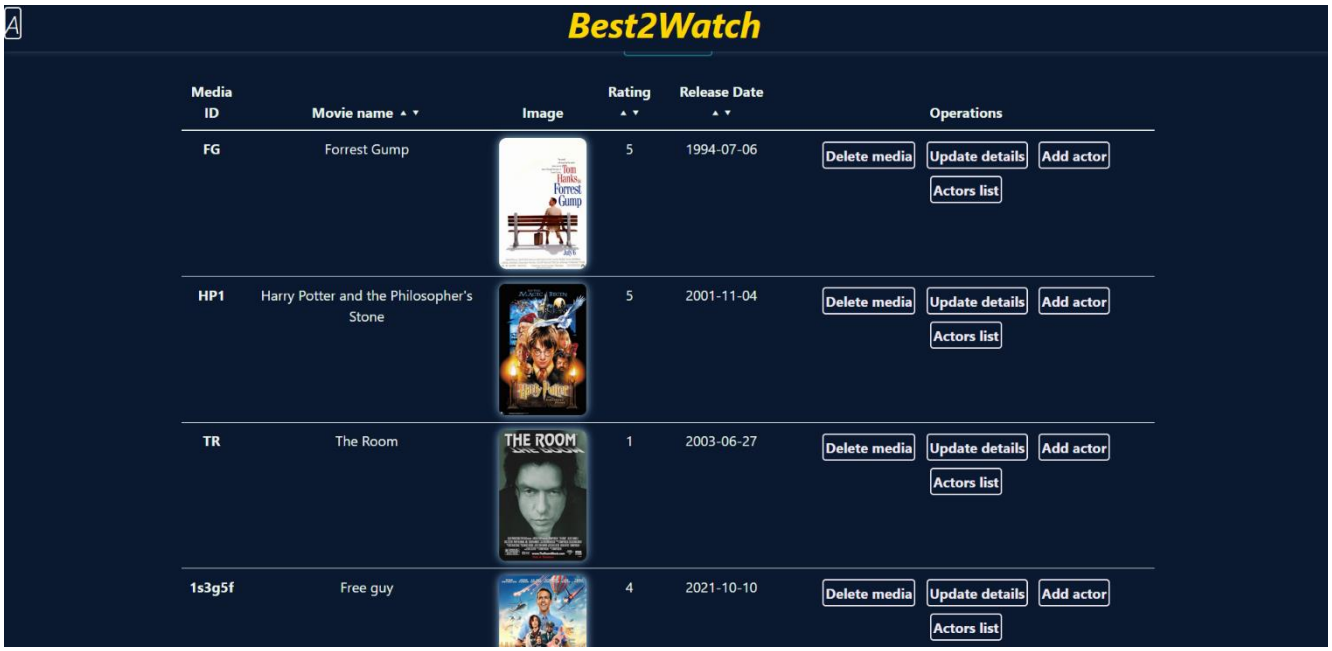

Рисунок 5.2.2 Головна сторінка сайту (фільми відсортовано по даті виходу - від найстаріших до найновіших)

5.2.3 Сортування за алфавітом

Найменування стовпчику “Name” є кнопкою для сортування фільмів за алфавітним порядком. Як і в попередньому пункті, існують два взаємо-замінні стани: сортування по порядку від «a» до «z», і навпаки, від «z» до «a». Звісно ж, якщо фільми починаються з однакової букви – їх буде відсортовано по другій, якщо і другі букви однакові, то сортування відбуватиметься по третій, і так далі.

Отже, при натисканні кнопки фільми відсортуються відповідно назв за алфавітом порядком. На рисунку 5.2.3 можна побачити результат виконання сортування:

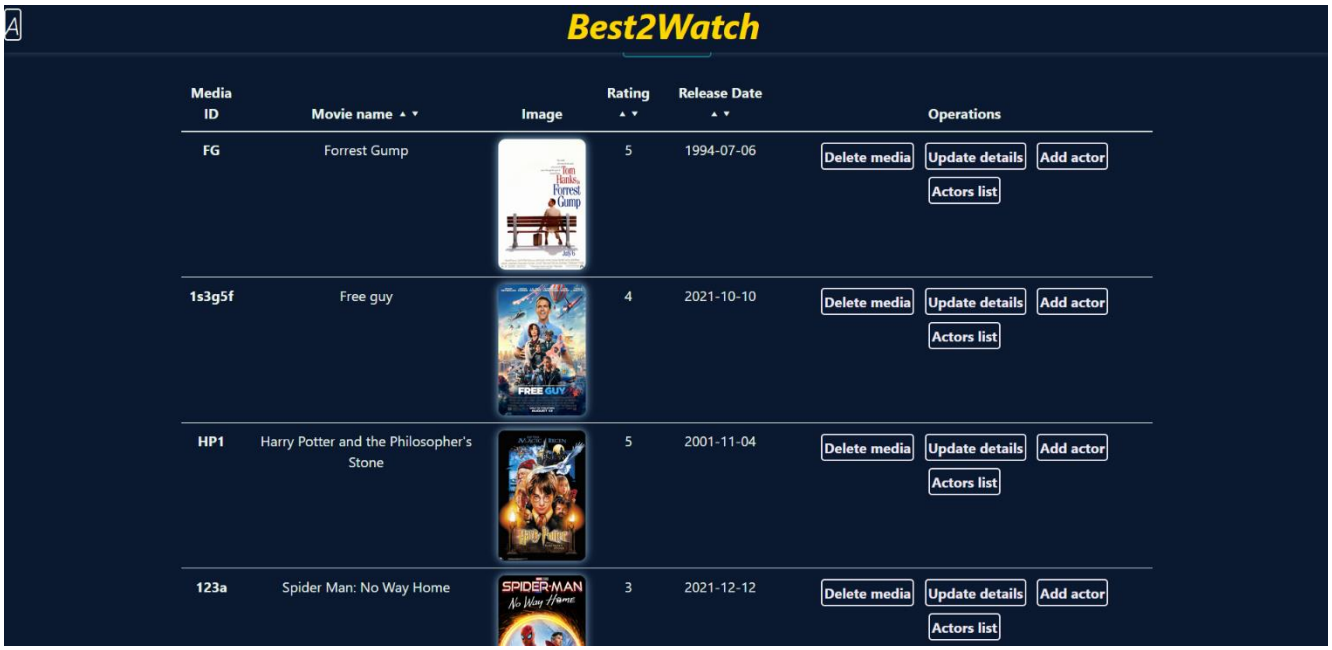

Рисунок 5.2.3 Головна сторінка сайту (фільми відсортовано по алфавіту відносно назви - від «a» до «z»).

5.2.4 Сортування за рейтингом два рази

Найменування стовпчику “Rating” також є кнопкою для сортування фільмів за алфавітним порядком. По аналогії із попередніми пунктами, існують два взаємозамінні стани:

• При першому сортуванні фільми вистроятьсяся від найгіршого до найкращого (від оцінки «1» до «5»).

• При повторному сортуванні - від найкращого до найгіршого (від «5» до «1» відповідно).

Нижче, на рисунку 5.2.4 наведено приклад обох станів сортування по рейтингу фільмів:

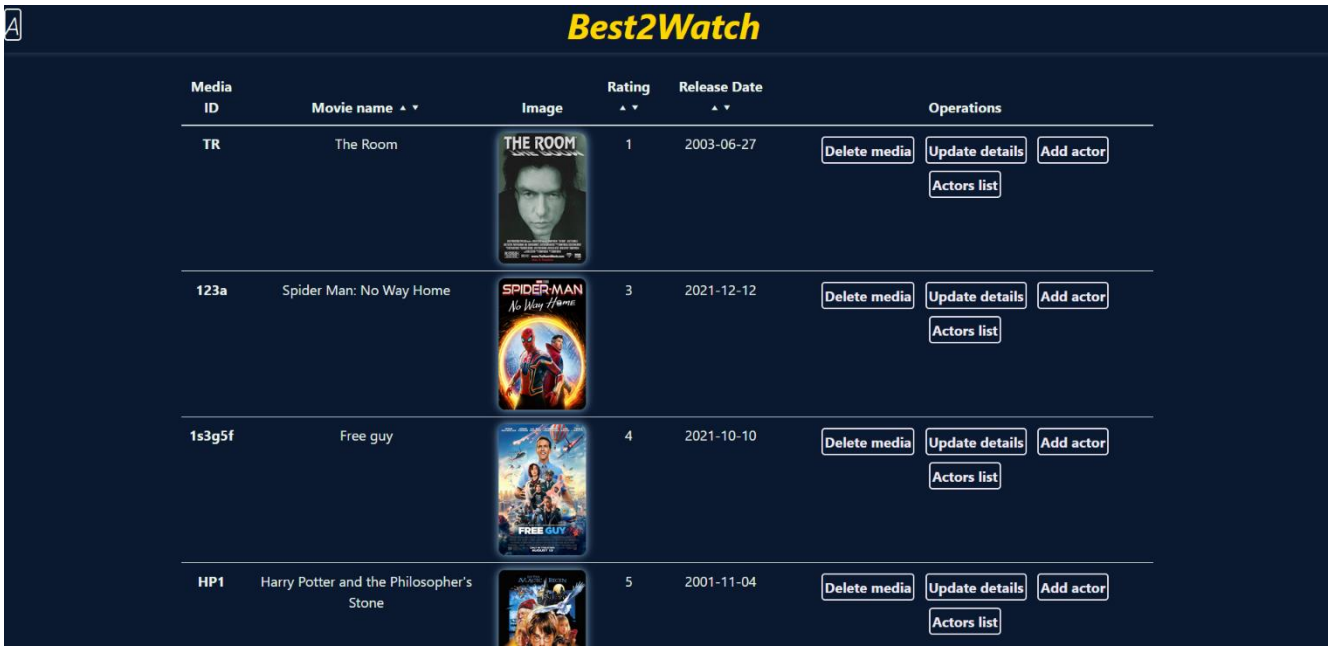

Рисунок 5.2.4 Фільми відсортовано по рейтингу - від найгіршого до найкращого.

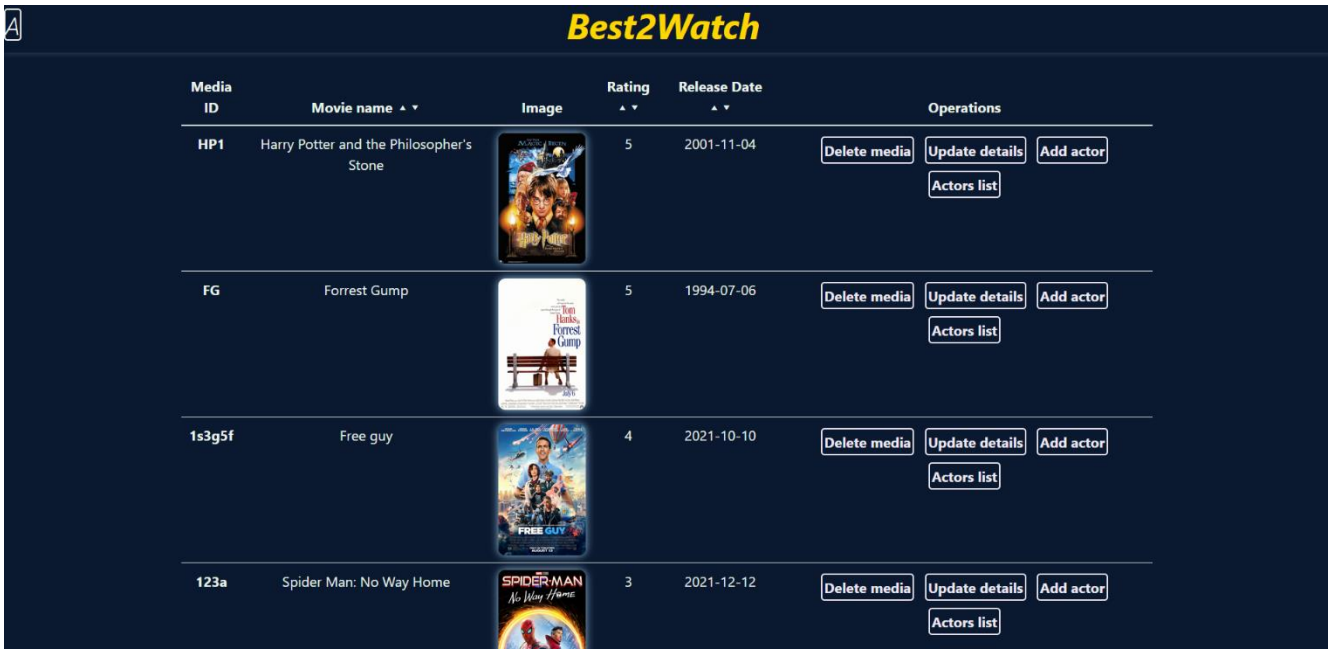

Рисунок 5.2.4.2 Фільми відсортовано по рейтингу - від найкращого до найгіршого.

5.2.5 Додавання актора

При натисканні кнопки «Add actor» у полі бажаного фільму, то відкриється сторінка для додання об’єкта «актор» в акторський склад об’єкта «фільм». У формі наявні поля «Name» (ім’я актора), «Image(url)» (посилання на картинку, що слугуватиме обкладинкою, портретом актора), «fan site» (посилання на сайт, де можна дізнатися додаткову інформацію про даного актора):

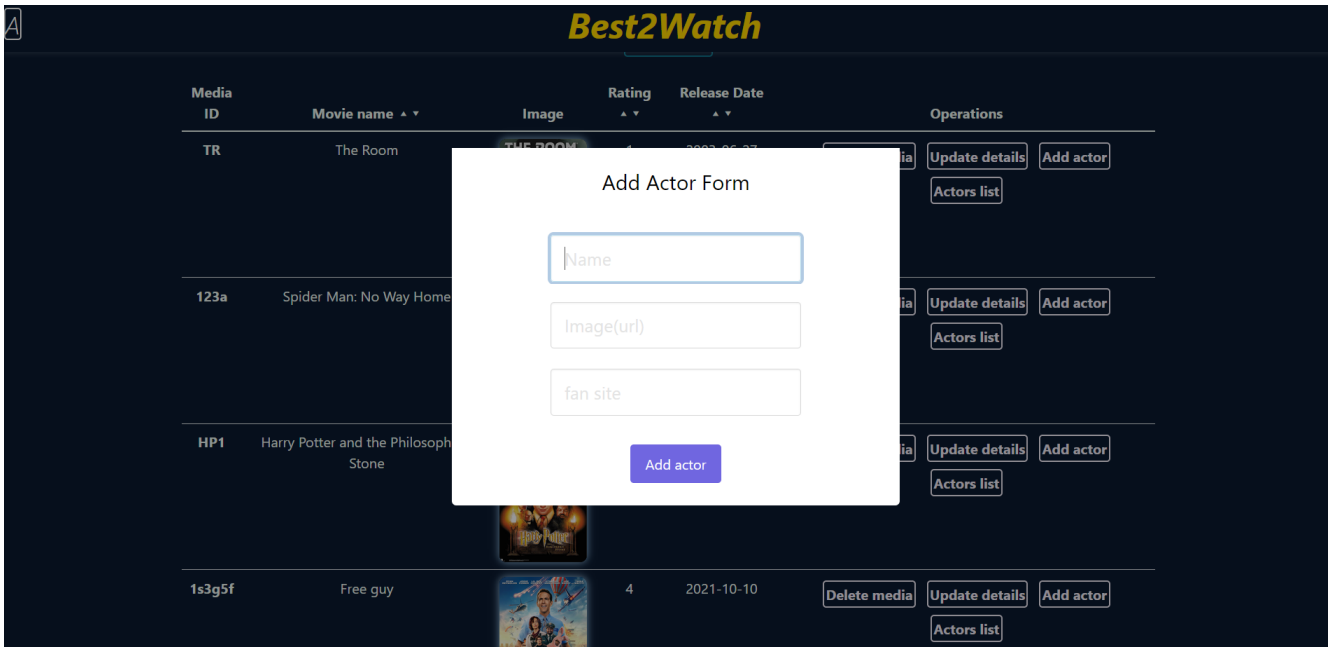

Рисунок 5.2.5 Вікно додання акторів

5.2.6 Заповнення форми додання акторів

При натисканні кнопки «Add actor» у полі бажаного фільму, то відкриється сторінка для внесення необхідних полів для актора. Користувач вводить ім’я, URL картинки та сайту з додатковою інформацією:

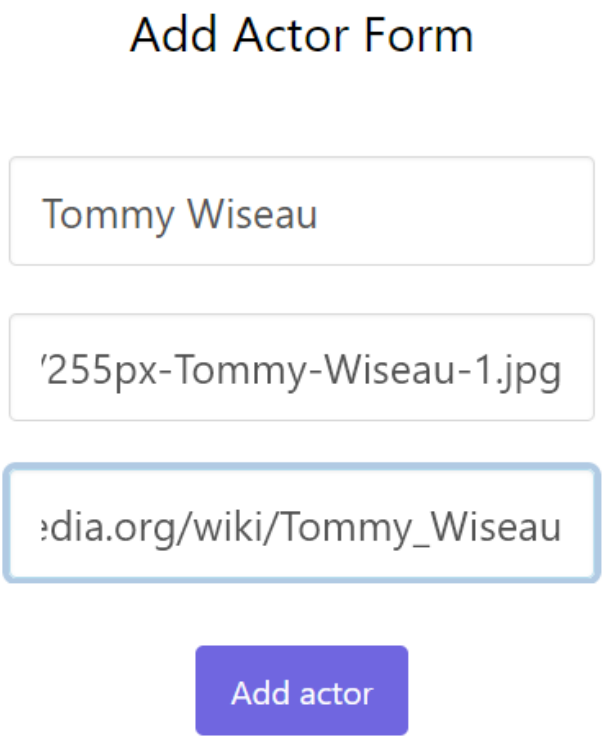

Рисунок 5.2.6 Заповнена форма на вікні додання акторів

5.2.7 Відкриття списку акторів

Користувач натискає на кнопку «Actor list» і відкривається спливаюче вікно акторів. Його зміст:

• Опис вмісту стовпчиків

• Рядки з медіа по конкретному актору: ім’я, фотографія, кнопка для видалення актора.

• Кнопка «Ок» для закриття вікна.

Нижче, на рисунку 5.2.7 наведено приклад спливаючого вікна для списку акторів:

Рисунок 5.2.7 Вікно списку акторів

5.2.8 Відкриття іншого списку акторів

Користувач бачить перед собою список акторського складу для фільму «Harry Potter».

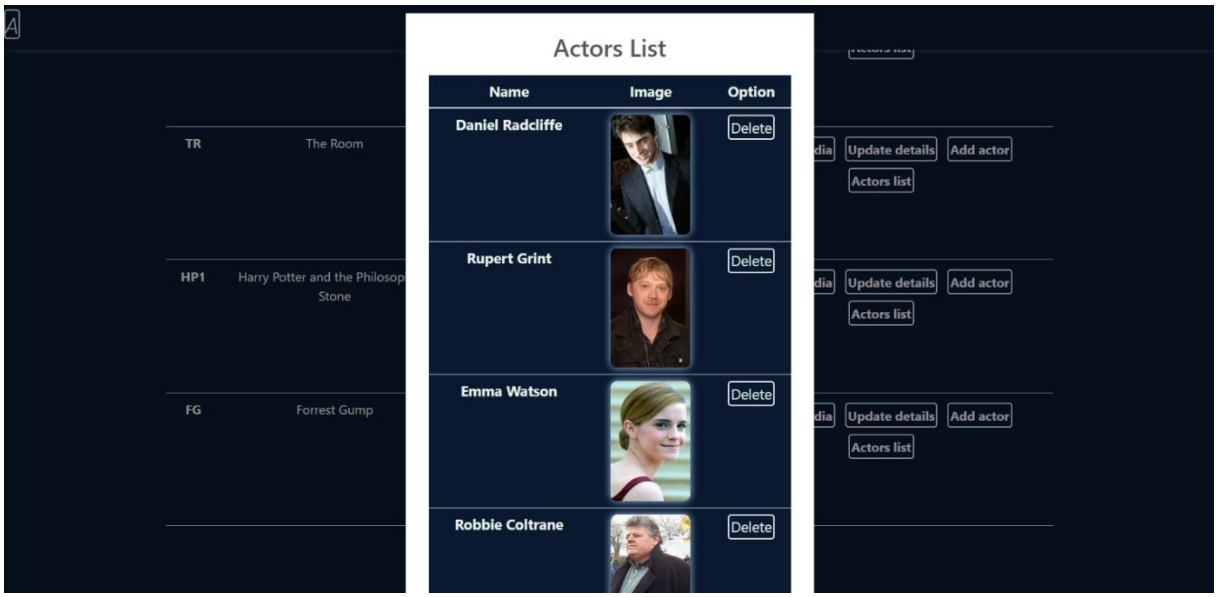

Рисунок 5.2.8 Вікно списку акторів іншого фільму

5.2.9 Видалення актора зі списку

Тепер користувач видаляє другого актора зі списку.

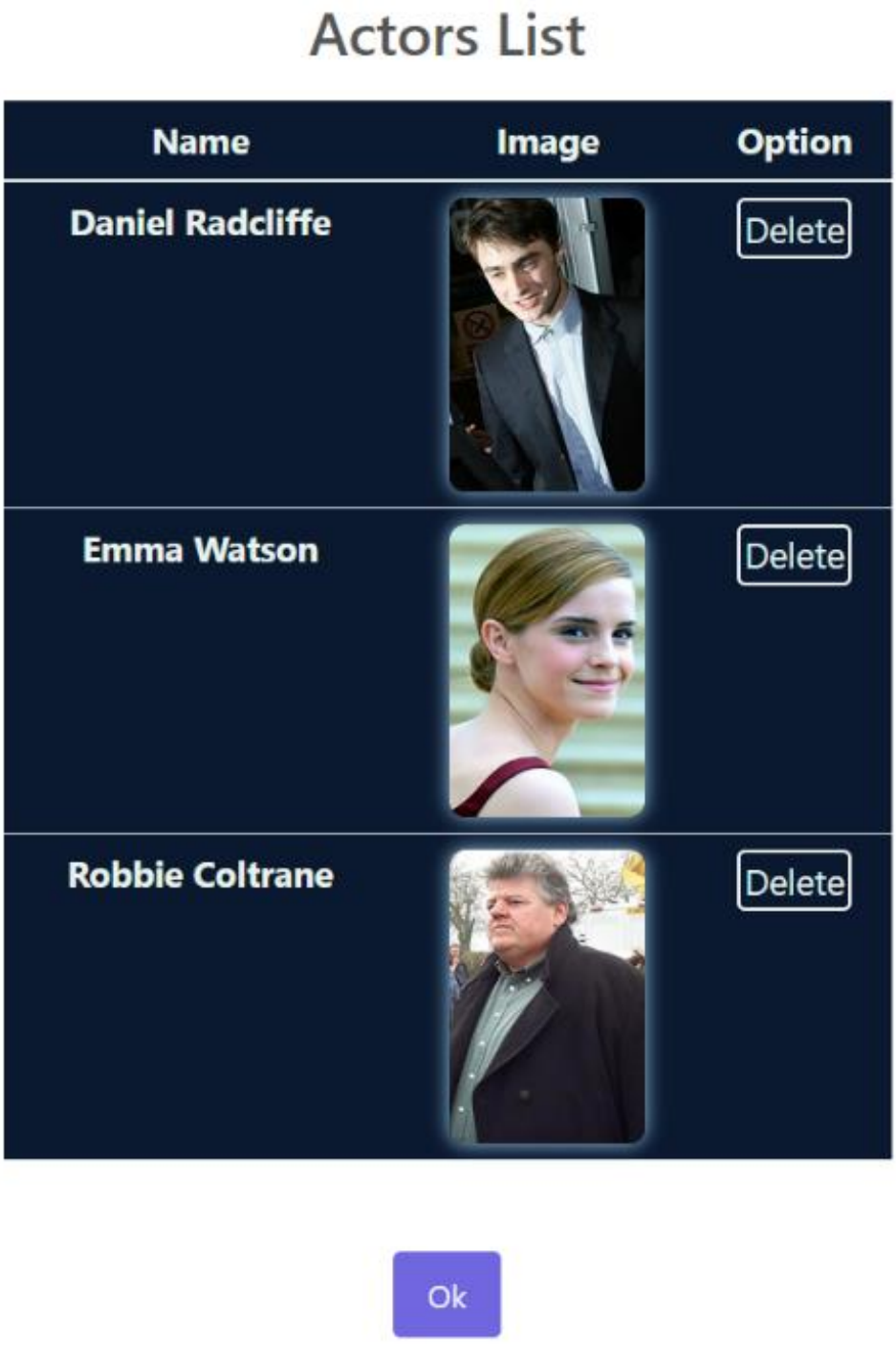

Рисунок 5.2.9 Вікно списку акторів іншого фільму після видалення актору

5.2.10 Виявлення неточності інформації

Користувач помітив неточність в інформації окремого фільму:

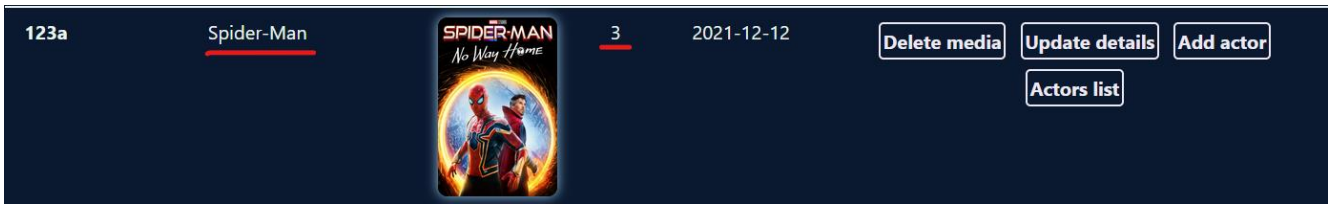

Рисунок 5.2.10 Дані фільму до оновлення

5.2.11 Поновлення даних

Користувач натискає кнопку “Update details” відкриває сторінку поновлення даних про фільм, редагує необхідні поля:

Рисунок 5.2.11 Сторінка оновлення даних по фільму

5.2.12 Видимість результатів

Користувач бачить результат змін:

Рисунок 5.2.12 Дані фільму після оновлення

5.2.13 Додавання фільму

Користувач натискає на кнопку “Add media” на головній сторінці веб-порталу:

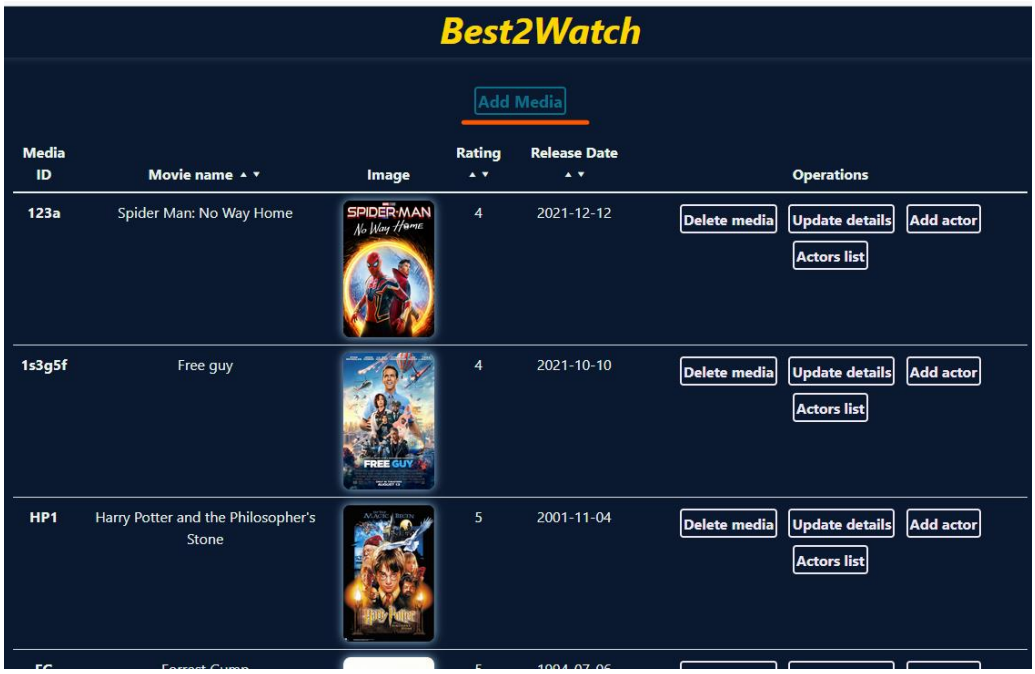

Рисунок 5.2.13 Кнопка “Add media” на головній сторінці веб-порталу

5.2.14 Сторінка додавання фільму:

Користувач попадає на сторінку додавання фільму:

Рисунок 5.2.14 Сторінка додавання нового фільму

5.2.15 Введення некоректних даних

Користувач намагається ввести некоректні дані у поля:

• Символи не по формату у поле “ID” (рисунок 5.2.15.1)

• Набір символів, що не є посиланням url у поле “Picture(url)” (рисунок 5.2.15.2)

• Неповна дата в календарі для поля “Release date” (рисунок 5.2.15.3)

• Пусте поле “ how many seasons?”, що набуває обов’язковості до введення після обраного пункту “Series” (рисунок 5.2.15.4)

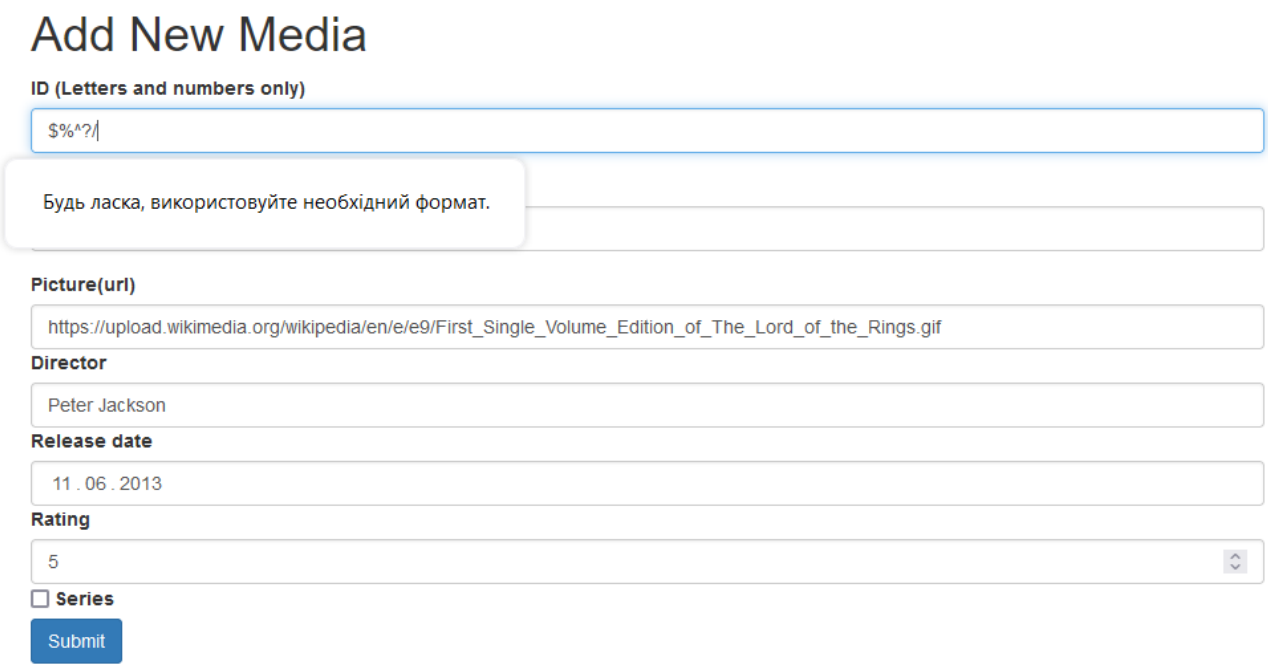

Рисунок 5.2.15.1 Валідація символів у полі “ID” при додаванні фільму

Рис 5.2.15.2 Перевірка коректності введення посилання у полі “Picture(url)” при додаванні фільму

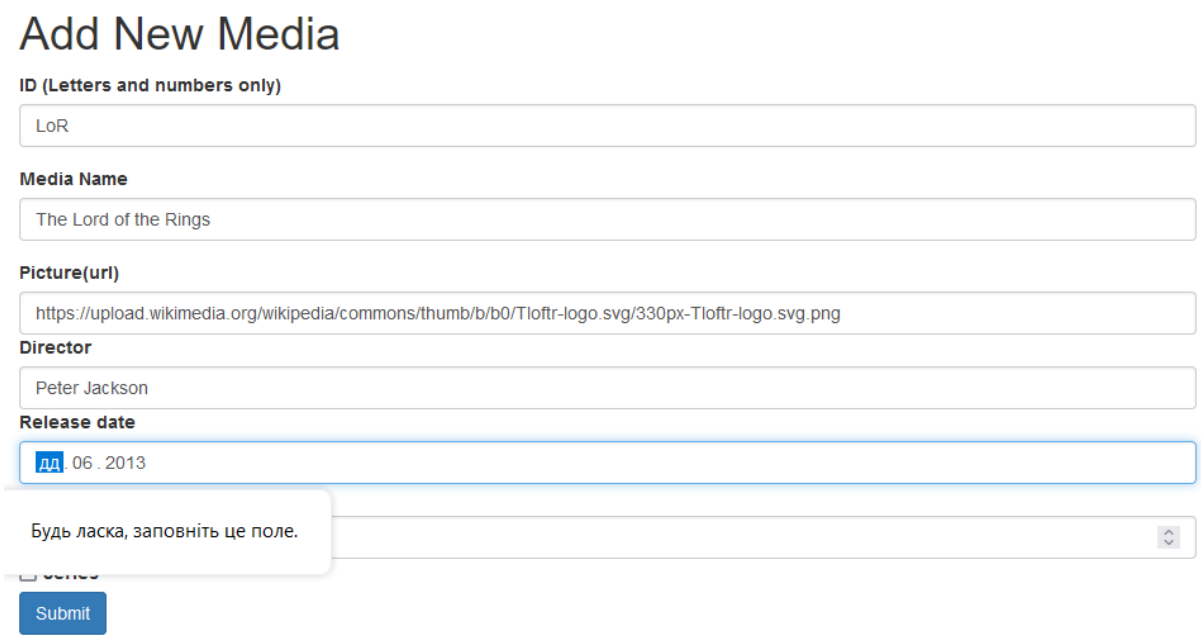

Рисунок 5.2.15.3 Валідація повноти дати у полі “Release date” при додаванні фільму

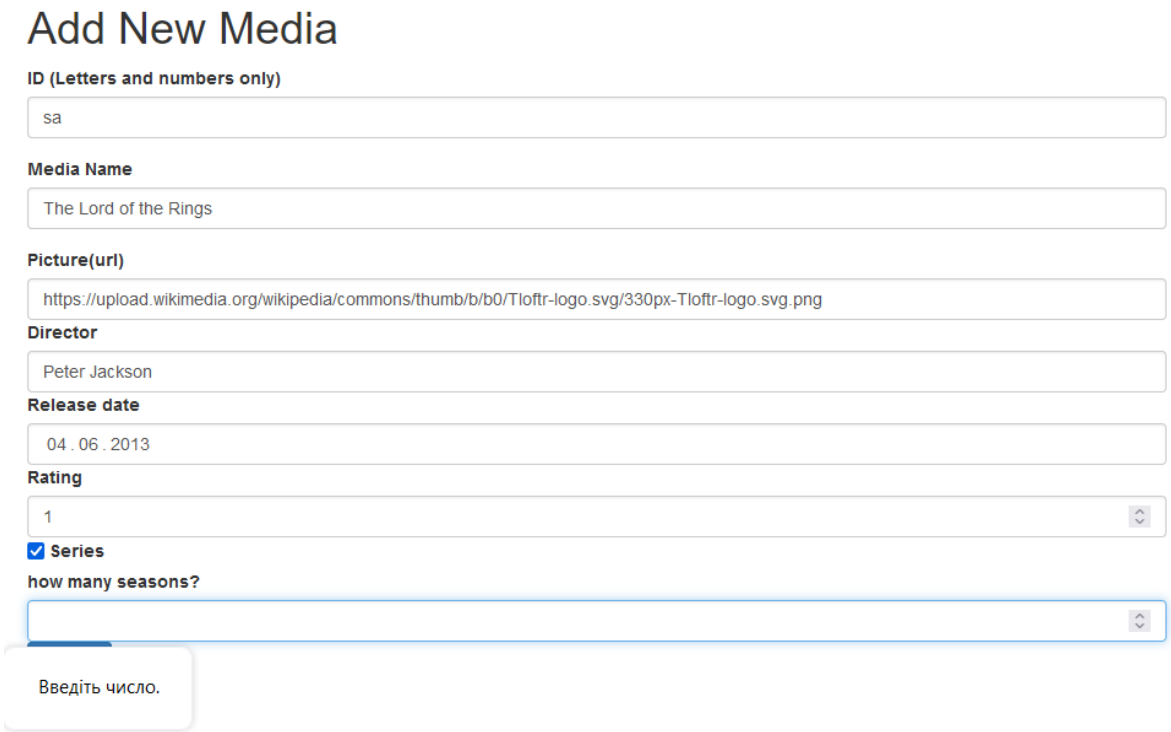

Рисунок 5.2.15.4 Перевірка числа сезонів у полі “ how many seasons?” у випадку позитивного вибору пункту “Series” при додаванні фільму

5.2.16 Додавання сезонів до фільму

У випадку, якби користувач додавав серіал і обрав у полі “ how many seasons?”, наприклад, число 5, сторінка би автоматично підтягнула 5 полів для кожного сезона відповідно:

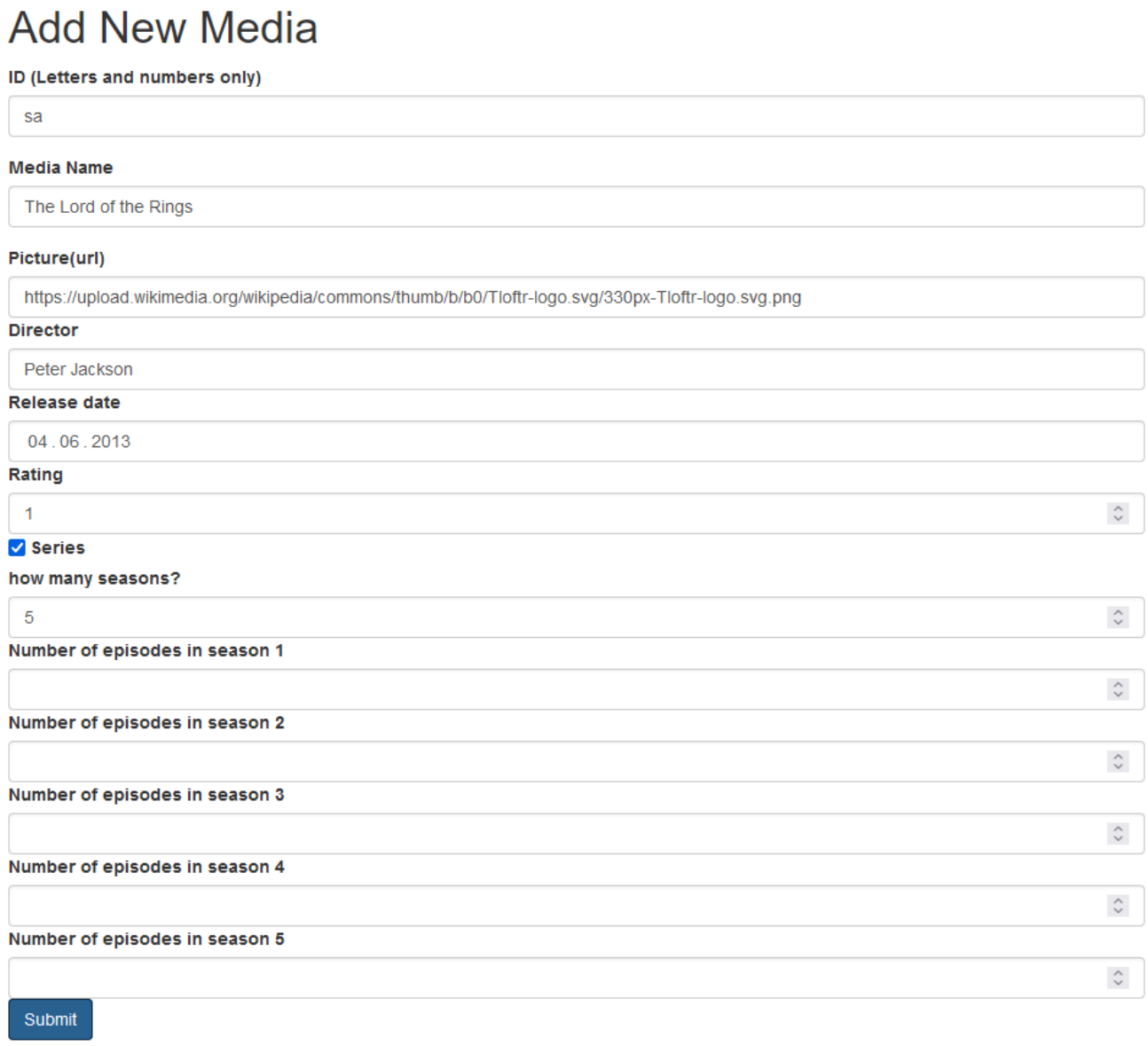

Рисунок 5.2.16 Автоматичне відображення полів для введення назви сезонів залежно від обраного числа у полі “ how many seasons?” при додаванні фільму

5.2.17 Погодження форми додавання фільму

Користувач додає новий фільм у каталог, натискаючи клавішу “Submit”. Всі обов’язкові поля заповнені, всі дані введено вірно.

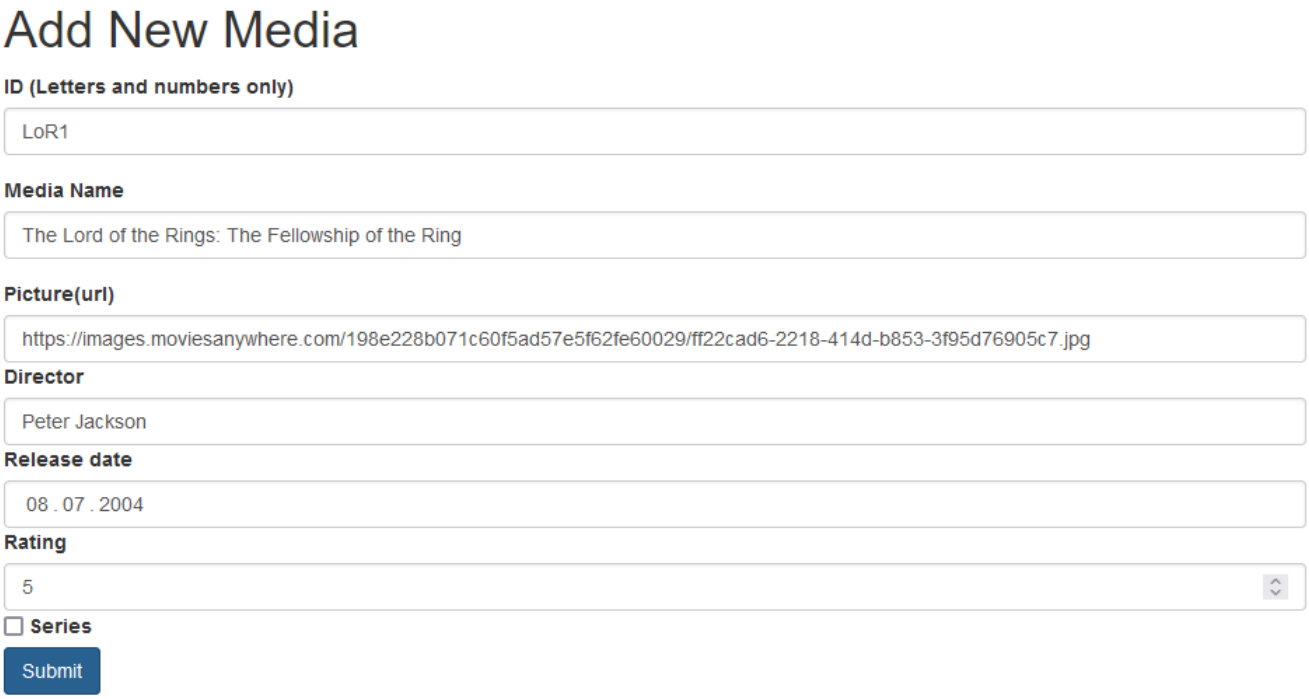

Рисунок 5.2.17 Додавання фільму у теку

5.2.18 Результат успішного додавання об’єкту фільм

Новий фільм відображається на головній сторінці порталу. Нижче, на рисунку 5.2.18 наведено скріншот головної сторінки сайту (порівняно з попередніми рисунками, на порталі з’явився новий об’єкт – фільм “ The Lord of the Rings: The Fellowship of the Ring”):

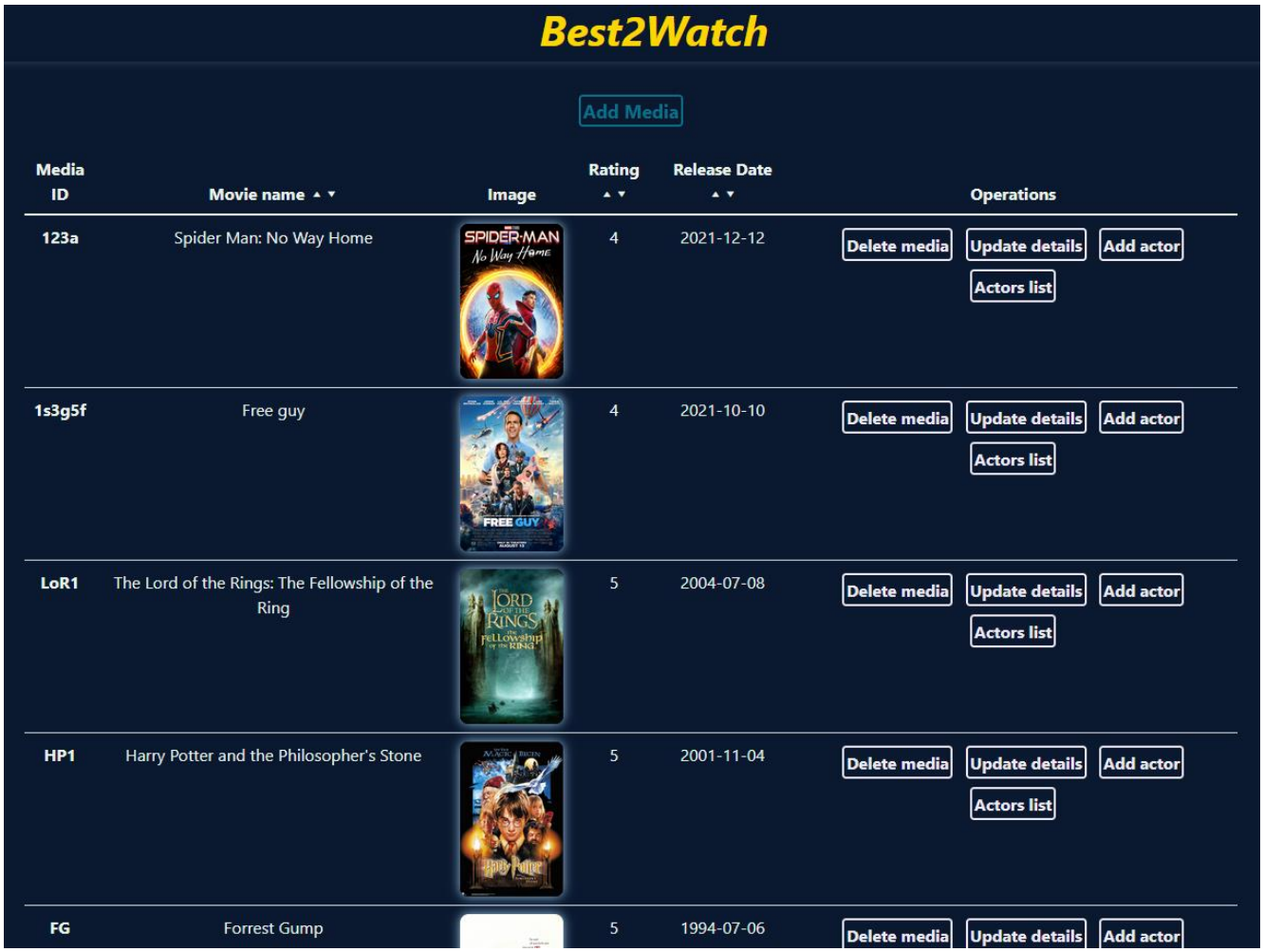

Рисунок 5.2.18 Відображення нового фільму на головній сторінці

5.2.19 Візуалізація бази даних

Користувач може побачити базу даних у первинному вигляді, додавши до шляху сайту /Media. Така можливість зумовлена впровадженням простої логіки за використанням наявних атрибутів node.js. На рисунку 5.2.19 наведено скріншот користувацького відображення модуля бази даних у зручному для перегляду форматі, із збереженням ієрархії об’єктів:

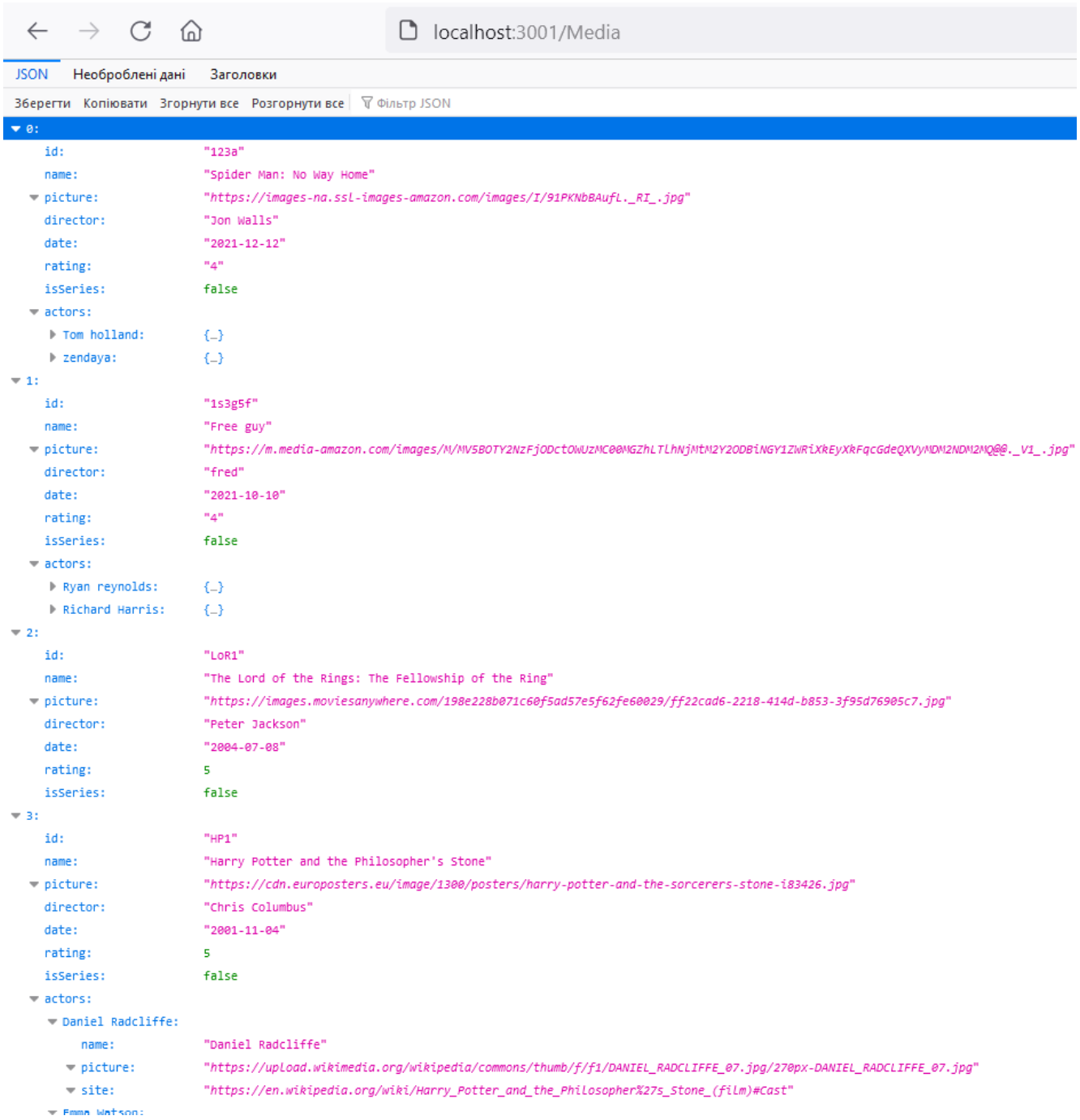

Рис 5.2.19 Відображення веб-варіанту бази даних

Таким чином, можна зробити висновок, що даний програмний продукт повністю відповідає клієнт-серверній архітектурі і є зручним для використання користувачем з метою ведення каталогу фільмів.

ВИСНОВКИ

В ході роботи було проведено аналіз дизайну веб-сторінок, побудовано шаблони, переведення їх в реальні сторінки. Під час створення порталу було проаналізовано та вивчено методику розробки та дизайну сайтів. Були покращені навички розробки сайтів та інтерфейсів.

Для побудови системи було використано середовище розробки WebStorm, що дуже спростило процес роботи з продуктом та створення клієнт-серверної архітектури, для написання сайту я використав мову гіпертекстової розмітки HTML, каскадні таблиці стилів CSS, мову програмування JavaScript та Фреймворк Bootstrap, що також прискорило процес написання сайту, .

Розроблену платформу можна використовувати у власних цілях, задля ведення теки фільмів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. С. Хокінгс Адміністрація Web-сервера Apache та посібник з електронної комерції: навч. посіб. Київ, 2001, 330 с. http://muff.kiev.ua/files/books/Administririvanie.web-servera.Apache.pdf (дата звернення 3.05.2022)
2. Кириченко О.В., Дубовик А.Є. Довідник HTML. Коротко, швидко, під рукою: навч. посіб. Київ, 2015, 288 с. (дата звернення 19.03.2022)
3. Д. Фленаган JavaScript: Повне керівництво: навч. посіб. Київ, 2018, 722 с. http://kharchuk.ru/JavaScript.pdf (дата звернення 15.01.2022)
4. Р. С. Мартін Чиста архітектура. Мистецтво розробки програмного забезпечення: навч. посіб. Київ, 2019, 368 с. (дата звернення 03.04.2022)
5. DNS: веб-сайт. URL: https://ru.wikipedia.org/wiki/DNS (дата звернення: 07.04.2022).
6. FTP: веб-сайт. URL: https://ru.wikipedia.org/wiki/FTP (дата звернення: 14.04.2022).
7. IMDb: веб-сайт. URL: https://en.wikipedia.org/wiki/IMDb (дата звернення: 01.05.2022).
8. IMDb, top movies: веб-сайт. URL: https://www.imdb.com/chart/top/?ref_=nv_mv_250 (дата звернення: 27.04.2022).
9. IMDb, sorted: веб-сайт. URL: https://www.imdb.com/chart/top/?sort=us,desc&mode=simple&page=1 (дата звернення: 24.04.2022).

10. IMDb, actor list: веб-сайт. URL: https://www.imdb.com/title/tt1745960/fullcredits/?ref_=tt_ql_cl (дата звернення: 02.05.2022).
11. AJAX: веб-сайт. URL: https://uk.wikipedia.org/wiki/AJAX (дата звернення: 03.05.2022).
12. AJAX: картинка. URL: https://uk.wikipedia.org/wiki/AJAX#/media/%D0%A4%D0%B0%D0%B9%D0%BB:Ajax-vergleich-en.svg (дата звернення: 08.05.2022).
13. WebStorm: веб-сайт. URL: https://ru.wikipedia.org/wiki/WebStorm (дата звернення: 01.04.2022).
14. CSS: веб-сайт. URL: https://uk.wikipedia.org/wiki/CSS (дата звернення: 06.05.2022).
15. CSS: веб-сайт. URL: https://en.wikipedia.org/wiki/CSS (дата звернення: 21.04.2022).
16. npm: веб-сайт. URL: https://en.wikipedia.org/wiki/Npm_(software) (дата звернення: 22.04.2022).
17. npm: веб-сайт. URL: https://ru.wikipedia.org/wiki/Npm_(software) (дата звернення: 22.04.2022).
18. Bootstrap: веб-сайт. URL: https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework) (дата звернення: 25.04.2022).

ДОДАТОК А

Створення клієнт-серверної системи обробки та аналізу вхідних даних користувача для оптимізації роботи веб-порталу

Лістинг програмного модулю

УКР.НТУУ“КПІ ім. Ігоря Сікорського” _ТР82_

Аркушів — 15

Київ 2022

Модуль logic.js

```
$(document).ready(function () {
  getMedia();
});

/**
 * Функція викликає Ajax, щоб отримати всю медіа з сервера.
 * З нього створюється таблиця для відображення
 */
function getMedia() {
  $.ajax({
    /** Посилання хосту **/
    url: "http://localhost:3001/media",
    type: "GET",
    /** У випадку успішної обробки **/
    success: function (response) {
      let mediaRes = "";
      /** Цикл для виведення кожного фільму **/
      for (let i = 0; i < response.length; i++) {
        /** Результатом одного проходження стане ряд із вмістом вичитаної із бази інформації + кнопок опцій (видалити, поновити...) **/
        mediaRes += `
              <tr>
              <th scope="row">${response[i].id}</th>
              <td>${response[i].name}</td>
              <td> <img src="${response[i].picture}" alt="media pic" class="icon"></td>
              <td>${response[i].rating}</td>
              <td>${response[i].date}</td>

              <td><button class="operationsDel" id="delete_media${i}">Delete media</button>
              <button class="operationsUp" id="update_media${i}">Update details</button>
              <button class="operationsAdd" id="add_actor${i}">Add actor</button>
              <button class="operationsAct" id="watch_actors${i}">Actors list</button></td>
```

```
            </tr>`;
      }
      /** Виведення у вигляді таблиці **/
      $("#table_id > tbody").html(mediaRes);
      /** Виклик функцій **/
      setTable();
      deleteMedia();
      updateDetails();
      addActor();
      watchList();
    },
    error: function (error) {
      console.log(error.responseText);
    },
  });
}

/** Ця функція виконує правила таблиці **/
function setTable() {
  $("#table_id").DataTable({
    /** Виставлення дефолтних параметрів **/
    searching: false,
    paging: false,
    info: false,
    order: [[4, "desc"]],
    columnDefs: [
      { orderable: false, targets: 0 },
      { orderable: false, targets: 2 },
      { orderable: false, targets: 5 },
    ],
  });
}

/** Функція викликає Ajax, щоб видалити окрему медіа з сервера. **/
function deleteMedia() {
  $(".operationsDel").click(function () {
    let id = $(this).parent().parent().find("th").text();
```

```
    $.ajax({
      url: `http://localhost:3001/media/${id}`,
      /** Виклик стандартгого атрибуту видалення для ajax **/
      type: "DELETE",
      /** Перевірка на правильність виконання/помилку **/
      success: function (response) {
        console.log(response);
        location.href = "/list";
      },
      error: function (error) {
        console.log(error.responseText);
      },
    });
  });
}

/** Функція зберігає id локально, щоби відправити на поновлену форму **/
function updateDetails() {
  $(".operationsUp").click(function () {
    let id = $(this).parent().parent().find("th").text();
    localStorage.setItem("mediaId", id);
    location.href = "/updateForm";
  });
}

/**
 * Функція відображає форму у вигляді вспливаючого вікна, для додавання нового актора
 * Після підтвердження (submit) функція викличе Ajax для додавання актора в об'єкт фільму
 */
function addActor() {
  $(".operationsAdd").click(function () {
    let id = $(this).parent().parent().find("th").text();
    Swal.fire({
      /** Стилі **/
```

```
    title:
      "<h4 style='background-color:white;color:black'> Add Actor Form</h4>",
    html: `<input type="text" id="name" class="swal2-input" style="background-
color:white" placeholder="Name">
      <input type="url" id="picture" class="swal2-input" style="background-
color:white" placeholder="Image(url)">
      <input type="url" id="site" class="swal2-input" style="background-color:white"
placeholder="fan site">`,
    confirmButtonText: "Add actor",
    focusConfirm: false,
    customClass: {
     title: "title-class",
     htmlContainer: "htmlContainer-class",
     actions: "actions-class",
    },
    preConfirm: () => {
     /** Вміст форми **/
     const name = Swal.getPopup().querySelector("#name").value;
     const picture = Swal.getPopup().querySelector("#picture").value;
     const site = Swal.getPopup().querySelector("#site").value;
     if (!name || !picture || !site) {
      Swal.showValidationMessage(`Please enter all details`);
     }
     /** Валідатор символів **/
     else if(!/^[a-zA-Z, ,-]*$/.test(name)){
      Swal.showValidationMessage(`Please enter valid name`);
     }
     /** Валідатор url **/
      else {
       try {
        let myURL = new URL(picture);
        let myURL2 = new URL(site);
       } catch (e) {
        Swal.showValidationMessage(`Invalid url`);
       }
     }
```

```
        return { name: name, picture: picture, site: site };
      },
    }).then((result) => {
      if (!result.isDismissed) {
        res = `{
              "name": "${result.value.name}",
              "picture" : "${result.value.picture}",
              "site": "${result.value.site}"
           }`;
        console.log(res);
        $.ajax({
          /** Занесення актора у відповідний об'єкт фільму по id **/
          url: `/media/${id}/actors`,
          contentType: "application/json",
          type: "PUT",
          datatype: "json",
          data: res,
          encode: true,
          success: function (response) {
            location.href = "/list";
          },
          error: function (error) {
            console.log(error.responseText);
          },
        });
      }
    });
  });
}

/**
 * Функція викличе Ajax для вичитки всіх акторів конкретного об'єкту фільму
 * Далі відобразить список акторів у вигляді спливаючого вікна
 */
function watchList() {
  $(".operationsAct").click(function () {
    let id = $(this).parent().parent().find("th").text();
```

```
    $.ajax({
      url: `http://localhost:3001/media/${id}`,
      type: "GET",
      success: function (response) {
        /** Формування спливаючого вікна **/
        let actorsTable = `<table class='table' id='actorsTable'>
                    <thead>
                      <tr>
                      <th scope="col">Name</th>
                     <th scope="col">Image</th>
                     <th scope="col">Option</th>
                    </tr>
                    </thead>
                    <tbody>`;
        /** Вичитка з бд + реагування на натискання кнопки видалення актора (deleteAct) **/
        if (response.actors) {
          for (actor in response.actors) {
            actorsTable += `<tr>
            <th scope="row">${response.actors[actor].name}</th>
            <td> <img src="${response.actors[actor].picture}" alt="actor pic" class="icon"></td>
            <td> <button class="deleteAct" style="border:solid 2px rgb(255, 255, 255); border-radius: 4px; padding:2px" id="delete_actor${response.actors[actor].name}">Delete</button></td>
          </tr>`;
          }
        }
        actorsTable += `</tbody>
        </table>`;
        Swal.fire({
          title: "Actors List",
          html: actorsTable,
          confirmButtonText: "Ok",
          focusConfirm: false,
          customClass: {
            title: "title-class",
```

```
          htmlContainer: "htmlContainer-class",
          actions: "actions-class",
        },
      });
      /** Виклик функції видалення актора (по id) **/
      deleteActor(id);
    },
    error: function (error) {
      console.log(error.responseText);
    },
  });
 });
}

/**
 * Функція викличе Ajax для видалення форми акторів конкретного об'єкту фільму
 * по id відповідного фільму - @param {} id
 */
function deleteActor(id) {
 $(".deleteAct").click(function () {
  let name = $(this).parent().parent().find("th").text();
  $.ajax({
    url: `http://localhost:3001/media/${id}/actors/${name}`,
    type: "DELETE",

    success: function (response) {
      console.log(response);
      location.href = "/list";
    },
    error: function (error) {
      console.log(error.responseText);
    },
  });
 });
}
```

Модуль server.js

```
/**
 * Цей файл є серверним. Підключає всі шляхи та ініціалізації
 */

const express = require('express');
const { validationResult } = require('express-validator');
bodyParser = require('body-parser'),
    path = require('path'),
    fs = require('fs'),
    cors = require('cors'),
    routers = require('./server/routes/routes.js');
/** Прописання порта для localhost **/
const port = 3001

/** Використання стандартної функції **/
const app = express()
app.use(cors());
/** Використання функцій бібліотеки express **/
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
/** Настроювання url для хосту кожної зі сторінок .html **/
app.use('/list', express.static(path.join(__dirname, 'client/html/list.html')));
app.use('/addForm', express.static(path.join(__dirname, 'client/html/form.html')));
app.use('/updateForm', express.static(path.join(__dirname,
'client/html/updateDetails.html')));
app.use('/css', express.static(path.join(__dirname, 'client/css')));
app.use('/js', express.static(path.join(__dirname, 'client/js')));


app.use('/', routers);


/** Вивід у термінал при успішному запуску сервера **/
app.listen(port, () => {
```

```
  console.log(`Hosted. Main page http://localhost:${port}/list`)
})
```

Модуль form.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/css/bootstrap.min.css"
integrity="sha384-
BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4
u" crossorigin="anonymous">
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

    <script src="//cdn.jsdelivr.net/npm/sweetalert2@11"></script>
    <script src = "../js/addlogic.js"></script>

    <title>New Media</title>
</head>
<body>
    <div class="col-sm-6 col-sm-offset-3">
    <h1 class="display-1">Add New Media</h1>
    <form name='add_form' id="target">
        <div class="form-group">
          <label for="IdInput">ID (Letters and numbers only)</label><br>
          <input type="text" class="form-control media_id " id="IdInput" required
pattern="[a-zA-Z0-9]*">
        </div>
        <div class="form-group">
            <label for="nameInput">Media Name</label><br>
            <input type="text" class="form-control mname" id="nameInput" required
pattern="[a-zA-Z0-9, '-:]*">
```

```
    </div>
    <div class="form-check">
      <label for="picInput">Picture(url)</label><br>
      <input type="url" class="form-control pic" id="picInput" required>
    </div>
    <div class="form-check">
      <label for="directorInput">Director</label><br>
      <input type="text" class="form-control dname" id="directorInput" required
pattern="[a-zA-Z, ,-]*">
    </div>
    <div class="form-check">
      <label for="rdateInput">Release date</label><br>
      <input type="date" class="form-control rdate" id="rdateInput" required>
    </div>
    <div class="form-check">
      <label for="ratingInput">Rating</label><br>
      <input type="number" step=1 class="form-control rating" id="ratingInput"
min=1 max="5" required>
    </div>
    <div class="form-check form-check-inline">
      <input class="form-check-input" type="checkbox" name="inlineRadioOptions"
id="seriesCheck" value="option1">
      <label class="form-check-label" for="inlineRadio1">Series</label>
    </div>

    <div id="res" style="display:none">
      <div class="form-check">
        <label for="seasonInput">how many seasons?</label><br>
        <input type="number" step=1 class="form-control rating" id="seasonInput" >
      </div>
      <div id = "episodes">

      </div>
    </div>

    <button type="submit" class="btn btn-primary" id="button">Submit</button>
```

```
      </form>
    </div>
</body>
</html>
```

Модуль list.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Ex3 - Best2Watch</title>
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
      rel="stylesheet"
      integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
      crossorigin="anonymous"
    />
    <link rel="stylesheet" href="../css/style.css" />


    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    <script src="//cdn.jsdelivr.net/npm/sweetalert2@11"></script>
    <script
      type="text/javascript"
      charset="utf8"
      src="https://cdn.datatables.net/1.11.5/js/jquery.dataTables.js"
    ></script>

    <script src="../js/logic.js"></script>
  </head>
  <body>
    <nav>
```

```
    <div class="logo">
      <h3>A</h3>
    </div>
    <h1>Best2Watch</h1>
  </nav>

  <main>
    <div class="container">
      <div class="container-fluid">
        <a href="http://localhost:3001/addForm" class="create" id="create1"
          >Add Media</a
        >

        <table class="table" id="table_id">
          <thead>
            <tr>
              <th scope="col">Media ID</th>
              <th scope="col">Movie name &#x25b4;&#x25be;</th>
              <th scope="col">Image</th>
              <th scope="col">Rating &#x25b4;&#x25be;</th>
              <th scope="col">Release Date &#x25b4;&#x25be;</th>
              <th scope="col">Operations</th>
            </tr>
          </thead>
          <tbody></tbody>
        </table>

        <a href="http://localhost:3001/addForm" class="create" id="create2"
          >Add Media</a
        >
      </div>
    </div>
  </main>
 </body>
</html>
```

Модуль updateDetails.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/css/bootstrap.min.css"
integrity="sha384-
BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4
u" crossorigin="anonymous">
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    <script src = "../js/updatelogic.js"></script>

    <title>Update</title>
</head>
<body>
    <div class="col-sm-6 col-sm-offset-3">
    <h1 class="display-1">Update Media</h1>
    <h2>Update the fileds you want</h2>
    <h3>(You don't have to update all of them)</h3>
    <form name='add_form' id="target">
        <div class="form-group">
            <label for="nameInput">Media Name</label><br>
            <input type="text" class="form-control mname" id="nameInput" pattern="[a-
zA-Z0-9, '-:]*">
        </div>
```

```
        <div class="form-check">
            <label for="picInput">Picture(url)</label><br>
            <input type="url" class="form-control pic" id="picInput">
        </div>
        <div class="form-check">
            <label for="directorInput">Director</label><br>
            <input type="text" class="form-control dname" id="directorInput" pattern="[a-zA-Z, ,-]*">
        </div>
        <div class="form-check">
            <label for="rdateInput">Release date</label><br>
            <input type="date" class="form-control rdate" id="rdateInput">
        </div>
        <div class="form-check">
            <label for="ratingInput">Rating</label><br>
            <input type="number" step=1 class="form-control rating" id="ratingInput" min=1 max="5">
        </div>
        <div class="form-check form-check-inline">
            <input class="form-check-input" type="checkbox" name="inlineRadioOptions" id="seriesCheck" value="option1">
            <label class="form-check-label" for="inlineRadio1">Series</label>
        </div>

        <div id="res" style="display:none">
            <div class="form-check">
                <label for="seasonInput">how many seasons?</label><br>
                <input type="number" step=1 class="form-control rating" id="seasonInput" >
            </div>
```

```
            <div id = "episodes">

            </div>
        </div>

        <button type="submit" class="btn btn-primary" id="button">Submit</button>

      </form>
    </div>
</body>
</html>
```