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 кіно-додатку на основі мови програмування JavaScript з використанням технологій html, css

Виконав: студент IV курсу, групи СТ-41

спеціальності 126 Інформаційні системи та технології
(шифр і назва спеціальності)

(підпис) Бачинський А.В. (прізвище та ініціали)

Керівник (підпис) Приймак М.В. (прізвище та ініціали)

Нормоконтроль (підпис) Марценко С.В. (прізвище та ініціали)

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

Рецензент (підпис) Цуприк Г.Б. (прізвище та ініціали)

Тернопіль
2023

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)
«__» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 126 Інформаційні системи та технології
(шифр і назва спеціальності)

Студенту Бачинському Андрію Васильовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка кіно-додатку на основі мови програмування JavaScript з використанням технологій html, css

Керівник роботи Приймак Микола Володимирович, доктор технічних наук, професор кафедри комп'ютерних наук
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 7 » лютого 2023 року № 4/7-13Х

2. Термін подання студентом завершеної роботи 23 червня 2023р.

3. Вихідні дані до роботи

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. Розділ 1. Створення розмітки додатку та стилізація коду. 1.1 Створення проекту у редакторі коду Visual Studio Code. 1.2 Написання розмітки за технологією HTML. 1.2.1 Технологія HTML. 1.3 Стилізація додатку за допомогою технології CSS. 1.4 Основі принципи технології CSS. 1.5 Підключення фреймворку Bootstrap до основної стилізації коду. 1.6 Сучасні веб-технології, та архітектура сайтів. 1.7 Висновок до першого розділу. Розділ 2. Побудова додатку на мові програмування JavaScript. 2.1 Мова програмування JavaScript. 2.2 Підключення JavaScript до розмітки. 2.3 Інтерактивність та функціонування додатку через JavaScript. 2.3.1 API база даних. 2.4 Висновок до другого розділу. Розділ 3. Безпека життєдіяльності, основи охорони праці. 3.1 Надзвичайні ситуації викликані пожежами, вибухами, техногенними та природними причинами. 3.2 Заходи щодо захисту установки від короткого замикання. 3.3 Висновок до третього розділу.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1.Титульна сторінка. 2. Мета роботи. 3. Актуальність обраних технологій. 4. Етап розробки інтерфейсу. 5. Етап розробки функціоналу. 6. Взаємодія з API. 7. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці	Окіпний І.Б., к.т.н., доцент		

7. Дата видачі завдання 23 січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	23.01.2023	*Виконано*
2.	Підбір джерел про веб-технології html, css	24.01.2023-26.01.2023	*Виконано*
3.	Опрацювання джерел по темі кваліфікаційної роботи	27.01.2023-31.01.2023	*Виконано*
4.	Виконання дослідження щодо мови JavaScript для	01.02.2023-07.02.2023	*Виконано*
	розробки кіно-додатку		
5.	Оформлення розділу «Створення розмітки додатку та	08.02.2023-09.02.2023	*Виконано*
	стилізація коду»		
6.	Оформлення розділу «Побудова додатку на мові	10.02.2023-12.02.2023	*Виконано*
	програмування JavaScript»		
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	05.06.2023-06.06.2023	*Виконано*
8.	Виконання завдання до підрозділу «Основи хорони	07.06.2023-08.06.2023	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	09.06.2023-11.06.2023	*Виконано*
10.	Нормоконтроль	12.06.2023-13.06.2023	*Виконано*
11.	Перевірка на плагіат	14.06.2023	*Виконано*
12.	Попередній захист кваліфікаційної роботи	15.06.2023	*Виконано*
13.	Захист кваліфікаційної роботи	23.06.2023	

Студент _____ Бачинський А.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Приймак М.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка кіно-додатку на основі мови програмування JavaScript з використанням технологій html, css // Кваліфікаційна робота освітнього рівня «Бакалавр» // Бачинський Андрій Васильович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СТ-41 // Тернопіль, 2023 // С. –56, рис. – 37, слайди – 7, бібліог. – 19.

Ключові слова: html, css, javascript, api, bootstrap, visual studio code.

Кваліфікаційна робота присв'ячена розробці кіно-додатку для перегляду фільмів та серіалів на основі JavaScript.

В першому розділі кваліфікаційної роботи описано початкову побудову проекту за розміткою та стилізацією. Висвітлено сучасні та зручні технології для створення сайтів та додатків. Розглянуто основі принципи роботи в редакторі коду Visual Studio Code. Та проаналізовано чому краще створювати проекти у сучасних редакторах коду.

В другому розділі кваліфікаційної роботи досліджено основні принципи роботи мови програмування JavaScript, та підключення технології до проекту. Подано та реалізовано функціонал через JavaScript, для коректної роботи додатку.

В третьому розділі кваліфікаційної роботи описано надзвичайні ситуації викликані пожежами, вибухами, техногенними та природними причинами, також було проаналізовано які є заходи щодо захисту від короткого замикання.

ANNOTATION

Development of a movie application based on the JavaScript programming language with the use of html, css technologies // Qualification work of the educational level "Bachelor" // Bachynskyi Andriy Vasyliovych // Ternopil National Technical University named after Ivan Pulyu, faculty of computer and information systems and software engineering, department of computer sciences, group ST- 41 // Ternopil, 2023 // C.– 56, fig. – 37, draw. – 7, ref – 19

Keywords: html, css, javascript, api, bootstrap, visual studio code.

The qualification work is dedicated to the development of a movie application for watching movies and series based on JavaScript.

The first section of the qualification work describes the initial construction of the project in terms of layout and stylization. Modern and convenient technologies for creating websites and applications are highlighted. The basics of working in the Visual Studio Code code editor are considered. But it is analyzed why it is better to create projects in modern code editors.

In the second section of the qualification work, the basic principles of the JavaScript programming language and the connection of technology to the project are explored.Submitted and implemented functionality through JavaScript, for the correct operation of the application.

In the third section of the qualification work, emergency situations caused by fires, explosions, man-made and natural causes are described, as well as the measures to protect against short circuits were analyzed.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

JS- JavaScript.

HTML – HyperText Markup Language.

CSS – Cascading Style Sheets.

DOM – об’єктна модель документа.

JSON – JavaScript object notation.

HTTP – протокол передачі даних.

VsCode – редактор коду.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі, з кожним днем стає все більше користувачів у інтернеті. Використання таких веб-додатків вже стало невід'ємною частиною мільйонів людей по всьому світу. Тому ця тема буде актуальною впродовж багатьох років.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи є розробка кіно-додатку для перегляду фільмів та серіалів. Багато схожих веб-сайтів на сьогодні заблоковані або працюють некоректно. Тому мета даної роботи полягала в тому, щоб розробити зручний додаток та забезпечити комфортне використання для всіх користувачів.

Для розробки даного проекту були виконані наступні завдання:

– Вибір сучасного редактора коду.

– Написання розмітки та стилізація проекту.

– Створення інтерфейсу для користувачів.

– Підключення JavaScript та інших бібліотек до проекту.

– Розробка функцій та інтерактивності на JavaScript.

Практичне значення одержаних результатів.

Створення кіно-додатку у результаті виконання кваліфікаційної роботи. Даний проект написаний за допомогою мови програмування JavaScript, та в майбутньому буде оновлюватись та модернізовуватись.

РОЗДІЛ 1. СТВОРЕННЯ РОЗМІТКИ ДОДАТКУ ТА СТИЛІЗАЦІЯ КОДУ

1.1 Створення проекту у редакторі коду Visual Studio Code

Редактор коду Visual Studio від Microsoft - це сучасний і безкоштовний текстовий редактор. Він має багато переваг та плюсів у роботі з кодом , а саме автозаповнення, підсвічування синтаксису і легкість у налаштуванні.

На початку створення проектів, він виглядає так (Див. рисунок 1.1).

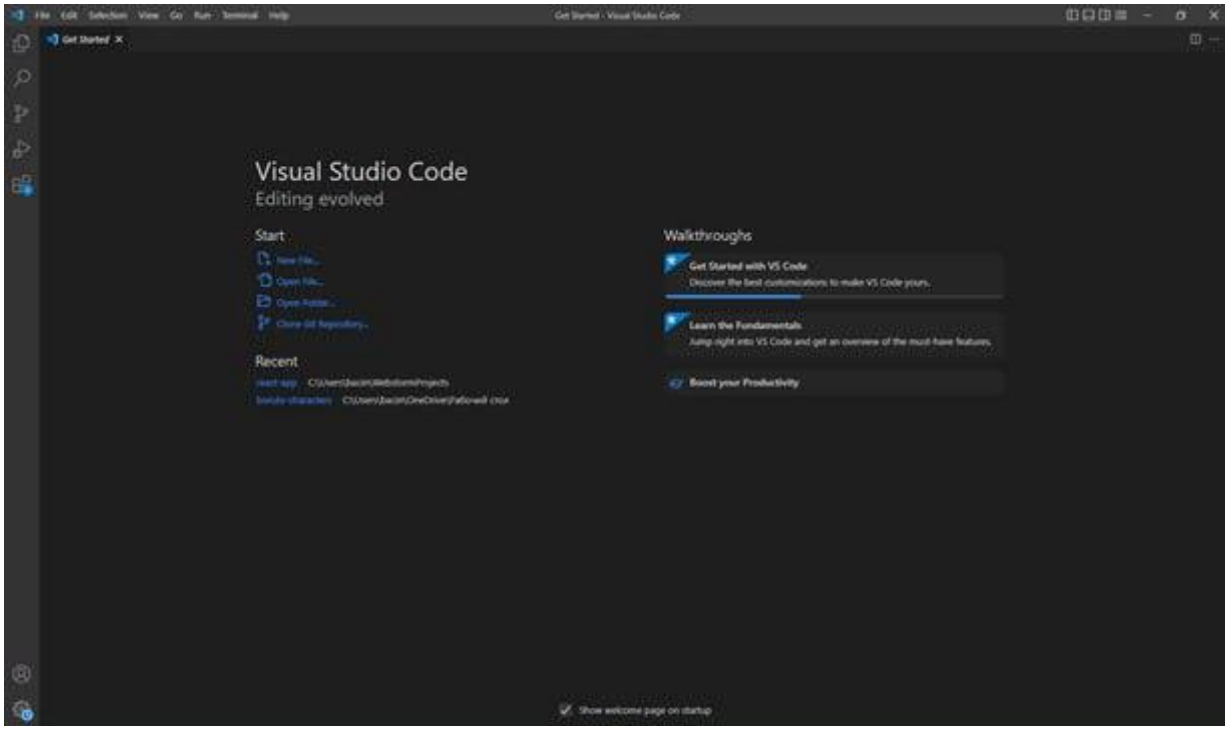

Рисунок 1.1 – Початкова сторінка Visual Studio Code

Команди системи контролю версій Git також вбудовані в цей редактор, щоб можна було відправляти запити, та швидко створювати гілки, репозиторії свого проекту у терміналі. Також є багато інших розширень, що дозволяють додавати нові мови, теми та інше (Див. рисунок 1.2).

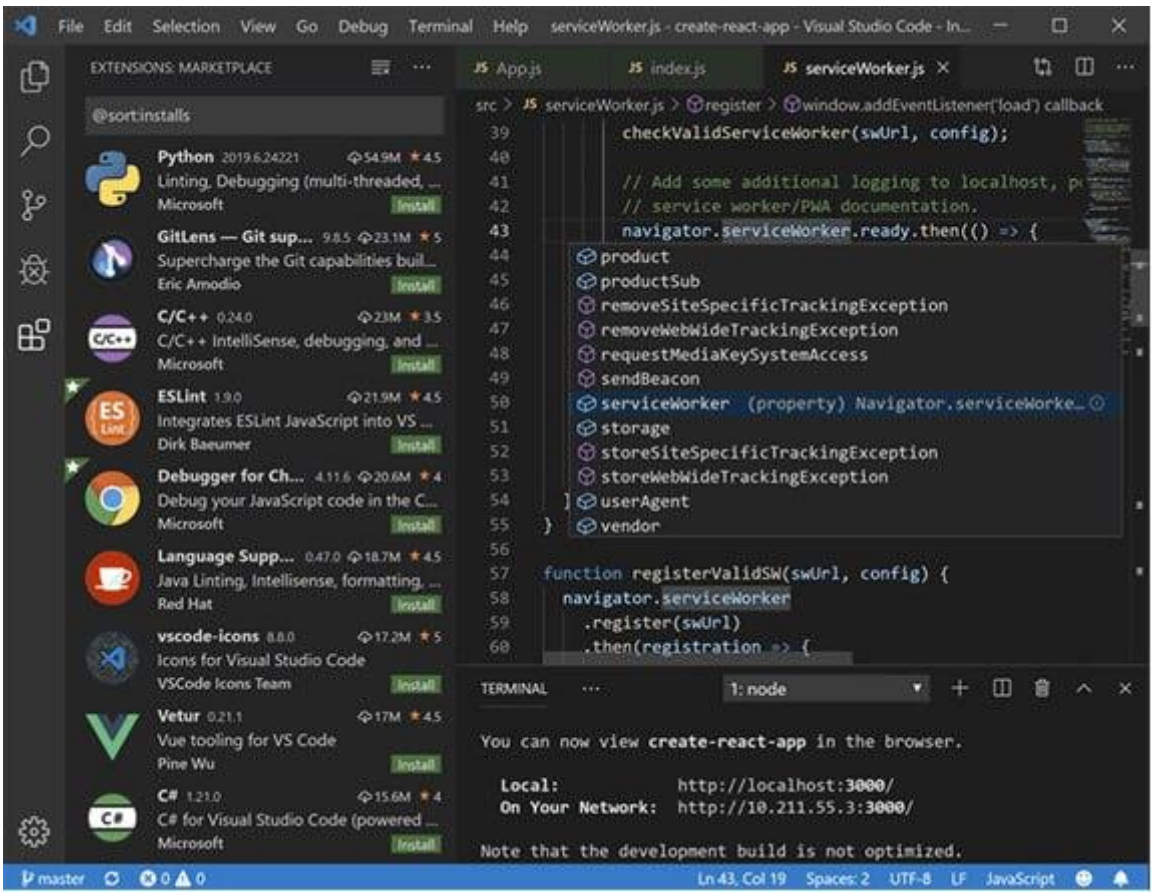

Рисунок 1.2 – Розширення для редактору

Із цим редактором коду я почав працювати ще 2 роки назад. Вибрав його після того коли працював з іншими програмами для написання коду, і зрозумів що для мене зручнішим та комфортнішим є саме цей редактор.

Швидкість написання коду, багато функцій, та автоматичне форматування, це великі переваги користуватись саме ним. Ось і свій проект кіно-додаток я вирішив створювати в цьому середовищі.

Один з великих переваг це плагін Live Server – через який можна відкрити свій проект в браузері по одному кліку, та слідкувати за результатом своєї роботи (Див. рисунок 1.3).

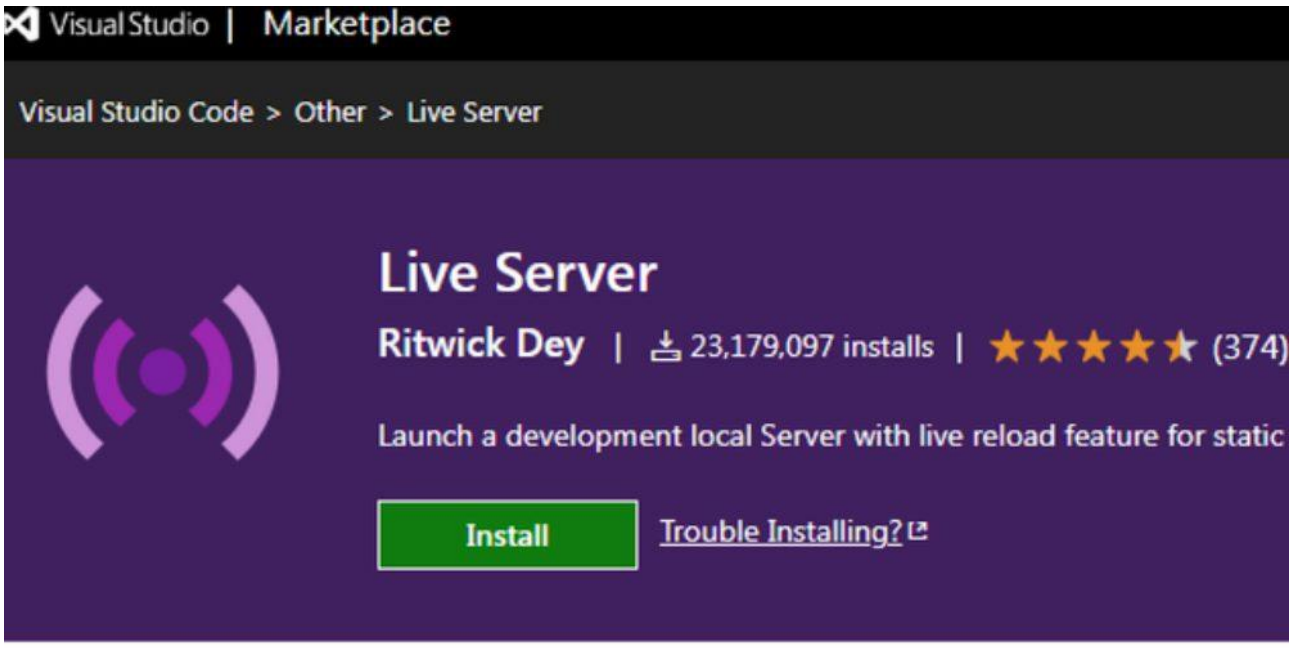

Рисунок 1.3 – Плагін Live Server

Він створює локальний і тимчасовий сервер лише для веб-сайту, що розробляється. За допомогою цього розширення можна візуалізувати як динамічні, так і статичні сторінки веб-сайту.

Текстовий редактор коду Visual Studio Code, дуже добре підходить для створення нових веб-проектів, та сайтів.

1.2 Написання розмітки за технологією HTML

Створення мого сайту як і всі сайти які існують сьогодні, починаються саме з цього, з розмітки в HTML. Структура html-документа на початку виглядає так (Див. рисунок 1.4).

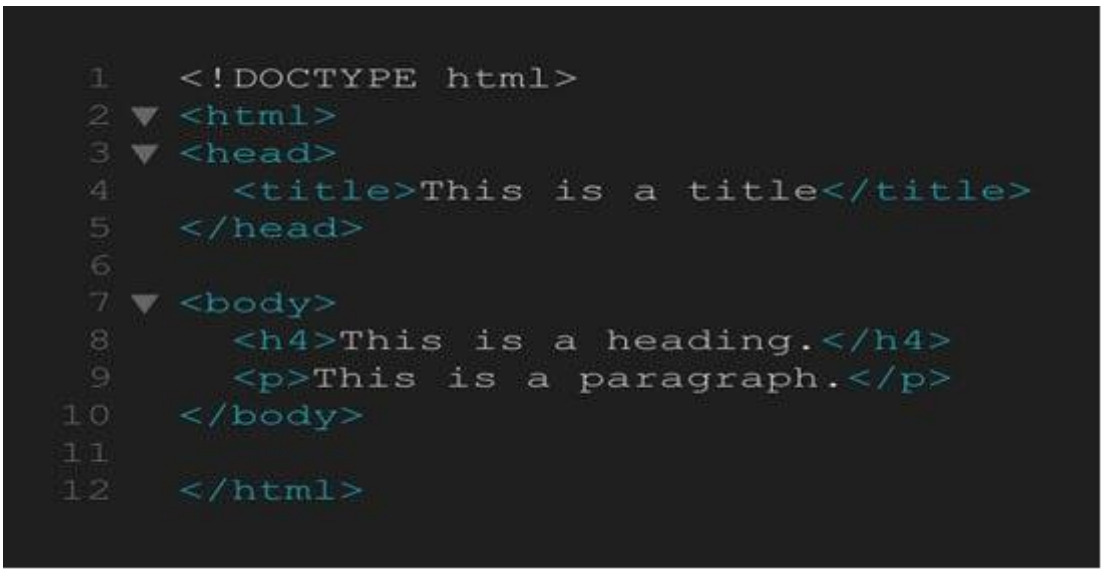

Рисунок 1.4 – Структура документа html

Іншими словами це можна назвати як “Скелет html-документа”. Кожен проект починається саме з нього, це початок роботи.

Я використовував сучасний html5 для побудови всіх елементів та блоків, які є на моєму сайті.

Для початку була створена пуста папка для проекту, а вже в ній перший файл під назвою «index.html», в якому і був написаний основний код html. В подальшій розробці проекту були створені та підключенні всі необхідні файли та програми (Див. рисунок 1.5).

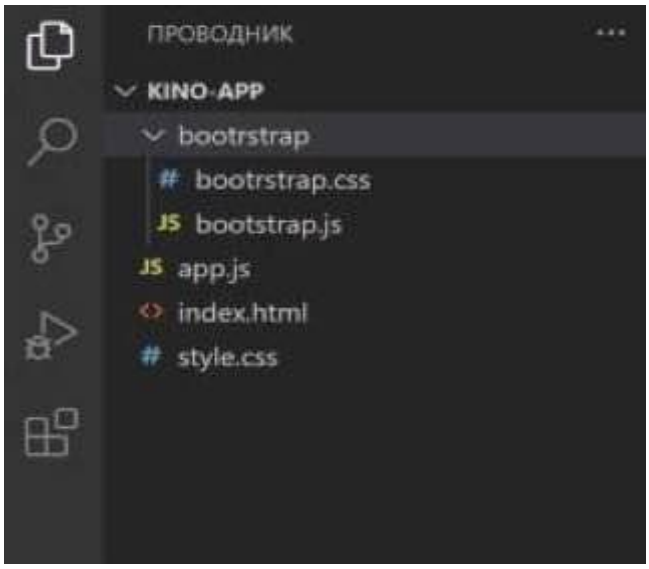

Рисунок 1.5 – Структура початкової папки проекту

Розширення файлу обов’язково повинно бути правильно записано, в іншому випадку, редактор просто не зрозуміє тип файлу, та не буде коректно працювати.

На самому початку документа прописано Doctype. Він говорить версію технології html. Як правило у кожному документі HTML має бути позначена версія документа яка використовується при кодуванні (Див. рисунок 1.6).

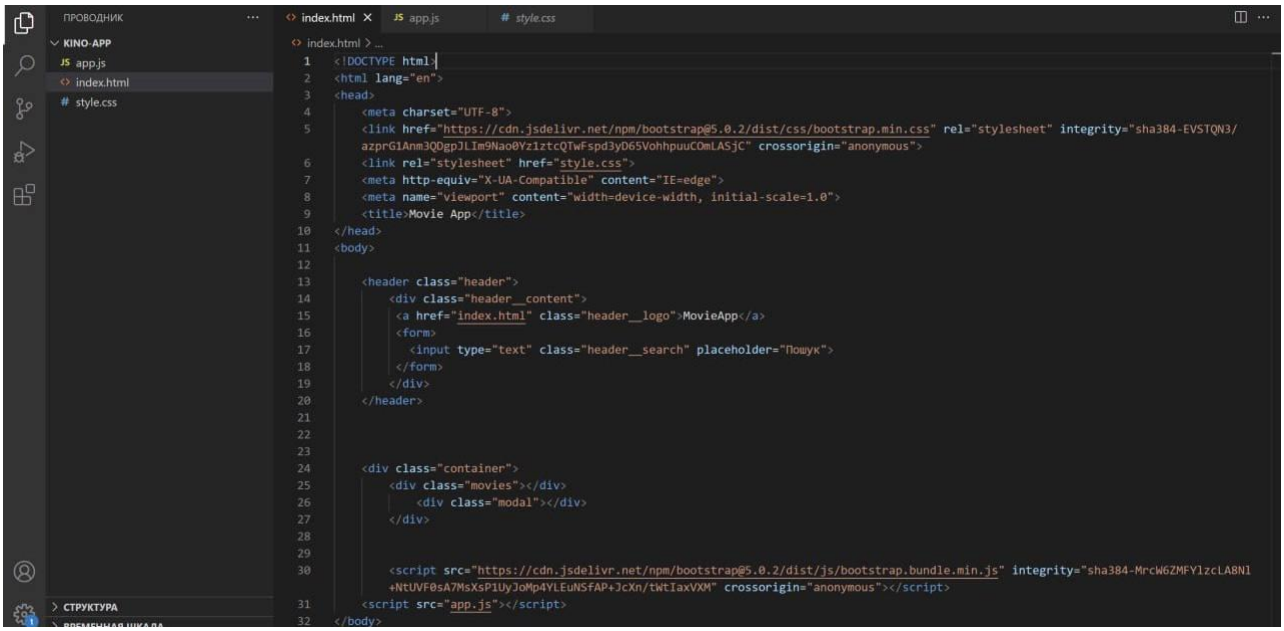

Рисунок 1.6 – Версія поточного елементу Doctype

Наступний рядок після декларації це тег “html”, це самий головний тег. Всі остальні основні теги знаходяться всередині тегу “html”.

Наступні теги після нього це “head” і “body”. Їх також можна вважати основними.

Головним тегом вважається тег “html”. Якщо видалити тег “html” то сторінка в браузері відкриється пустою.

В середині цих двох тегів входять інші теги, вони називаються “дочірні”. Це теги “meta” “link” “title”. Основа розмітка тримається у всіх цих тегах. Нічого не може виходити за рамки тега “body”, в іншому випадку код просто не буде працювати, це як обгортка всього написаного коду.

В тег “title” записується назва сайту. Те що буде написано в цьому тезі, те і буде показуватись користувачу у браузері. Тег “link” використовується в основному для підключення різних файлів, шрифтів, стилів до проекту.

1.2.1 Технологія HTML

HTML (Hyper Text Markup Language)– це мова розмітки, яка використовується при створенні веб-сайтів в інтернеті (Див. рисунок 1.7).

Рисунок 1.7 – Логотип HTML 5

Веб-браузери дуже добре розуміють html і можуть інтерпретувати у зрозумілому для користувача вигляді.

В 1991 році програміст Тім Бернерс з своїми помічниками, створили та опублікували в інтернеті першу мову розмітки HTML. Ця мова розмітки дуже швидко набрала популярності та почала використовуватись різними програмістами по всьому світу, для створення сайтів та додатків.

HTML показує браузеру, з яких саме елементів і в якому конкретному порядку будувати веб-сторінку, а також вказує де брати стилі елементів і скрипти інших бібліотек.

Він також дозволяє перетворити інформацію у різні способи. Наприклад, можна створювати таблиці, списки, параграфи, форматувати текст, додавати зображення, створювати відповідні заголовки, посилання та кнопки.

За допомогою цієї технології можна створювати практично все. Блоки, текст, відео матеріали, фотографії, та багато іншого.

Велика кількість тегів яка добавлена в html дозволяє створювати та добавляти у проект все необхідне для розробника. Кожен тег відповідає за відповідний матеріал. Наприклад, ми не можемо вмістити фотографію у тег “h1” тому що, цей тег відповідає за заголовок. Для вмісту з фотографіями створений інший тег “img” (Див. рисунок 1.8).

Тег	Запис	Значення
 	Непарний	Розрив рядка
<p>	Непарний	Новий абзац із відступом
<h1>-<h6>	Парний	Заголовок: <h1> — найвищий рівень, <h6> — найнижчий
<b>	Парний	Напівжирний шрифт
<i>	Парний	Курсив
<u>	Парний	Підкреслений шрифт
<font>	Парний	Розмір (size 1...7) і колір (color) символів

Рисунок 1.8 – Зразок тегів та їх опис

Хотілось би описати роботу кожного тегу, але щоб проговорити кожен тег потрібно буде запастись вільним часом, адже їх у HTML більше 100. Кожен з них важливий для написання правильної розмітки.

Цим і зручна мова розмітки html, що у ній є така велика кількість тегів та функцій, які можна використовувати для створення своїх проектів.

Але однієї розмітки html не вистачає, щоб розробити зовнішній вигляд елементів та змусити їх реагувати на дії користувачів. Через це, розробники використовують стилі CSS та мову програмування JavaScript.

1.3 Стилізація додатку за допомогою технології CSS

CSS (Cascading Style Sheets) - це мова стилів, яка використовується для оформлення і стилізації веб-сторінок. Вона дозволяє задати зовнішній вигляд елементів HTML, таких як кольори, шрифти, розміри, відступи, рамки та інше.

CSS використовується для стилізації та оформлення документів, котрі розмічені мовою розмітки HTML про яку я говорив раніше (Див. рисунок 1.9).

Рисунок 1.9 – Логотип CSS3

Для чого потрібен CSS? Чи можливо без нього створювати додатки? Ні. Хоч і в чистому HTML існує багато тегів для створення різних елементів, проте за їх вигляд він не відповідає.

За вигляд та коректне розміщення всіх блоків, елементів, матеріалів відповідає мова стилів CSS.

За допомогою лише одного HTML зробити цілий проект не получиться. Тут і приходить на допомогу ця технологія, яка допомагає вирішувати багато завдань, які відносяться до стильового оформлення сторінки.

1.4 Основні принципи технології CSS

Основні принципи технології CSS включають наступні поняття:

– Селектори. CSS використовує селектори для вибору елементів, до яких будуть застосовані стилі. Селектори можуть бути базовими (наприклад, назва елементу, клас або ідентифікатор) або складними (наприклад, комбінація селекторів, псевдокласів тощо).

– Властивості. CSS визначає різні властивості, які встановлюють стиль елемента. Властивості можуть включати кольори, шрифти, розміри, позиціонування, фони, рамки тощо. Кожна властивість має своє значення, яке встановлюється для відповідного елемента.

– Каскадування. CSS використовує механізм каскадування, який дозволяє визначити пріоритетність стилів. Коли кілька правил CSS застосовуються до одного елемента, використовуються правила спадкування, специфічності та порядку визначення для вирішення конфліктів та встановлення остаточних стилів.

– Боксова модель. Кожен елемент веб-сторінки може бути розглянутий як прямокутна область, відома як боксова модель. Вона включає контент, внутрішній відступ, зовнішній відступ та рамку. Властивості CSS, такі як width, height, margin та padding, використовуються для керування розмірами і відступами елементів.

– Розміщення. CSS дозволяє керувати розташуванням елементів на сторінці. Властивості, такі як position, display, float та flexbox, використовуються для зміни способу, яким елементи розміщуються і взаємодіють один з одним.

– Вкладеність. CSS дозволяє вкладати елементи один в одного, створюючи ієрархію структури. Це дозволяє змінювати стилі елементів на основі їх місця в структурі документа.

– Адаптивність. CSS пропонує різні техніки для створення адаптивних і резинових макетів, які змінюються залежно від розміру екрану або пристрою. Це включає в себе використання медіа-запитів, гнучкого розташування та резинових одиниць виміру.

Ці принципи дозволяють стилізувати і керувати виглядом веб-додатку, дозволяючи створювати красиві та функціональні користувацькі інтерфейси.

Ось як виглядав мій код CSS у проекті (Див. Рисунок 1.10).

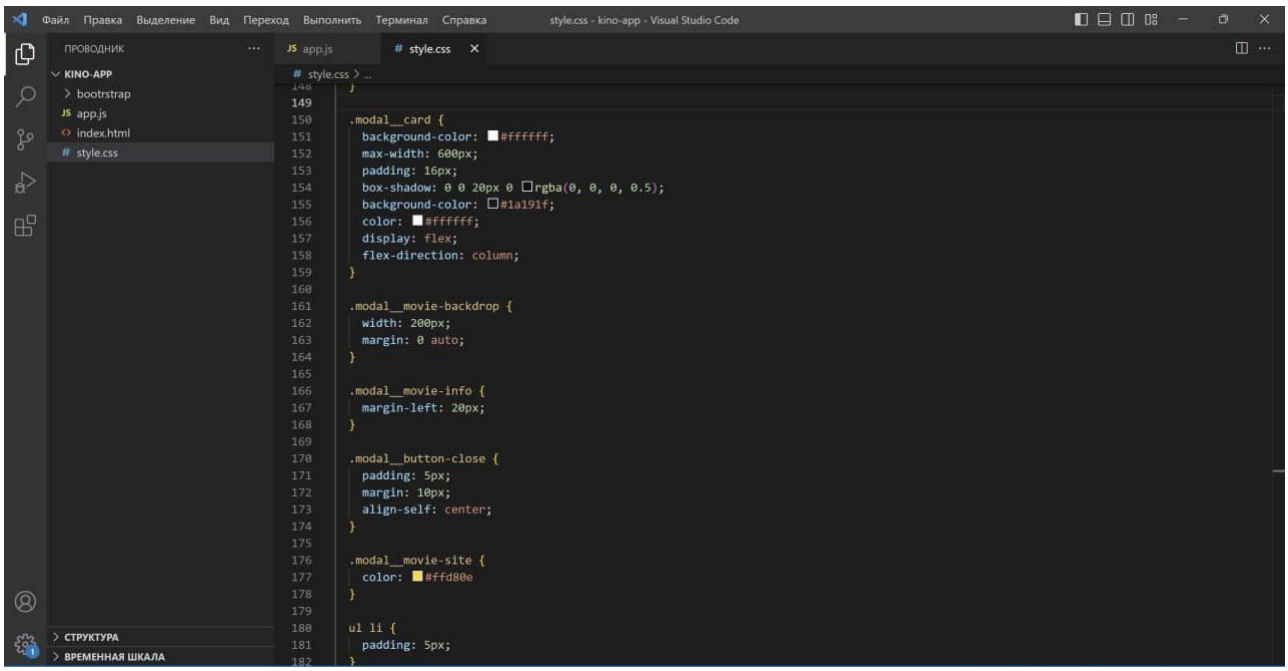

Рисунок 1.10 – Приклад CSS коду

Саме в цьому файлі, відбувалась основа стилізація всіх елементів на сторінці сайту. Шрифти, колір, розмів блоків та картинок, позиція кожного елементу та послідовність інформації.

Що дає розробнику користування мовою стилів. Використання мови стилів, такої як CSS, має кілька переваг для розробника:

– Розділення стилів і вмісту. Використання CSS дозволяє розділити логіку стилізації від HTML-структури документа. Це дозволяє розробникам легко змінювати вигляд веб-сторінки, не змінюючи сам вміст. Можна змінювати стилі однієї сторінки або використовувати ті ж самі стилі на різних сторінках.

– Повторне використання стилів. CSS дозволяє використовувати класи та ідентифікатори для застосування стилів до багатьох елементів на сторінці. Це сприяє полегшенню розробки та підтримки, оскільки ви можете застосовувати однакові стилі до різних елементів без необхідності повторювати код.

– Зручність в редагуванні. Використання CSS дозволяє легко змінювати стилі, навіть у великих проектах. Завдяки централізованому управлінню

стилями в окремому файлі CSS можна швидко знайти й змінити стиль, що застосовується до всіх відповідних елементів.

– Створення кращого користувацького досвіду. CSS дозволяє створювати привабливі та зручні для використання користувацькі інтерфейси. Тут можна задавати кольори, шрифти, розміри, анімацію та інші ефекти, щоб покращити візуальну привабливість та функціональність веб-додатку.

– Адаптивність та респонсивний дизайн. CSS дозволяє створювати адаптивні та респонсивні макети, що змінюються залежно від розміру екрану або пристрою. Можна використовувати медіа-запити та гнучке розташування, щоб адаптувати веб-додаток до різних пристроїв і забезпечити зручний досвід користувача.

Загалом, використання мови стилів, такої як CSS, дозволяє розробникам ефективно стилізувати та управляти виглядом веб-додатків, покращувати їх естетику та функціональність, забезпечуючи зручний користувацький досвід.(Див. Рисунок 1.11).

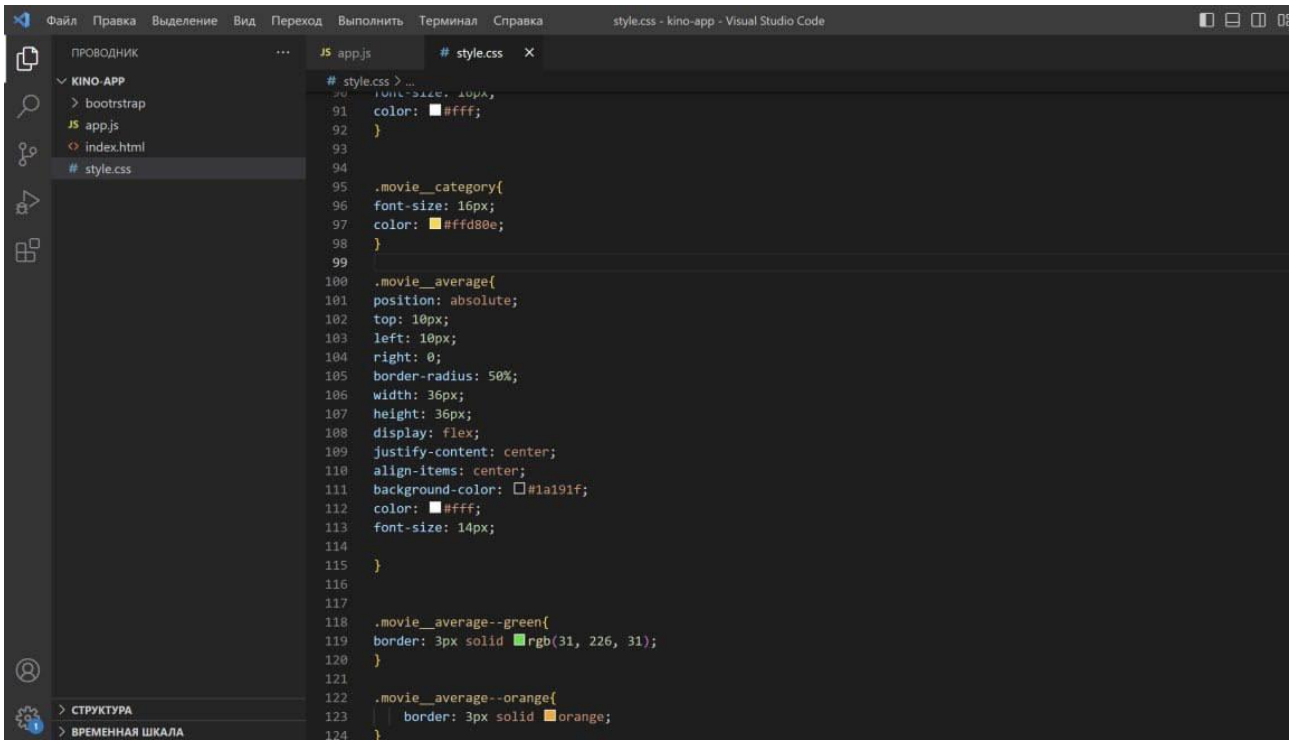

Рисунок 1.11 – Зразок стилізації блоку меню

Отже, стилі дають розробнику можливіть розмежувати етапи створення html-файлу й модернізовувати зовнішній вигляд сторінки.

1.5 Підключення фреймворку Bootstrap до основної стилізації коду

Bootstrap – Це CSS-framework. Bootstrap на сьогодні це один з найпопулярніших фреймворків HTML, CSS і JS (Див. рисунок 1.12).

Рисунок 1.12 – Логотип фреймворку Bootstrap

Це набір інструментів для створення сайтів. Він побудований так, що в ньому вже знаходиться готовий код який можна використовувати для побудови та розміщення блоків у верстці (Див. рисунок 1.13).

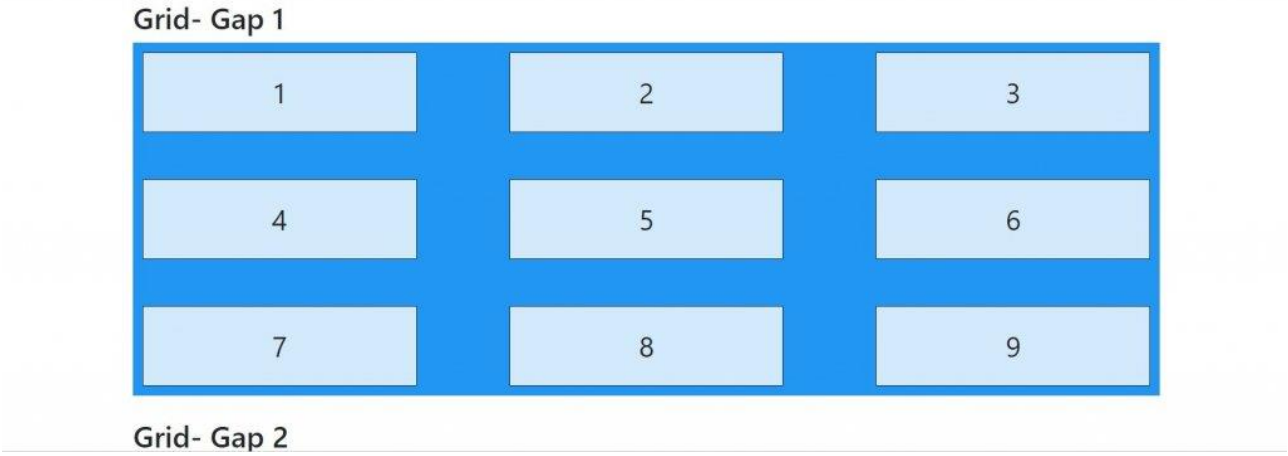

Рисунок 1.13 – Зразок стилізації блоків у Bootstrap

Bootstrap робить адаптивний веб-дизайн реальністю. Це дає змогу веб-сторінці чи додатку визначати розмір і орієнтацію екрана відвідувача та автоматично адаптувати відображення відповідно до цього (Див. рисунок 1.14).

Рисунок 1.14 – Зразок Grid системи у Bootstrap

Підхід, орієнтований на мобільні пристрої, передбачає, що смартфони, планшети та мобільні програми для конкретних завдань є основними інструментами співробітників для виконання роботи.

Bootstrap відповідає вимогам цих технологій у дизайні та включає компоненти інтерфейсу користувача, макети, інструменти JavaScript і структуру реалізації.

Програмне забезпечення доступне попередньо скомпільоване або у вигляді вихідного коду.

Основним напрямком фреймворку визнана розробка складних мобільних проектів. Основна перевага Bootstrap полягає в тому, що це не тільки фреймворк до CSS, але і Javascript-бібліотека.

У Bootstrap вже розроблені готові стилі та скрипти якими можна користуватись, а підключати їх можна за допомогою готових класів, які прописуються в тегах html..

В своєму проекті я також вирішив підключити та використовувати цей фреймворк, адже з ним дуже сильна економія часу та коду.

Особливо легко він підключається до проекту, можна підключити через тег “link” , або через JavaScript.

Я раджу підключати і першим і другим способом. Часто буває так, що виникає помилка при підключенні, і вона може залежати від різних причин, що спричиняє труднощі під час роботи. Тож підключення всіма можливими способами не буде зайвою (Див. рисунок 1.15).

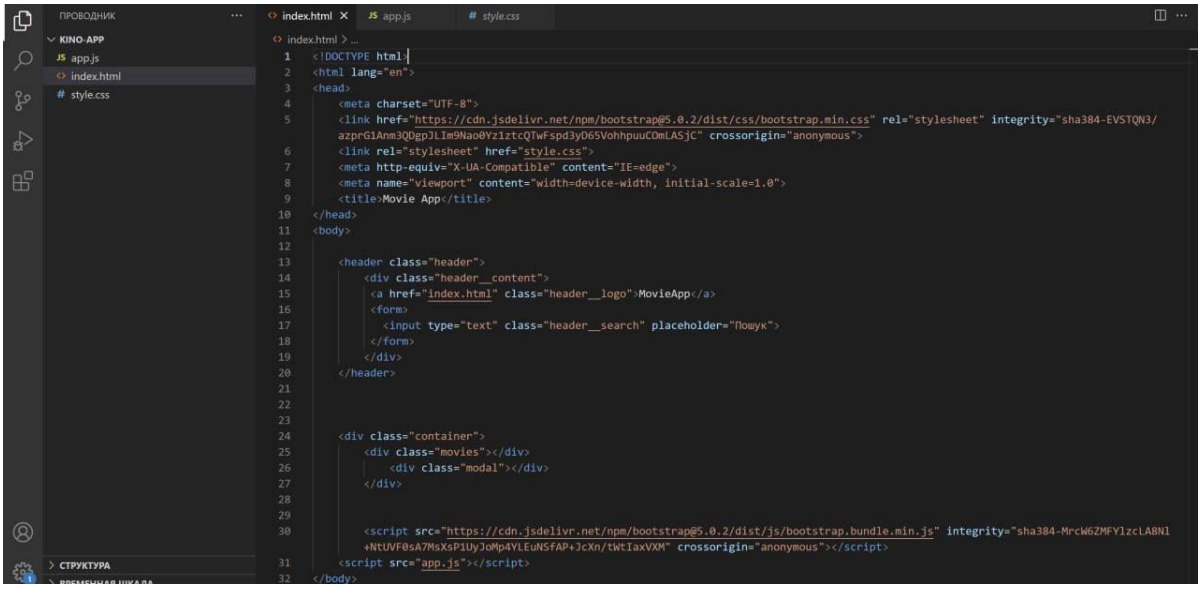

Рисунок 1.15 – Підключення Bootstrap до проекту

В перших рядках коду відбулось підключення через тег “link” , а вже в самому низу я ще раз підключив фреймворк через “script”. Що гарантувало мені впевненість та правильність у роботі з цією технологіїю.

Сам код у bootstrap виглядає не так зрозуміло як у чистому CSS, но це не проблема, адже до цієї технології зазвичай вже беруться спеціалісти з досвідом, які вже мають певне розуміння та принцип роботи стилізації.

Ось як виглядав мій код, коли я міняв стилі в розмітці (Див. рисунок 1.16).

```
42          <div class="movie__cover-inner fs-0 pb-0 d-flex">
43          <img src="${movie.posterUrlPreview}" alt="${movie.nameRu}" class="movie__cover">
44          <div class="movie__cover--darkened"></div>
45      </div>
46
47      <div class="movie__info bg-transperent">
48          <div class="movie__title fs-6 mt-0 ms-0">${movie.nameRu}</div>
49          <div class="movie__category pb-1 mb-2 text-center">${movie.genres.map(genre=> ` ${genre.genre}`)}</div>
50
51          ${movie.rating && `
52          <div class="movie__average movie__average--${getClassByRate(movie.rating)}">${movie.rating}</div>`
53          }
54      </div>
55          `
```

Рисунок 1.16 – Зразок стилізації розмітки

Економія часу відбувається по тій причині, що стилі вже написані як готовий інструмент, “mt,” ”pb”, ”text-centert”, ”d-flex” про що я говорив раніше. Що і економить більше часу.

Но ще один головний плюс у цьому фреймворку є те, що надавати елементам стилі та оформлення можна прямо у розмітці html в назві классів тегів.

Тобто тільки я створив певний елемент, я відразу можу оформити його по дизайну, і мені не потрібно для цього переходити в інші папки, файли, або створювати окремі класси для стилізації конкретного елементу.

По закінченню написання розмітки та стилів основний шаблон виглядає наступним чином (Див. рисунок 1.17).

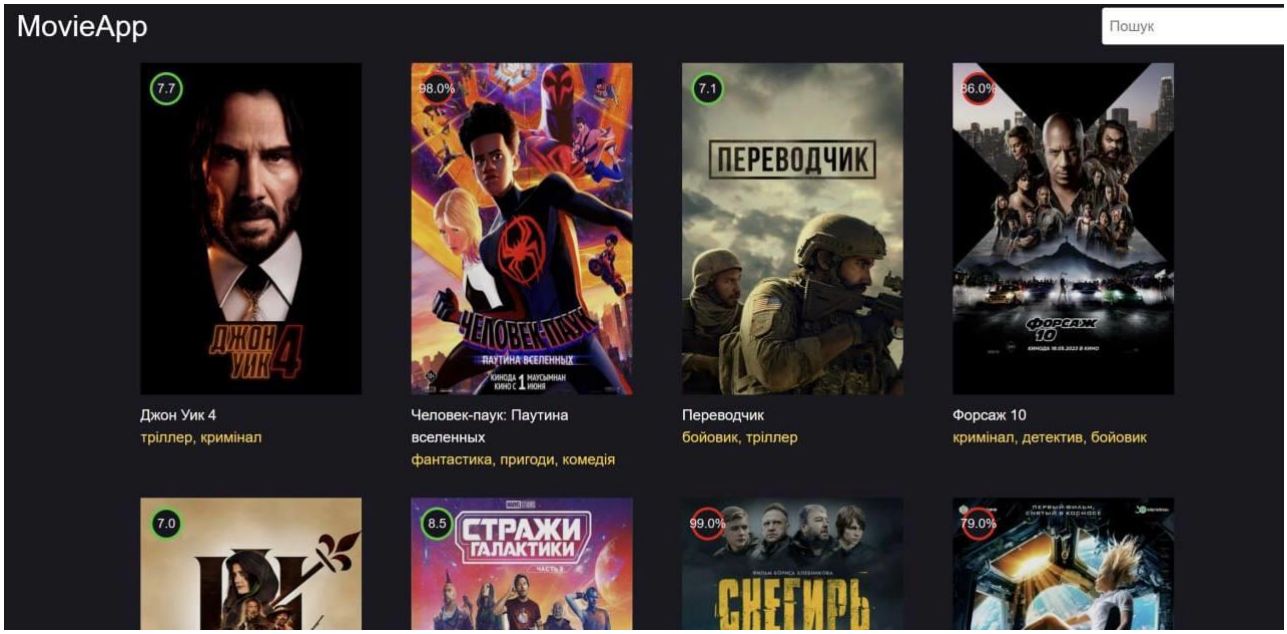

Рисунок 1.17 – Основний шаблон додатку

На основній сторінці присутній весь необхідний функціонал для зручного та комфортного перегляду фільмів та серіалів. Створена функція пошуку фільмів.

Елемент “input” з пошуком знаходиться в правому верхньому куті. Саме там користувач може легко та швидко знайти те що йому потрібно (Див. рисунок 1.18).

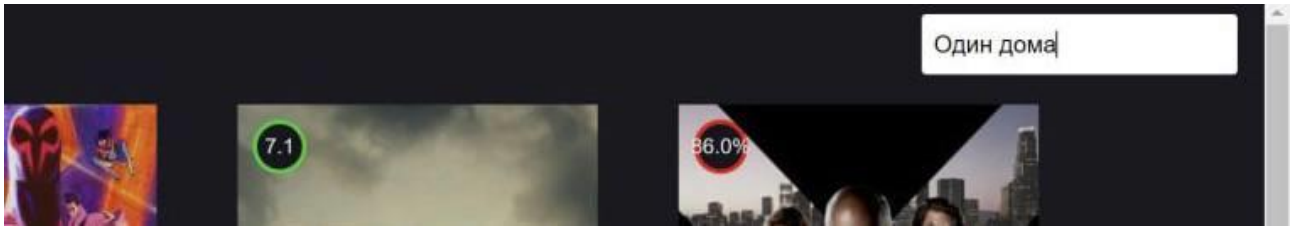

Рисунок 1.18 – Поле вводу для пошуку

Створення функції пошуку відбувалась наступним чином. Для початку елемент з пошуком “input” я записав у окрему константу.

Для того щоб можна було надати їй обробник подій, я записав сам елемент у константу та наклав на неї подію “submit” (Див. рисунок 1.19).

```
61
62  const form = document.querySelector('form')
63  const search = document.querySelector('.header__search')
64
65  form.addEventListener('submit', (e)=> {
66      e.preventDefault();
67
68      const apiSearchUrl = `${API_URL_SEARCH}${search.value}`
69      if(search.value){
70          getMovies(apiSearchUrl)
71          search.value = '';
72      }
73  })
74
75
76
```

Рисунок 1.19 – Зразок коду обробки функції пошуку

Після якої відправляється запит до сервера, та як результат, видає на головну сторінку те, що було записано у пошуковій системі (Див. рисунок 1.20).

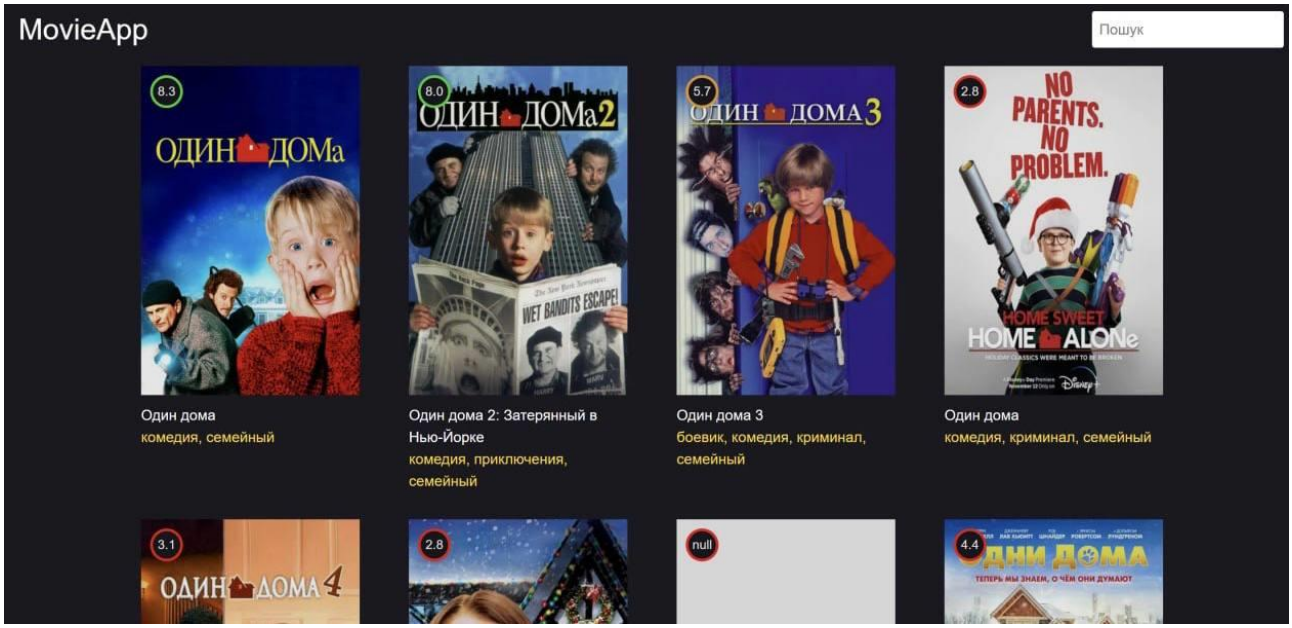

Рисунок 1.20 – Результат виконання пошуку фільму

Пошук фільму відбувається швидко, після завершення події “submit”, відпрацьовує функція, яка відправляла запит на сервер. Та показує на сторінці перші результати які були найдені у базі даних.

1.6 Сучасні веб-технології та архітектура сайтів

Більшість веб-сайтів будуються завдяки мові розмітки HTML а також каскадних таблиць CSS. Ці технології є обов’язковими для кожного веб-розробника. Також частина сайтів створюється завдяки мові програмування JavaScript. Усі ці технології відносять до фронтенд частини сайту.

Наприклад, сама мова HTML допомагає на сторінці створити контент та структуру. Тобто з’являється на пустому полотні текст, картинки, відео, різні таблиці та інші об’єкти, які необхідні власнику сайту. А от CSS допомагає увесь цей контент структурувати та надати йому потрібного вигляду.

Тобто якщо не використовувати HTML, то від CSS не буде ніякої користі. Він суттєво доповнює функціонал мови програмування HTML, наприклад допомагає прибрати підкреслення під посиланням та має інші функції, що допомагають створити більш гарний вигляд для сайту.

JavaScript розширює функціонал та можливості веб-розробки. Завдяки цій мові збільшується інтерактивність сайту.

Архітектура веб-сайту визначає організацію та структуру веб-сторінок і їх взаємозв'язок. Вона визначає, як інформація буде розміщена, організована і доступна для користувачів. Основні принципи архітектури веб-сайтів включають:

– Структурованість ієрархії. Веб-сайт повинен мати чітку ієрархію сторінок. Зазвичай це включає головну сторінку (homepage), з якої можна перейти до різних секцій і підрозділів. Навігаційні меню і посилання повинні бути організовані логічно, спрощуючи користувачам пошук потрібної інформації.

– Консистентність дизайну. Важливо, щоб всі сторінки веб-сайту мали однаковий дизайн і стиль. Це включає використання однорідного оформлення, кольорової палітри, шрифтів, стилів кнопок, форм і інших елементів. Консистентний дизайн допомагає користувачам легше орієнтуватися на сайті і забезпечує єдність бренду.

– Ієрархія і структура URL. URL-адреси повинні бути логічними і інтуїтивно зрозумілими. Вони можуть відображати структуру сайту і розташування конкретної сторінки в ієрархії. Це сприяє кращій навігації, SEO-оптимізації та зручності використання.

– Зручна навігація. Навігаційна система повинна бути легкою у використанні та інтуїтивно зрозумілою. Вона може включати головне меню, додаткові посилання в підвалі сторінок, хлібні крихти і пошукову панель. Забезпечення зручної навігації допомагає користувачам швидше знаходити потрібну інформацію і поліпшує загальний досвід використання.

– Розширюваність. Архітектура веб-сайту повинна бути гнучкою і легко розширюватися. Це означає, що нові сторінки, розділи або функціональність можуть бути легко додані до сайту без необхідності переробки всієї структури.

– Відповідність до стандартів. Дотримання веб-стандартів, таких як HTML, CSS і доступність, є важливим для забезпечення сумісності веб-сайту з різними браузерами і пристроями. Це дозволяє забезпечити, що сайт буде належним чином відображатися і працювати на різних платформах.

Загальна мета архітектури веб-сайту полягає у забезпеченні логічної структури, зручної навігації та приємного користувацького досвіду. Добре спроектована архітектура веб-сайту допомагає залучити і утримати відвідувачів, спрощує розробку та підтримку, а також поліпшує SEO-показники.

HTTP (Hypertext Transfer Protocol) є протоколом передачі гіпертекстової інформації через мережу. Він використовується для передачі даних між веб-браузерами та веб-серверами і є основою для комунікації веб-сайтів.

Основні характеристики та принципи HTTP:

– Клієнт-серверна модель. Протокол HTTP використовує модель клієнт-сервер, де веб-браузер виступає в якості клієнта, який надсилає запити, і веб-сервер служить як сервер, який обробляє ці запити і повертає відповіді.

– Безстаний протокол. HTTP є безстаним протоколом, що означає, що він не зберігає жодної інформації про попередні запити клієнта. Кожен запит розглядається окремо, і сервер надсилає незалежну відповідь.

– Запити і відповіді. HTTP використовує різні методи запитів, такі як GET, POST, PUT, DELETE, для здійснення різних дій над ресурсами на сервері. Клієнт надсилає запит з методом і URL-адресою ресурсу, а сервер повертає відповідь, яка може бути HTML-сторінкою, зображенням, даними JSON і т.д.

– URI (Uniform Resource Identifier). Кожний ресурс на веб-сервері ідентифікується унікальним URI, який включає URL-адресу ресурсу та метод запиту.

– Структура повідомлення. Кожне HTTP-повідомлення складається з заголовків і необов'язкового тіла. Заголовки містять метадані про запит або відповідь, такі як тип контенту, код стану, кешування і інші параметри.

– Стан і коди відповіді. Сервер повертає код стану в HTTP-відповіді для позначення статусу виконання запиту. Наприклад, 200 OK означає успішну відповідь, 404 Not Found показує, що ресурс не знайдено, а 500 Internal Server Error показує, що сталася помилка на сервері.

HTTP також підтримує безпечний варіант - HTTPS, який використовує шифрування SSL/TLS для захисту передачі даних між клієнтом і сервером.

HTTP є основою взаємодії між веб-браузерами та веб-серверами, що дозволяє передавати запити та отримувати відповіді для завантаження веб-сторінок, виконання дій та обміну даними через Інтернет.

Рисунок 1.21 – HTTP протокол

Веб-сервіси (Web services) - це стандартний спосіб обміну даними між різними програмними системами через мережу Інтернет. Вони дозволяють різним програмам, розробленим на різних платформах і мовах програмування, взаємодіяти та обмінюватися даними незалежно від їх внутрішньої реалізації.

Основні характеристики веб-сервісів:

– Стандарти протоколу. Веб-сервіси використовують стандартні протоколи, такі як SOAP (Simple Object Access Protocol), REST (Representational State Transfer), XML-RPC (XML Remote Procedure Call) і інші, для обміну даними між клієнтом і сервером.

– Інтерфейси опису. Веб-сервіси надають інтерфейси опису, які описують доступні методи та параметри, які можуть бути викликані клієнтом. Описи інтерфейсів можуть бути представлені у форматі WSDL (Web Services Description Language), RAML (RESTful API Modeling Language), OpenAPI (раніше відомий як Swagger) та інші.

– Протоколи обміну даними. Дані, які передаються між клієнтом і сервером веб-сервісу, зазвичай представлені у форматі XML (Extensible Markup Language), JSON (JavaScript Object Notation) або інших форматах. Протоколи веб-сервісів дозволяють серіалізувати та десеріалізувати дані для передачі і обробки.

– Універсальність. Веб-сервіси можуть бути використані для різних типів додатків, включаючи веб-додатки, мобільні додатки, клієнт-серверні додатки, системи електронної комерції та багато інших. Вони дозволяють різним системам взаємодіяти та обмінюватися даними незалежно від їх платформи і мови програмування.

Веб-сервіси можуть бути реалізовані на різних технологіях і мовах програмування, і їх використання вимагає взаємозрозуміння між розробниками клієнтської і серверної сторін.

Вони забезпечують можливість інтеграції різних систем та додатків, розширюють функціональність та полегшують обмін даними в розподілених середовищах.

Веб-сторінки можуть і будуть виглядати досить різними одна від одної, але всі вони, як правило, мають однакові стандартні компоненти.

Шапка сторінки “Header”, в якому зазвичай позначається логотип, або основний заголовок. Панель навігації “Navigation bar”, посилання на основні розділи сайту.

Зазвичай це представлено кнопками меню, посиланнями або вкладками. Бічна панель “Side bar”, там знаходиться певна периферійна інформація, посилання, цитати, реклама тощо.

Деякі веб-сайти мають більше стовпців, деякі набагато складніші. За допомогою правильного CSS можна використовувати майже будь-які елементи, щоб охопити різні розділи та спроектувати додаток так, як потрібно.

У своєму HTML-коді можна позначати розділи вмісту на основі їх функціональних можливостей — можна використовувати елементи, які однозначно представляють розділи вмісту, описані вище.

Щоб реалізувати таку семантичну розмітку, HTML надає спеціальні теги, які можна використовувати для представлення таких розділів.

А допоміжні технології, як-от програми зчитування з екрана, можуть розпізнавати ці елементи та допомагати виконувати такі завдання, як «знайти головну навігацію» або «знайти основний вміст»

У архітектурі веб-додатків зазвичай виділяються різні компоненти, які спільно працюють для забезпечення функціональності та взаємодії з користувачем.

Основні компоненти веб-додатків включають:

Клієнтська частина (Client-Side). Це компонент, який працює на боці клієнта, тобто веб-браузера користувача. Він відповідає за відображення інтерфейсу користувача, взаємодію з користувачем та передачу запитів до серверної частини додатку.

Клієнтська частина використовує мови програмування, такі як HTML (HyperText Markup Language), CSS (Cascading Style Sheets) і JavaScript, для створення динамічних та інтерактивних веб-сторінок.

Серверна частина (Server-Side). Це компонент, який працює на боці сервера і обробляє запити від клієнта.

Він відповідає за обробку бізнес-логіки, доступ до бази даних, обробку запитів, генерацію веб-сторінок та відповідей, які надсилаються клієнту. Серверна частина може використовувати різні технології та мови програмування, такі як PHP, Java, Python, Ruby, Node.js і багато інших, для реалізації функціональності додатку.

База даних (Database). Це компонент, який використовується для зберігання та управління даними, які використовуються веб-додатком.

База даних дозволяє зберігати, організовувати та отримувати доступ до даних, які використовуються для відображення, обробки та зберігання користувальницьких даних. Популярні бази даних включають MySQL, PostgreSQL, MongoDB, SQLite та багато інших. (Див. рисунок 1.22).

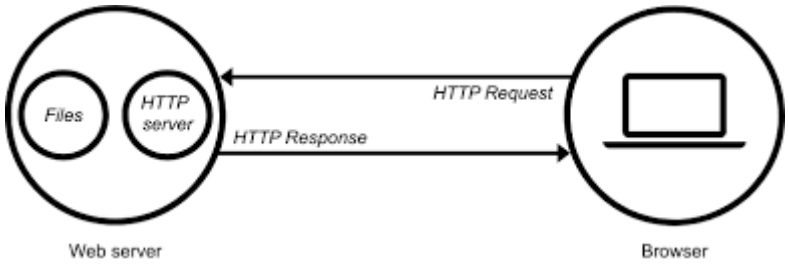

Рисунок 1.22 – Приклад взаємодії браузера з сервером

Ці три основні компоненти - клієнтська частина, серверна частина та база даних - працюють разом, щоб забезпечити функціональність та взаємодію веб-додатку.

Вони виконують свої відповідальності та взаємодіють між собою для створення повноцінного веб-додатку.

Більш надійною в плані захисту інформації є трьохрівнева архітектура, є представлено три рівні (Див. рисунок 1.23).

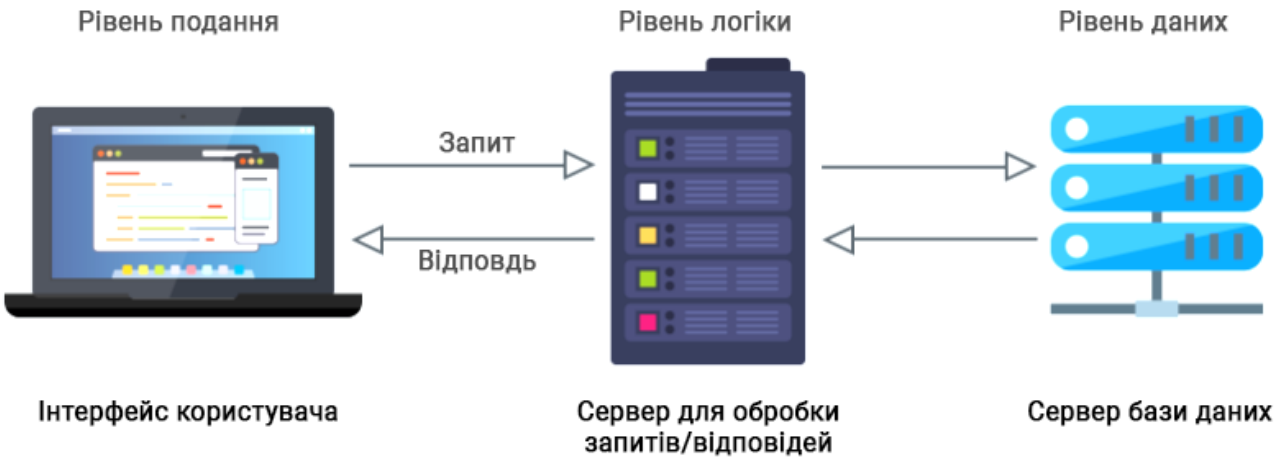

Рисунок 1.23 – Трьохрівнева архітектура

Трьохрівнева архітектура (також відома як трьохрівнева модель або архітектура з розділенням на рівні) - це популярний архітектурний шаблон для веб-додатків, який розділяє додаток на три логічні рівні або компоненти: представлення (Presentation Layer), бізнес-логіку (Business Logic Layer) і доступ до даних (Data Access Layer). Кожен рівень має визначені функції та відповідальності, а розділення між рівнями забезпечує модульність, розширюваність та зручну розробку додатків.

Основні рівні трьохрівневої архітектури:

– Представлення (Presentation Layer). Цей рівень відповідає за відображення інтерфейсу користувача та обробку взаємодії з користувачем. Він включає компоненти, такі як веб-сторінки, графічний інтерфейс користувача (GUI), веб-служби або API, які дозволяють користувачам взаємодіяти з додатком.

Інтерфейс прикладного програмування (API) (Див. Рисунок 1.24).

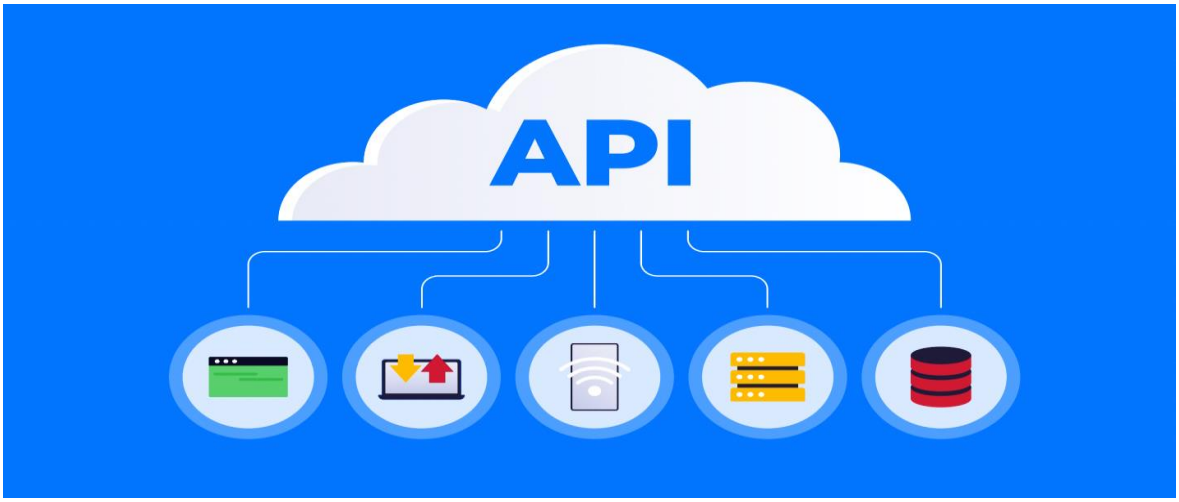

Рисунок 1.24 – API

Представлення передає запити користувачів до бізнес-логіки для обробки та отримує відповіді для відображення результатів користувачеві.

– Бізнес-логіка (Business Logic Layer). Цей рівень відповідає за обробку бізнес-логіки та логіку додатку. Він містить компоненти, які виконують розрахунки, перевірки, обробку даних та бізнес-правила, що визначають функціональність додатку. Бізнес-логіка отримує дані від представлення або доступу до даних, обробляє їх та генерує відповіді для представлення.

– Доступ до даних (Data Access Layer). Цей рівень відповідає за доступ до даних та зберігання інформації. Він включає компоненти, такі як бази даних, ORM (Object-Relational Mapping) або інші засоби доступу до даних. Доступ до даних забезпечує можливість зчитування та запису даних в базу даних, виконання запитів та маніпулювання даними.

Така архітектура дозволяє розділити веб-додаток на логічно залежні компоненти, що полегшує розробку, тестування, підтримку та масштабування. Кожен рівень має свою відповідальність та може бути розроблений і підтримуваний окремо, що дозволяє збільшити ефективність розробки та зменшити залежності між компонентами.

1.7 Висновок до першого розділу

В першому розділі кваліфікаційної роботи було описано середовище розробки веб-додатку, його переваги та можливості.

Також було висвітлено сучасні веб-технології, за допомогою яких створюються всі веб-додатки та сайти. Було проаналізовано з чого починається розробка любого веб-проекту, та описано будову і архітектуру сучасних веб-сайтів.

РОЗДІЛ 2. ПОБУДОВА ДОДАТКУ НА МОВІ ПРОГРАМУВАННЯ JAVASCRIPT

2.1 Мова програмування JavaScript

JavaScript – це мова програмування, яка дозволяє реалізувати ряд складних рішень в веб-документах. Вона допомагає зробити сторінки сайту інтерактивними та “живими”.

Через спеціальні функції написані на джаваскрипті, є можливість обробляти дії користувачів на сайті. JS — це одна з найпопулярніших мов програмувань на сьогодні. На ній пишуть близько 20 мільйонів девелоперів з усього світу (Див. рисунок 2.1).

Рисунок 2.1 – Логотип JavaScript

На цій мові програмування написано 97% сайтів з всього інтернету. Це говорить про те, що на який сайт ми б не потрапили — він побудований саме на цій мові програмування.

Головний плюс цієї мови полягає в тому, що можна працювати як у сфері Front End, так і у Back End. Тобто працювати із клієнтською базою, інтерфейсом, коректним функціоналом сайту, так і з серверною частиною проекту, архівом, базою данних, тощо.

Багато що можна реалізувати у свій проект завдяки JavaScript. Наприклад створення яскравої анімації і графічних об’єктів у форматі 2D/3D, опції прокрутки відеозаписів в медіа програвачі, створення ігор з анімаціями та логічними послідовностями. Та багато іншого можна створювати через код на JavaScript.

JavaScript працює в браузері користувача і дозволяє зробити веб-сторінки більш динамічними та інтерактивними. Ось деякі з основних можливостей та переваг JavaScript у розробці:

– Взаємодія з користувачем. JavaScript дозволяє створювати інтерактивні елементи на веб-сторінках. Ви можете обробляти події, які відбуваються на сторінці, такі як натискання кнопок, переміщення миші, введення тексту тощо. Ви можете реагувати на ці події, змінювати вміст сторінки, валідувати дані, анімувати елементи та багато іншого.

– Маніпуляція DOM. JavaScript надає доступ до Document Object Model (DOM), який представляє структуру веб-сторінки. Можна змінювати структуру, стилі, вміст елементів на сторінці за допомогою JavaScript. Це дозволяє динамічно оновлювати сторінку без перезавантаження, створювати нові елементи, змінювати атрибути та багато іншого.

– Робота з AJAX. JavaScript дозволяє виконувати асинхронні запити до сервера за допомогою технології AJAX (Asynchronous JavaScript and XML). Це дозволяє оновлювати частини сторінки без повного перезавантаження, отримувати та відправляти дані на сервер асинхронно. Це дозволяє реалізувати динамічне завантаження контенту, обмін даними з сервером та покращення користувацького досвіду.

– Розширення функціональності браузера. JavaScript надає доступ до різних API браузера, які дозволяють отримати інформацію про користувача, робити маніпуляції зі зображеннями, геолокацію, роботу з файлами, робити запити до веб-служб та багато іншого. Це дає можливість розширити функціональність веб-додатків і створити багатоцільові застосунки.

– Велика співпраця з HTML та CSS. JavaScript взаємодіє з HTML та CSS, що дозволяє динамічно змінювати вміст сторінки, стилізувати елементи, керувати класами, маніпулювати атрибутами та багато іншого. JavaScript також дозволяє створювати анімації, валідувати форми та забезпечувати багато інших взаємодій з HTML та CSS.

– Багатофункціональність. JavaScript підтримує різноманітні функції, такі як робота з рядками, масивами, об'єктами, регулярними виразами, математичні операції та багато іншого. Це дозволяє створювати складну логіку, обробляти та маніпулювати даними, виконувати розрахунки та створювати переваги у розробці.

– Велике співтовариство та ресурси. JavaScript має велике співтовариство розробників, існує багато ресурсів, форумів, бібліотек, фреймворків та інструментів, що полегшують розробку. Ви можете використовувати готові рішення та переваги, які надаються розширеннями та плагінами для JavaScript.

JavaScript є могутнім інструментом для розробки веб-додатків, який дозволяє створювати динамічні, інтерактивні та функціональні веб-сторінки. Він є широко використовуваною мовою програмування вебу та продовжує розвиватися, надаючи нові можливості для розробників.

Поряд з HTML і CSS, Джаваскріпт — третій важливий блок, на основі якого розробник створює більшість стандартних веб-інтерфейсів. Говорячи просто, ці 3 технології дуже підходять для роботи у розробці одна для одної.

Окремо, працювати буде дуже важко та не зручно. Поєднавши ці технології в один проект, можна створити великий, цікавий, та багатофункціональний проект для користувачів на різну тематику. Як в науковій сфері, так і розважальній.

2.2 Підключення JavaScript до розмітки

Якщо говорити про чистий JavaScript без бібліотеки або фреймворку, то підключення до проекту відбувається дуже просто, через тег “script”. В

середині тегу вказується шлях до файлу з розширенням “.js” в основній кореневій папці. Виглядає це ось так (Див. рисунок 2.2).

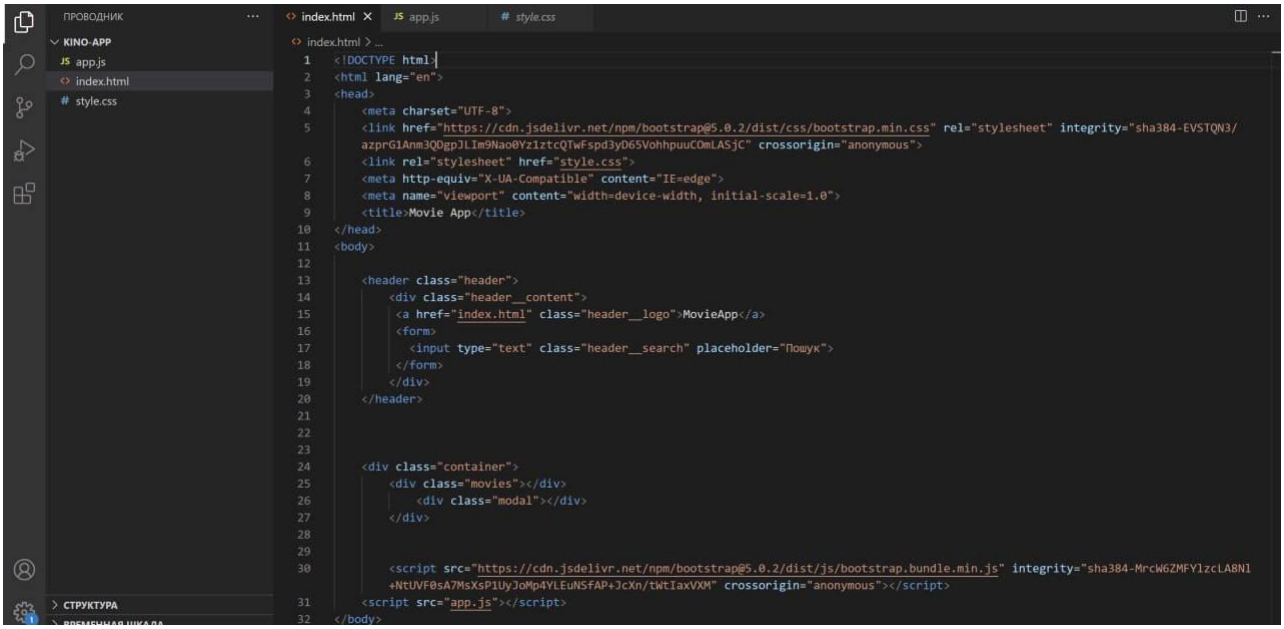

Рисунок 2.2 – Спосіб підключення JavaScript до проекту

На 31 рядку коду, з рисунка вище, вказано на шлях до файлу через тег “script” Це і було підключення JavaScript, для роботи із ним.

2.3 Інтерактивність та функціонування додатку через JavaScript

Інтерактивність в JavaScript – це простими словами коректне функціонування всіх елементів на сайті. До базових інтерактивних об’єктів відносяться елементи форм, посилання, кнопки, інпути, функції пошуку та багато інших елементів. У цьому проекті, мені довелося багато працювати саме над інтерактивністю. Щоб користувач при відвідуванні кіно-сайту, міг зручно та швидко знайти улюблений фільм або серіал.

Обробка подій в JavaScript дозволяє реагувати на дії користувача або інші події, що відбуваються в браузері. Це дозволяє створювати інтерактивні веб-сторінки та додатки. Основний принцип обробки подій в JavaScript полягає в

призначенні функції-обробника для певної події, і коли ця подія виникає, виконується відповідна функція.

Нижче наведені приклади деяких подій, на які можна реагувати в JavaScript.

Клік на елементі: Коли користувач натискає на елемент (наприклад, кнопку або посилання), можна виконати певну дію, таку як відкриття посилання або виклик функції.

Зміна значення поля вводу: Коли користувач вводить або змінює значення в полі вводу (текстове поле, чекбокс тощо), можна виконати певні дії, наприклад, перевірку даних або оновлення вмісту сторінки.

Відправлення форми: Коли користувач натискає кнопку "Відправити" у формі, можна перевірити та обробити дані форми, відправити їх на сервер або виконати інші дії, пов'язані з введенням користувача.

Завантаження сторінки: Коли сторінка повністю завантажується у браузері, можна виконати певні дії, такі як завантаження додаткового вмісту, ініціалізація додатку тощо.

Перетягування елементів (Drag and Drop): Коли користувач перетягує елемент на сторінці, можна відслідковувати цю дію та виконувати відповідні дії, наприклад, переміщення елементів або зміну їх положення.

Це лише кілька прикладів подій, на які можна реагувати в JavaScript. За допомогою обробки подій ви можете створювати інтерактивні, динамічні та відзивчиві веб-додатки, які сприяють зручності та взаємодії з користувачем. (Див. Рисунок 2.3).

Подія	Характеристика події	Обробник події
Click	Клік кнопкою миші на елементі форми або гіперпосилання	onClick
KeyDown	Натиск на клавіші клавіатури	onKeyDown
Load	Завантажується документ в браузер	onLoad
MouseDown	Натиск на кнопці миші	onMouseDown
MouseOver	Курсор знаходиться над елементом	onMouseOver
MouseOut	Курсор покидає зону над елементом	onMouseOut

Рисунок 2.3 – Зразок обробників подій в JavaScript

За допомогою однієї з цих подій, а саме «onClick» я створив функцію яка буде відображати колір рейтингу за нумерацією.

Якщо рейтинг більше семи, буде зелений колір, більше п’яти — оранжевий, в інакшому випадку буде червоний (Див. рисунок 2.4).

Рисунок 2.4 – Зразок if-else у JavaScript

Я вирішив написати таку функцію для того, щоб користувач міг відразу знати загальний рейтинг фільму або серіалу який має бажання переглянути. Це справді дуже зручно та швидко.

Не потрібно заходити у пошукову систему та витрачати час на те, щоб взнати який рейтинг того чи іншого фільму.

Була створення функція для відображення модального вікна про опис фільму, та закриття модального вікна (Див. рисунок 2.5).

```
const button = document.querySelector('.modal__button-close')
button.addEventListener('click', closeModal)
}




window.addEventListener('keydown', (event)=>{
if(event.keyCode === 27){
    closeModal()
}
})


window.addEventListener('click', (e) => {
if(e.target === modalEl){
closeModal()
}
})


function closeModal(){
    modalEl.classList.remove('modal--show')
    document.body.classList.remove('stop-scrolling')
}
```

Рисунок 2.5 – Функція виклику модального вікна

Ця функція дозволяла користувачу переглядати основну інформацію про фільм, одним кліком мишки по заставці. Та при необхідності закрити це вікно та продовжити пошук. На все вікно я добавив обробник подій “click”, та створив окрему функцію, яка слідкувала які дії робить користувач. Якщо користувач курсором миші натискав на любе місце окрім вікна з інформацією, то вікно закривалося.

Крім того я створив ще одну функцію, але на цей раз за допомогою обробника подій “keydown”

Якщо користувач на клавіатурі використовував клавішу “ESC”, функція спрацьовувала та модальне вікно закривалось.

У третій функції була реалізована зупинка “скроллінгу”. Тобто коли користувач відкриває модальне вікно щоб прочитати інформацію про фільм, задній фон всього вікна перестає рухатись у верх чи низ. Це було зроблено для того щоб користувач після перегляду інформації про фільм, міг закрити вікно, та повернутись на те саме місце сайту перед тим як відкрив це вікно.

Ось як виглядає саме модальне вікно з основною інформацією про фільм (Див. рисунок 2.6).

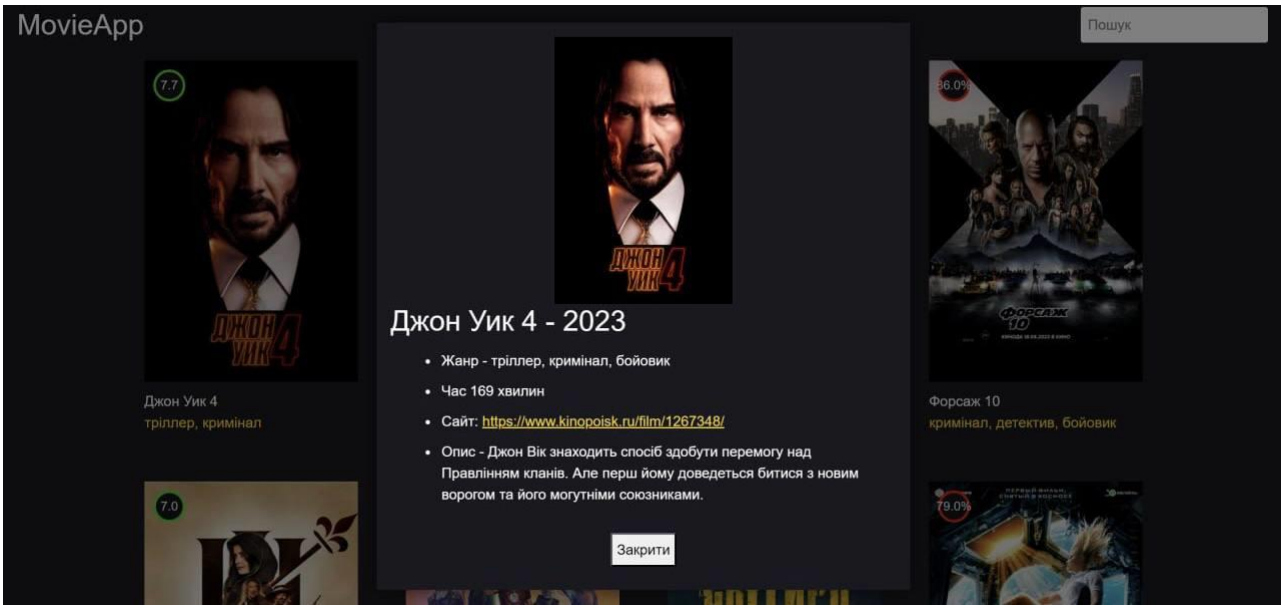

Рисунок 2.6 – Зразок модального вікна

В ньому описана основна інформація про фільм, жанр, тривалість, та опис фільму (Див. рисунок 2.7).

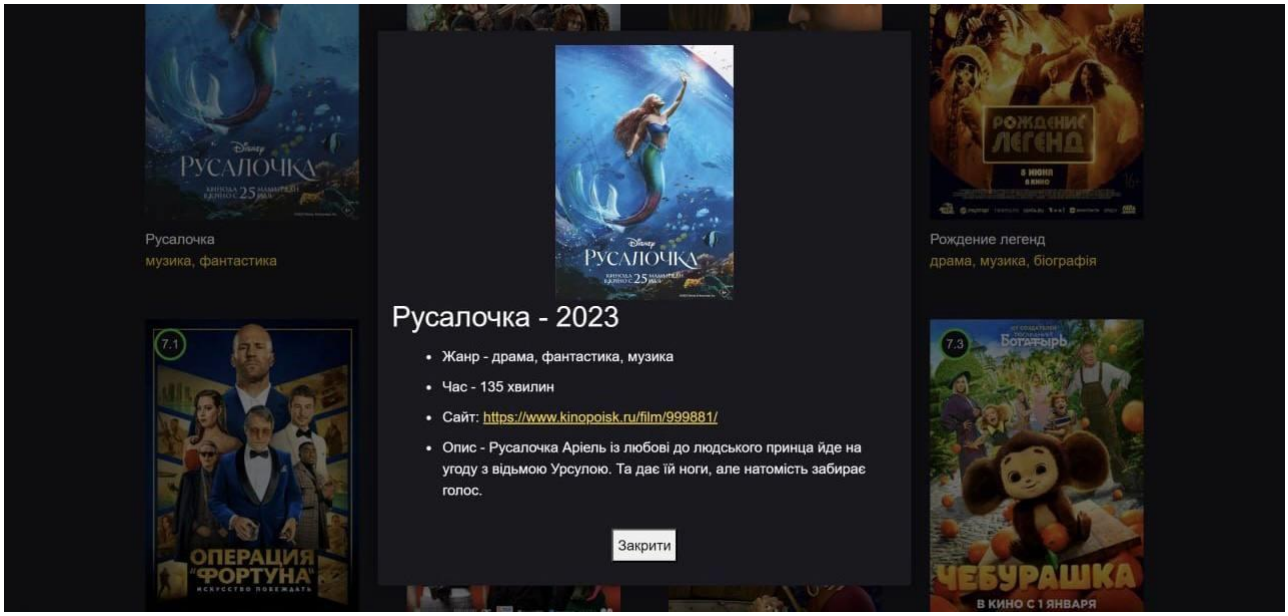

Рисунок 2.7 – Опис вибраного фільму

Серед поданою користовачу інформацією є посилання на сайт звідки за допомогою API, витягувалась основна інформація про фільм (Див. рисунок 2.8).

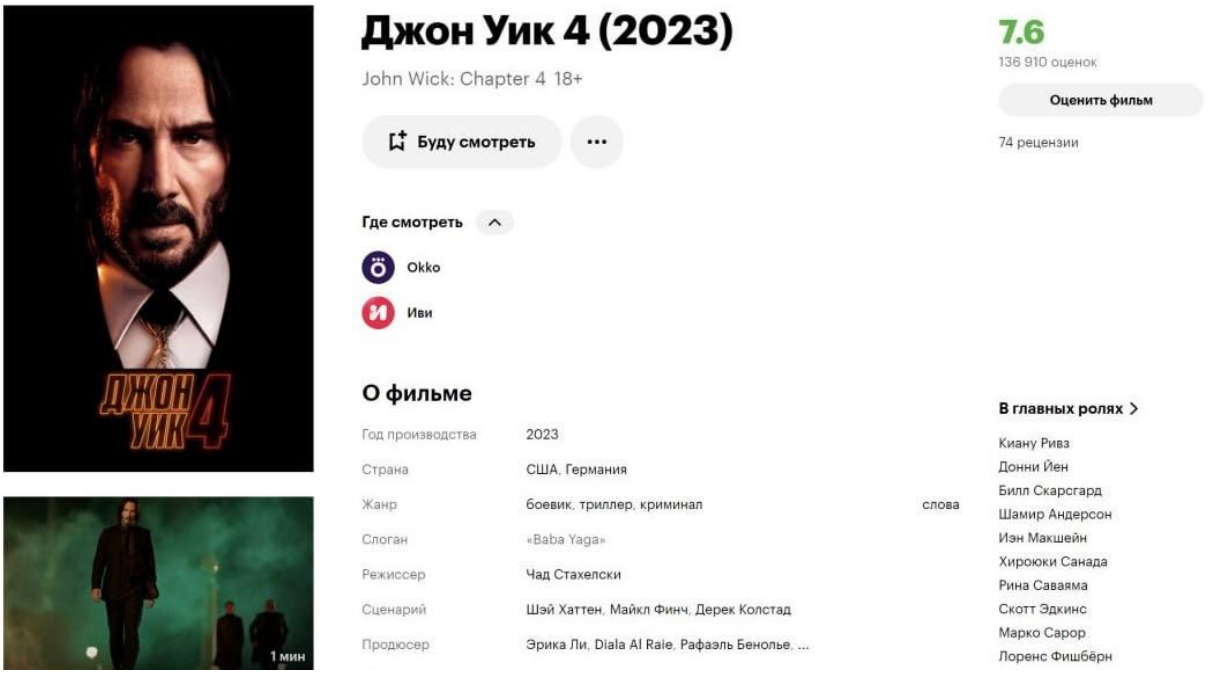

Рисунок 2.8 – Посилання на сайт про обраний фільм

На сайті більш детально описано всю інформацію, акторський склад, знімальну групу та інша інформація.

Також є можливість оцінити фільм та залишити свій відгук або враження, якщо користувач натисне на кнопку “Оцінити фільм”.

Після того як користувач залишив свій відгук, інформація про рейтинг фільму оновляється та визначається новий актуальний рейтинг. Оновлення інформації спрацьовує автоматично та виводиться на основну сторінку (Див. рисунок 2.9).

Рисунок 2.9 – Вигляд рейтингу фільмів

В лівому верхньому куті виведено загальний рейтинг фільму
Так виглядає сама розмітка модального вікна у файлі з розширенням js.

Для більш комфортного написання коду, та кращої читабельності, я переніс всю розмітку у джаваскріпт, та об’єднав її з скриптами.

Це суттєво покращило роботу проекту. Це набагато менше навантажує весь проект, та зменшує час його завантаження у браузері. Тому що перед тим як запускається локальний сервер, браузер спочатку читає весь код, і вже потім відображає його на головній сторінці. І в цей час відбувається завантаження.

Тому, чим більше коду, тим більше навантаження на сервер, та триваліший час завантаження. Саме такі дії покращують роботу веб-додатку (Див. рисунок 2.10).

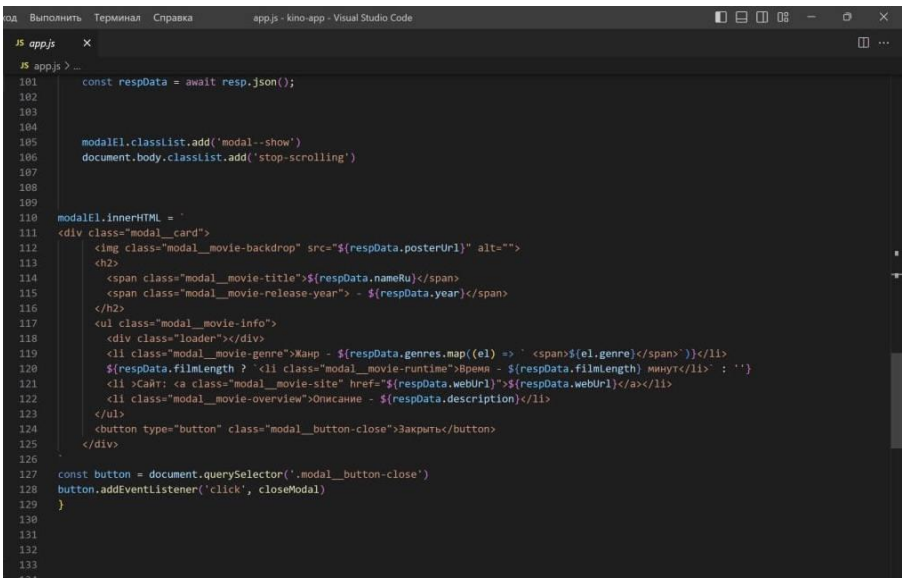

Рисунок 2.10 – З’єднання скриптів з розміткою

В ній вже створенні всі елементи, які користувач може бачити на головній сторінці.

Модальне вікно було інтегровано у JavaScript, для того щоб можна було легко взаємодіяти між HTML документами та обробниками подій у JavaScript. Іншими словами це називається об'єктна модель документа(DOM)

DOM (Document Object Model) в JavaScript - це представлення веб-сторінки або документа у вигляді структури, яка складається з об'єктів. Кожен елемент на сторінці, такий як кнопка або текст, представлений як окремий об'єкт. DOM дозволяє нам отримувати доступ до цих об'єктів та взаємодіяти з ними, змінюючи їх властивості, стилі, додавати або видаляти їх.

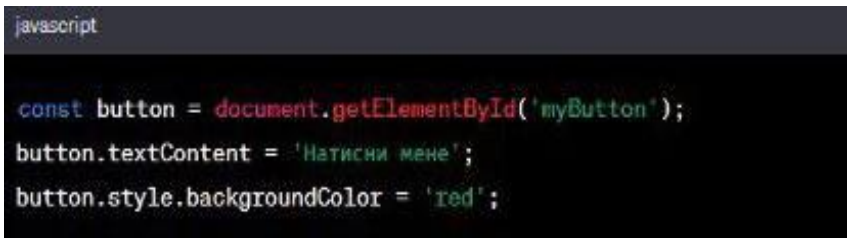

Рисунок 2.11 – Доступ до елемента через DOM

Наприклад, якщо на сторінці є кнопка з ідентифікатором "myButton", ми можемо отримати доступ до цієї кнопки в JavaScript за допомогою методу getElementById і змінити її текст або стиль (Див.рисунок 2.11).

Також, за допомогою DOM, ми можемо взаємодіяти з елементами сторінки, наприклад, додавати нові елементи, видаляти їх або змінювати їх розташування.

Отже, DOM в JavaScript дозволяє нам змінювати вміст і вигляд сторінки, додавати і видаляти елементи, змінювати їх властивості та взаємодіяти з користувачем на веб-сторінці.

2.3.1 API база даних

API (Application Programming Interface) - це набір правил та протоколів, що визначають спосіб взаємодії між різними програмами або компонентами програмного забезпечення. API визначає, які дії можна виконувати з програмного забезпечення, які дані можна отримувати або надсилати та які формати даних слід використовувати.

API може бути представлений у вигляді бібліотек, фреймворків або веб-сервісів, з якими ви можете взаємодіяти за допомогою JavaScript. Зазвичай, коли ми говоримо про веб-розробку, ми маємо на увазі веб-сервіси та API, доступ до яких здійснюється за допомогою HTTP-запитів.

За допомогою JavaScript можна взаємодіяти з різними типами API, наприклад через веб-сервіси API є можливість виконувати HTTP-запити з JavaScript, такі як GET, POST, PUT або DELETE, для взаємодії з веб-серверами. Відповідь на запит отримується у форматі JSON або XML, який можна обробити у своєму JavaScript-коді. Це дозволяє, наприклад, отримувати дані з сервера, надсилати дані на сервер або оновлювати сторінку без перезавантаження.

Браузерні API дозволяють взаємодіяти з різними можливостями браузера. Наприклад, можна використовувати API для маніпулювання DOM, створення

анімацій, роботи з кукісами (cookies), роботи з локальним сховищем (local storage) та багато іншого.

Існує безліч сторонніх API, які також можна використовувати в своїх JavaScript-проектах. Це можуть бути API для мап, соціальних мереж, платіжних систем, медіафайлів та багато іншого. Можна взаємодіяти з цими API, слідуючи їхнім документаціям та виконуючи необхідні HTTP-запити.

Усі ці API дозволяють розширити можливості JavaScript-коду, отримувати дані з різних джерел, взаємодіяти з серверами та іншими програмними компонентами, що використовують ці API. Це дозволяє створювати більш потужні та інтерактивні веб-додатки.

У своєму проекті я створив три константи у яких записав всі ключі, які надавали доступ до бази данних де знаходились всі фільмі для перегляду, топ 100 популярніших фільмів, інформація про всі фільми яка взята з бази даних, та функція пошуку фільму по ключовим словам (Див. рисунок 2.12).

```
1  const API_KEY = "8c8e1a50-6322-4135-8875-5d40a5420d86"
2  const API_URL_POPULAR = "https://kinopoiskapiunofficial.tech/api/v2.2/films/top?type=TOP_100_POPULAR_FILMS&
3
4  const API_URL_SEARCH = "https://kinopoiskapiunofficial.tech/api/v2.1/films/search-by-keyword?keyword="
5
6  const API_URL_MOVIE_DETAILS = "https://kinopoiskapiunofficial.tech/api/v2.2/films/"
7
8  getMovies(API_URL_POPULAR)
```

Рисунок 2.12 – Приклад використаних API

Разом з тим, мова не містить деяких корисних інструментів. Тут немає:

– Стандартної бібліотеки.
– Стандартних інтерфейсів для роботи з серверами і базами даних.
– Системи для керування пакунками.

Своїми словами, API надає нам можливість використовувати чужі напрацювання в своїх цілях. Отже, застосування API програмістам спрощує створення коду.

API виступає посередником між розробником додатків і будь-яким середовищем, з яким цей додаток повинен взаємодіяти (Див. рисунок 2.13).

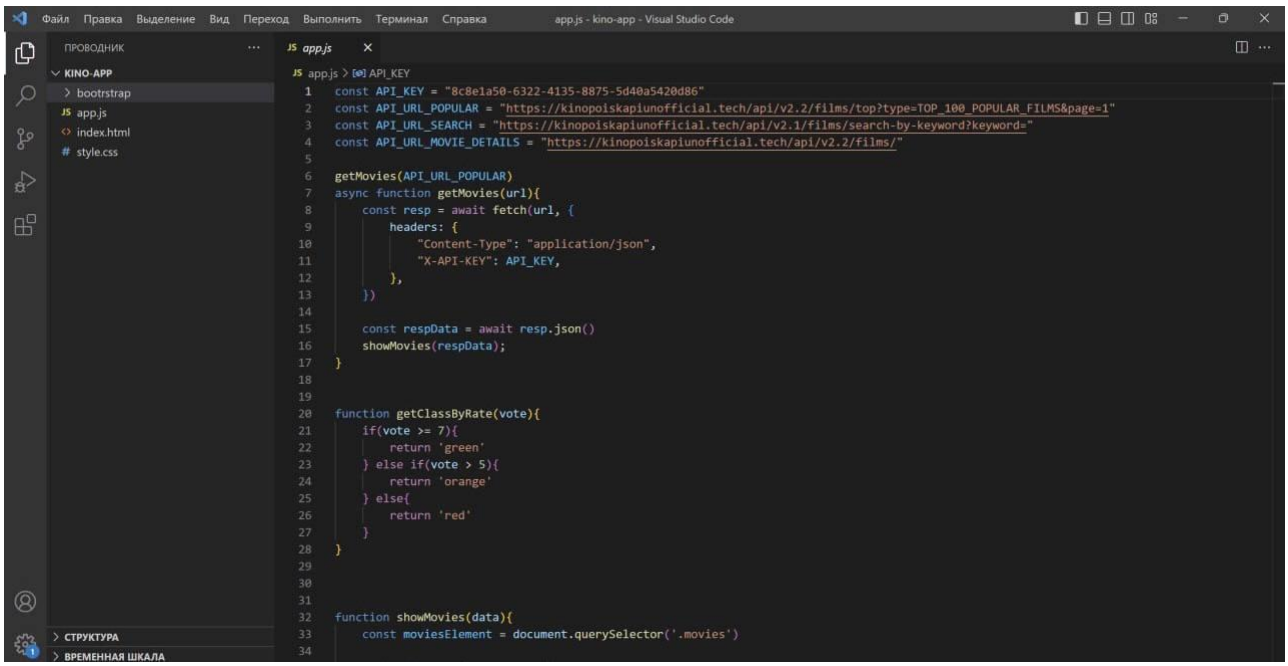

Рисунок 2.13 – Приклад роботи функції з API

Сучасні API часто приймають форму веб-сервісів, які надають користувачам (як людям, так і іншим веб-сервісам) якусь інформацію.

2.4 Висновок до другого розділу

В другому розділі кваліфікаційної роботи було описано будову мого веб-додатку, його підключення та обговорено принцип роботи.

Висвітлено основні технології та програми за допомогою яких був створений проект.

Було продемонстровано можливості веб-додатку, його функціонал та роботу як з сторони інтерфейсу, так і сервера.

РОЗДІЛ 3. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ХОРОНИ ПРАЦІ

3.1 Надзвичайні ситуації викликані пожежами, вибухами, техногенними та природними причинами

Надзвичайна ситуація – обстановка на окремій території чи суб'єкті господарювання, яка характеризується порушенням нормальних умов життєдіяльності населення, спричинених катастрофою, аварією, пожежею, стихійним лихом, епідемією, епізоотією, епіфітотією, застосуванням засобів ураження або іншою небезпечною подією, що призвела (може призвести) до виникнення загрози життю або здоров'ю населення, великої кількості загиблих і постраждалих, завдання значних матеріальних збитків, а також до неможливості проживання населення на такій території чи об'єкті, провадження на ній господарської діяльності.

Від надзвичайних ситуацій (НС) щорічно в Україні гине більше 70 тис. осіб, населення і держава зазнають значних матеріальних збитків.

Так, наприклад, у 2008 році внаслідок НС техногенного та природного характеру державі було завдано збитків на суму понад 4,7 млрд. грн., що у 5,7 раз перевищує показники 2007 року і майже в 11 разів втрати від НС 2006-го. При цьому понад 4,6 млрд. грн. Складають збитки від НС природного характеру.

Надзвичайні ситуації класифікують за характером походження, ступенем поширення, розміром людських втрат і матеріальних збитків.

Залежно від характеру походження подій, що можуть зумовити виникнення надзвичайних ситуацій на території України, визначаються такі види надзвичайних ситуацій: техногенного характеру; природного характеру; соціальні; воєнні.

НС техногенного характеру – це промислові, транспортні аварії (катастрофи) з вибухом, пожежі, аварії з викидом небезпечних хімічних,

радіоактивних, біологічних речовин, раптове руйнування споруд і будівель, аварії на інженерних мережах, гідродинамічні аварії на греблях, дамбах тощо.

НС природного характеру – це порушення нормальних умов життя і діяльності людей на окремій території чи об’єкті, пов’язане з небезпечним геофізичним, геологічним чи гідрологічним явищем (землетруси, повені, урагани, снігові замети), деградацією грунтів чи надр, пожежею у природних екологічних системах, зміною стану повітряного басейну, інфекційною захворюваністю та отруєнням людей, інфекційним захворюванням свійських тварин, масовою загибеллю диких тварин, ураженням сільськогосподарських рослин хворобами та шкідниками тощо.

Класифікація природних надзвичайних ситуацій.

Природні надзвичайні ситуації залежно від виду, масштабу та наслідків умовно поділяють на стихійні лиха (великі за масштабом і з важкими наслідками) та небезпечні природні явища.

В останній час все більше небезпечних явищ призводять до серйозних наслідків і розцінюються як надзвичайні ситуації. Наприклад, іній та обледеніння на початку минулого століття не завдавали серйозної шкоди, тоді як взимку 2001 року кілька областей України залишилися без електроенергії, що, звісно, завдало величезних матеріальних та економічних збитків.

Стихійні лиха - це небезпечні процеси літосферного, атмосферного, гідрологічного, біосферного або іншого походження таких масштабів, які призводять до катастрофічних ситуацій з раптовим порушенням систем життєдіяльності населення, руйнуванням і знищенням матеріальних цінностей, об'єктів народного господарства

Види стихійних лих: Метеорологічні: буря, ураган, смерч, засуха, значне підвищення чи зниження температури.

Тектонічні: землетрус, цунамі, виверження вулкану, зсув. Топологічні: повінь, селевий потік, лавина, каменепад, снігові замети, пожежа. Космічні: підвищене радіоактивне випромінювання, падіння великого космічного тіла.

Біологічні: аномальне підвищення кількості макробіологічних об'єктів, захворювання та враження рослин і тварин, епідемія.

Наслідки надзвичайних ситуацій техногенного характеру при аварії на вибухонебезпечних об’єктах. До вибухонебезпечних об’єктів відносять виробництва вибухових речовин тротил, тетрил, гексоген, нафтопереробні підприємства, млинарські комбінати та елеватори, деревообробні та інші підприємства, що використовують або виробляють горючі речовини та матеріали.

Аварії на таких об’єктах, як правило, супроводжуються техногенними вибухами. Під вибухом розуміється процес звільнення великої кількості енергії за короткий проміжок часу. В результаті вибуху речовина перетворюється в сильно нагрітий газ з дуже високим тиском, що впливає на навколишнє середовище, повітря, викликаючи його рух, і утворення чинників, що уражають.

Тому вони називаються уражаючими факторами. Основний уражаючий фактор вибуху – це повітряна ударна хвиля (УХ). Ударна хвиля – це зона сильно стислого повітря, що розповсюджується в усі боки від центру вибуху з надзвуковою швидкістю (більше 330 м/с).

Основним параметром УХ, що визначає її руйнівну дію, є надмірний тиск у фронті УХ.

3.2 Заходи щодо захисту установки від короткого замикання

Коротке замикання - це вид роботи електричного обладнання, яке виникає при зменшенні опору до значень близких до 0 Ом.

У більшості випадків електричне замикання є аварійним режимом роботи, що призводить до виникнення несправності обладнання, пожежі через значне нагрівання струмоведучих частин великим струмом.

Коротке замикання є нормальним режимом роботи для вимірювальних трансформаторів струму.

Електронагрівальні прилади перетворюють електричну енергію у теплову. Для цього перетворення використовують дроти з матеріалів які мають опір, зазвичай це ніхром або фехраль. Нагрівальні елементи отримують шляхом використання проволоки, яка зкручена у пружину.

Нагрівальні елементи розміщуються у жаростійкому матеріалі або підвішуються на спеціальні кріплення.

Запобіжник - це прилад, який призначений для швидкого відключення електрообладнання у випадку короткого замикання чи значного перевантаження за струмом.

Запобіжник являє собою проволоку певного перерізу яка плавиться і розриває електричний контакт якщо струм значно перевищує номінальне значення плавкої вставки.

Конструкція вставки буває різна.

На електричних підстанціях, де значні струми близько 600А, вставка являє собою набір пластин, які мають невелику площу з'єднання.

За невеликих номінальних струмів конструкція плавкої вставки спрощується і являє собою просто відрізок мідної проволоки.

В момент спрацювання запобіжника може виникнути електрична дуга.

Для запобігання її появі і швидкого згасання використовують пісок.

Корпус запобіжника виготовляють найчастіше з скла або фарфору. Скло використовують термостійке, яке у випадку великого тиску всередині розриває на пил.

Це робиться з метою безпеки, захисту від поранень людей частинками скла, які можуть перебувати поруч.

Струмообмежуючі запобіжники відрізняються простотою конструкції і як наслідок відносно невеликою вартістю.

Проте запобіжники мають ряд істотних недоліків: одноразову дію, нестабільні струмочасові характеристики, недостатню експлуатаційну надійність; обмежену зону використання по величині номінальних струмів номінальних напруг, некерованість від зовнішніх пристроїв, зокрема від

пристроїв релейного захисту трудність здійснення циклу АПВ ланцюга, що захищається. З урахуванням сказаного область застосування струмообмежуючих запобіжників тих, що існують конструкторські обмеження. Як правило, вони встановлюються в ланцюгах менш відповідальних споживачів.

Запобіжники можуть використовуватися як основні струмообмежуючі комутаційні апарати, включені безпосередньо в ланцюг, що захищається, так і як допоміжні апарати, що шунтують, наприклад, в нормальному режимі секційні або лінійні реактори електроустановки.

Обмежувачі ударного струму вибухового дії є надшвидкодіючі керовані комутаційні апарати одноразової дії з відносно великим номінальним струмом.

Струмообмежуючі реактори можуть мати різне конструктивне виконання, а також технічні та техніко-економічні характеристики і параметри.

Струмообмежувальний реактор можна класифікувати за різними ознаками. Вони бувають з лінійною, нелінійною та обмежено-лінійною або квазілінійною характеристикою; без сталі (без магнітопроводу) і зі сталлю; зі стрижневою, броньовою, бронестержневою, тороїдальною, циліндричною системою, магнітом, проводом, нерегульовані, регульовані, керовані; з поздовжнім, поперечним і кільцевим підмагнічуванням; з масляною або сухою ізоляцією (бетонні реактори); секційні, лінійні і заземлюючі; одинарні та здвоєні.

В даний час в енергосистемах для обмеження струмів КЗ використовуються тільки нерегульовані реактори з лінійною характеристикою.

У мережах 6-10 кВ застосовуються одинарні та здвоєні бетонні реактори а в мережах 35-220 кВ - масляні реактори. Можливість створення і доцільність використання конкурентоспроможних реакторів інших типів досліджується в ряді організацій.

Самонакопичувальний реактор - це некерований реактор з нелінійною характеристикою (зі сталлю), нелінійність якої обумовлена насиченням магнітної системи або її частини полем обмотки змінного струму.

Реактор складається з магнітопровода (замкнутого або з повітряними зазорами) тієї чи іншої конструкції і посадженої на нього обмотки змінного струму.

Опір реактора нелінійно залежить від струму його обмотки. При відсутності

повітряних зазорів ця залежність подібна залежності $\mu = f(H)$.

Керовані реактори - це регульовані реактори зі сталлю, зміна параметрів яких здійснюється за рахунок підмагнічування магнітопроводу(зазвичай полем постійного або випрямленого струму).

Розрізняють керовані реактори з поздовжнім, поперечним і кільцевим підмагнічуванням. Шляхом спеціальної схеми з'єднання обмоток або їх відповідного взаємного 21 розташування в зазначених реакторах забезпечується розв'язка індуктивних зв'язків, ланцюгів постійного і змінного струмів.

Самонасичувальний реактор може працювати як на висхідній, так і на низхідній частинах своєї характеристики $xp = f(I)$. У першому випадку еквівалентний опір реактора зростає зі збільшенням струму і може змінюватися в межах від x4 до x5, а в другому випадку зменшується зі 22 збільшенням струму і може змінюватися в межах від x5 до x6.

Робота на висхідній частині характеристики більш краща, тому що при цьому можна отримати бажаний струмообмежувальний ефект без появи нелінійних спотворень параметрів режиму.

Однак як з технічних, так і по технікоекономічним умовам забезпечення роботи реактора на висхідній частині характеристики при широкому діапазоні зміни струму в його обмотці вельми складно.

При роботі на низхідній частині характеристики вдається отримати регулює ефект, однак при цьому з'являються нелінійні спотворення параметрів режиму.

На цій частині характеристики принципово можливо отримати також і струмообмежувальним ефект реактора, якщо зі збільшенням струму реактора буде забезпечено зменшення напруженості магнітного поля в магнітопроводі.

З урахуванням сказаного самонасичувальні реактори можуть або безпосередньо використовуватися в якості струмообмежуючих пристроїв, або входити в якості нелінійних елементів до складу більш складних струмообмежуючих пристроїв.

3.3 Висновок до третього розділу

В третьому розділі кваліфікаційної роботи описано надзвичайні ситуації які виникають через природні та техногенні катастрофи. Їх види та наслідки.

Проаналізовано які дії можуть спасти та допомогти якщо сталась катастрофа, та обговорено яких правил потрібно дотримуватись щоб запобігти пожежі чи іншої катастрофи через коротке замикання.

ВИСНОВКИ

В першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр»: описано сучасні та популярні технології за допомогою яких створюються всі веб-додатки та сайти. Розглянуто будову та архітектуру сучасних веб-сайтів.

Проаналізовано в якому редакторі коду краще створювати проект.

В другому розділі кваліфікаційної роботи розглянуто мову програмування JavaScript на основі якої розроблений проект. Було проаналізовано основні аспекти мови, головні функції для коректної роботи додатку. Також було реалізовано функцію пошуку фільмів за допомогою API.

У розділі «Безпека життєдіяльності, основи хорони праці було розглянуто які бувають надзвичайні ситуації, з яких причин вони виникають, та як запобігти небезпеці якщо надзвичайна ситуація сталась. Були описані типи надзвичайних ситуацій, та заходи безпеки щодо запобігання наслідків.

Також було обговорено про заходи щодо захисту установки від короткого замикання. Проаналізовано як запобігти короткому замиканню, та описано про техніку безпеки, якщо замикання все таки сталося.

ПЕРЕЛІК ДЖЕРЕЛ

1. Історія Інтернету від APARTNET до сьогодні [Електронний ресурс] // Ucloud. – 2019. – Режим доступу до ресурсу: https://ucloud.ua/istoriya-internetuvid-arpanet-do-sogodni/\.

2. Правильна структура веб-сайту під SEO: приклади, види і рекомендації з розробки структури [Електронний ресурс] // BMB. – 2021. – Режим доступу до ресурсу: https://www.bmb.com.ua/2021/02/seo.html.

3. HTML [Електронний ресурс] // Wikipedia. – 2022. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/HTML.

4. CSS [Електронний ресурс] // Wikipedia. – 2022. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/CSS.

5. PHP проти HTML – у чому різниця між PHP і HTML [Електронний ресурс] // Myservername. – 2020. – Режим доступу до ресурсу: https://uk.myservername.com/php-vs-html-what-is-difference-between-php.

6. Основні переваги СУБД MySQL [Електронний ресурс] // НТУ. – 2018. – Режим доступу до ресурсу: https://studfile.net/preview/5607354/page:3/.

7. Обґрунтування вибору технології розробки, програмного середовища та мови програмування [Електронний ресурс] // Студиопедія. – 2020. – Режим доступу до ресурсу: https://studopedia.ru/21_90335_obruntuvannya-viborutehnologii-rozrobki-programnogo-seredovishcha-ta-movi-programuvannya.html.

8. Москаленко А. В. Програмне середовище для веб розробки. [Електронний ресурс] / А. В. Москаленко // OSPanel.io. – 2022. – Режим доступу до ресурсу: https://ospanel.io/.

9. Діаграма прецедентів. [Електронний ресурс] // Wikipedia. – 2021. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Діаграма прецедентів

10. Що таке трирівнева архітектура? [Електронний ресурс] // Розвиток. – 2022. – Режим доступу до ресурсу: https://uk.theastrologypage.com/three-tier-architecture.

11. Павлов А. В. Огляд підходів до організації інтелектуального інтерфейсу користувача [Електронний ресурс] / А. В. Павлов // МАН. – 2019. – Режим доступу до ресурсу: http://www.mgua.irtc.org.ua/attach/IMCS/2019/10.pdf.

12. Лекція 3. GRID і бази даних [Електронний ресурс] // ShporaMe. – 2022. – Режим доступу до ресурсу: https://shpora.me/mike/db2.

13. Трач Р. В. Моделювання організаційної структури проекту [Електронний ресурс] / Р. В. Трач // Вісник НУВГП. – 2019. – Режим доступу до ресурсу: http://ep3.nuwm.edu.ua/16624/.

14. Різниця між логічною та фізичною моделлю даних [Електронний ресурс] // Sawakinome. – 2021. – Режим доступу до ресурсу: https://ua.sawakinome.com/articles/technology/difference-between-logical-andphysical-data-model-2.html.

15. Перехід до фізичної моделі бази даних [Електронний ресурс] // Підручники для студентів онлайн. – 2022. – Режим доступу до ресурсу: https://stud.com.ua/77259/informatika/perehid_fizichnoyi_modeli_bazi_danih.

16. Проектування інтерфейсу для сайту та онлайн сервісу [Електронний ресурс] // D.Logic. – 2022. – Режим доступу до ресурсу: https://dlogic.com.ua/ua/service/proektuvannya-interfeisiv/.

17. Як правильно оформити головну сторінку сайту [Електронний ресурс] // HOSTiQ. – 2021. – Режим доступу до ресурсу: https://hostiq.ua/blog/ukr/main-page/.

18. Тег «form» [Електронний ресурс] // htmlbook. – 2018. – Режим доступу до ресурсу: http://htmlbook.ru/html/form.

19. Оператор if [Електронний ресурс] // The PHP Group. – 2021. – Режим доступу до ресурсу: https://www.php.net/manual/ru/controlstructures.if.php.