НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
„Київський політехнічний інститут імені Ігоря Сікорського”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

“ЗАТВЕРДЖЕНО”

Керівник роботи

_____ Євгенія СУЛЕМА

“___” _____ 2024 р.

ПЕРСОНАЛЬНИЙ АСИСТЕНТ ДЛЯ ПІДБОРУ ФІЛЬМІВ З ВИКОРИСТАННЯМ МУЛЬТИМЕДІЙНОГО ПОМІЧНИКА ТА ГОЛОСОВОГО АСИСТЕНТА. РОЗРОБЛЕННЯ АРХІТЕКТУРИ І ЛОГІКИ ПРОГРАМНОЇ СИСТЕМИ ТА КЛІЄНТСЬКОЇ ЧАСТИНИ

Пояснювальна записка

Виконавець:

_____ Максим МОРОЗ

2024

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

REST (скор. від англ. Representational State Transfer) – стиль архітектури для розроблення вебсервісів, що дозволяє передавати дані між клієнтом і сервером.

DOM – Document Object Model (стандартний спосіб представлення вебсторінок за допомогою набору об’єктів, що визначає набір інтерфейсів, незалежних від платформи і мови, який дозволяє програмам і сценаріям динамічно змінювати структуру, вміст і стиль документів).

Virtual DOM – це техніка та набір бібліотек і алгоритмів, які дозволяють покращити продуктивність на клієнтській стороні, уникаючи прямої роботи з DOM шляхом взаємодії з JavaScript-об'єктом, що імітує.

DOM-дерево; ПЗ – програмне забезпечення, що складається з сукупності програм для виконання комп’ютером.

Машинне навчання – алгоритми із галузі штучного інтелекту, для яких характерне не пряме рішення поставленої задачі, а прогнозування результату після проходження процесу навчання на зібраних даних, використовуючи при цьому математичну статистику, теорію ймовірностей тощо.

API (скор. від англ. application programming interface) – прикладний програмний інтерфейс, який надає набір чітко визначних правил для взаємодії одних програм з іншими.

HTTP – HyperText Transfer Protocol (протокол передачі даних, що використовується в комп’ютерних мережах)

STT (скор. від англ. Speech-to-Text) – технологія перетворення голосу в текстову інформацію.

TTS (скор. від англ. Text-to-Speech) – технологія перетворення текстової інформації в аудіо файл, в якому озвучується людським голосом надані слова у вигляді тексту.

ВСТУП

На сьогоднішній день існує достатньо програмного забезпечення, яке застосовується в різних сферах життя. Очевидним є той факт, що розробники та дизайнери прагнуть створити такий застосунок, який би був найбільш зручним для використання кінцевими користувачами, тому що в світі загалом існує досить багато програмного забезпечення майже для всіх можливих задач, але користуються попитом лише ті програми, які виконують свої функціональні можливості за мінімальну кількість взаємодій, і причому коли ці взаємодії є інтуїтивними та зрозумілими. Тому в останні роки можна побачити стрімкий розвиток голосових асистентів, за допомогою яких можна швидко виконати будь-які дії з однієї «точки входу».

Даний проєкт присвячений розробленню персонального асистента для підбору фільмів, який об’єднує в собі мультимедійного помічника, функції голосового управління та систему персоналізованих рекомендацій. Однією з ключових складових цієї системи є модуль голосового інтерфейсу, який відповідає за розпізнавання голосових запитів, їх обробку та озвучування відповідей. Голосовий інтерфейс забезпечує користувачам можливість швидкого і зручного доступу до функцій пошуку та рекомендації фільмів, роблячи процес використання системи простішим та більш інтерактивним.

Метою даної роботи є розробка функціональної та зручної клієнтської частини системи, яка забезпечить ефективну взаємодію користувача з основними функціональними можливостями. Основні задачі, які вирішуються в рамках цього завдання:

- Розробка інтерфейсу користувача (UI) відповідно до вимог дизайну.
- Реалізація логіки взаємодії користувача з системою.
- Інтеграція клієнтської частини з серверною.

1.ТЕХНОЛОГІЇ ДЛЯ РЕАЛІЗАЦІЇ ГОЛОСОВОГО ІНТЕРФЕЙСУ

1.1 . Налаштування запитів до серверного API

Налаштування запитів до серверного API є критичним етапом у розробці клієнтської частини веб-додатка. Важливо правильно організувати обмін даними між фронтендом (клієнтською частиною) та бекендом (сервером), щоб забезпечити ефективну взаємодію, коректне оновлення інтерфейсу та безпеку даних. У рамках цього етапу Angular надає потужні інструменти для роботи з HTTP-запитами, які допомагають розробникам налаштувати запити до серверу.

В Angular для роботи з HTTP-запитами використовується HttpClientModule, який є частиною @angular/common/http. Цей модуль дозволяє надсилати HTTP-запити (GET, POST, PUT, DELETE) до сервера та отримувати відповідь від нього у зручному для подальшої обробки форматі.

1. Імпорт модуля HttpClientModule:

   Спочатку потрібно імпортувати модуль HttpClientModule у основний модуль додатка. Це дає можливість використовувати HttpClient в сервісах або компонентах для відправки запитів.
2. Налаштування HttpClient у сервісі:

   Для організації запитів зазвичай створюється окремий сервіс, який відповідає за комунікацію з API. Це дає можливість централізувати логіку обробки запитів та зробити її легкою для повторного використання.

Angular підтримує всі основні типи HTTP-запитів, які використовуються для взаємодії з сервером:

1. GET

   Цей метод використовується для отримання даних з сервера. Наприклад,

запит на отримання списку фільмів, на основі якого клієнт може відобразити їх у вигляді карток або списку.

- Запит на сервер може містити параметри для пагінації, фільтрації або сортування даних, що дозволяє отримати лише потрібну частину інформації.

2. POST

   Цей метод використовується для надсилання даних на сервер. Це може бути, наприклад, форма для реєстрації нового користувача, або надсилання запиту на додавання нового елемента до бази даних.

   - Дані можуть передаватися у вигляді JSON-об'єкта або форми. У разі використання JSON це забезпечує чистоту і зручність обміну даними.

3. PUT та PATCH

   Ці методи використовуються для оновлення існуючих даних на сервері. Наприклад, оновлення інформації про фільм або змінення налаштувань користувача.

   - PUT застосовується для повного оновлення ресурсу, тоді як PATCH — для часткового оновлення (наприклад, зміна тільки одного параметра).

4. DELETE

   Цей метод використовується для видалення ресурсу з сервера. Він може бути корисним для функціоналу видалення елементів, таких як фільм або коментар.

При здійсненні запитів до API важливо враховувати можливість обробки асинхронних відповідей. В Angular використовується Observable для обробки результатів запитів. Це дає змогу працювати з даними після їх отримання, а також управляти асинхронними подіями.

1.2 . Вибір технологій для реалізації клієнтської частини

У процесі вибору технології для клієнтської частини веб-додатка була обрана платформа Angular. Це рішення було обґрунтоване кількома важливими факторами, які роблять Angular одним з найкращих варіантів для створення масштабованих, підтримуваних і високопродуктивних веб-додатків.

Angular — це фреймворк для розробки односторінкових додатків (SPA), який забезпечує потужні інструменти для побудови складних веб-додатків з багатьма функціональними можливостями. Основними особливостями Angular є:

- Модульність: Angular дозволяє розділяти додаток на окремі компоненти і модулі, що сприяє зручності організації коду, його повторного використання та тестування.
- Декларативний підхід: Завдяки декларативним механізмам, таким як шаблони та зв'язок даних, Angular значно спрощує процес створення та керування інтерфейсами користувача.
- Зв'язок даних (Data Binding): Односторонній та двосторонній зв'язок даних дозволяє ефективно синхронізувати стан між компонентами та інтерфейсом.

Однією з основних причин вибору Angular стало те, що цей фреймворк надає багато вбудованих функцій без потреби в додаткових бібліотеках чи інструментах. Це включає:

- Інтегровану систему маршрутизації (Routing): Angular має потужний механізм маршрутизації, який дозволяє легко створювати односторінкові додатки, переходити між різними видами даних і підтримувати стани сторінок.
- Форми та валідація: Angular забезпечує розвинену систему для роботи з формами, включаючи реактивні форми (Reactive Forms) і

форми на основі шаблонів (Template-Driven Forms), що робить їх зручними для збору та валідації даних.

- Інтеграція з RxJS: Angular підтримує використання RxJS для асинхронного програмування, що забезпечує гнучкість при роботі з потоками даних, обробці подій, запитах до серверу та інших асинхронних операціях.

Angular має вбудовані інструменти для модульного тестування компонентів, сервісів і інших частин додатка. Завдяки використанню Jasmine і Karma для юніт-тестів, а також Protractor для інтеграційного тестування, Angular дозволяє легко тестувати кожну частину програми.

Angular був обраний для розробки клієнтської частини завдяки його комплексності, вбудованим можливостям для створення масштабованих веб-додатків, ефективній обробці асинхронних запитів, підтримці тестування та великій спільноті. Можливості Lazy Loading, Change Detection і RxJS дозволяють створювати високопродуктивні додатки, які працюють швидко і ефективно навіть при високих навантаженнях. Використання цього фреймворку дозволяє значно прискорити розробку додатка, забезпечити високу його стабільність та зручність для кінцевого користувача.

2. РЕАЛІЗАЦІЯ ГОЛОСОВОГО ІНТЕРФЕЙСУ

2.1 Обробка відповіді сервера на клієнті

У цьому підпункті розглянемо, як клієнтська частина реагує на дані, що надходять від сервера, використовуючи приклад пагінації, фільтрації контенту та мінімальної оцінки від глядачів. Цей процес демонструє інтеграцію клієнта з моделлю через серверну API.

1. Запит на сервер і обробка даних

- Пагінація

  Клієнт надсилає HTTP-запит на сервер для отримання певної сторінки з даними. Параметр пагінації (наприклад, page та limit) передається у query-параметрах URL. Клієнтська реакція: після отримання відповіді клієнт динамічно оновлює список відображених елементів (фільми, записи тощо) на основі поточної сторінки. Користувач бачить кнопки "Наступна сторінка" та "Попередня сторінка", які оновлюють параметр page при натисканні.

- Фільтрація контенту

  Запити з фільтрацією дозволяють отримати відфільтровані дані за обраними параметрами (наприклад, жанр, рік випуску). Клієнтська реакція: у відповіді сервер надсилає відфільтрований масив даних, який відображається на інтерфейсі. Користувач бачить лише ті елементи, що відповідають обраним фільтрам. Зміна фільтрів автоматично викликає новий запит до сервера та оновлення інтерфейсу.

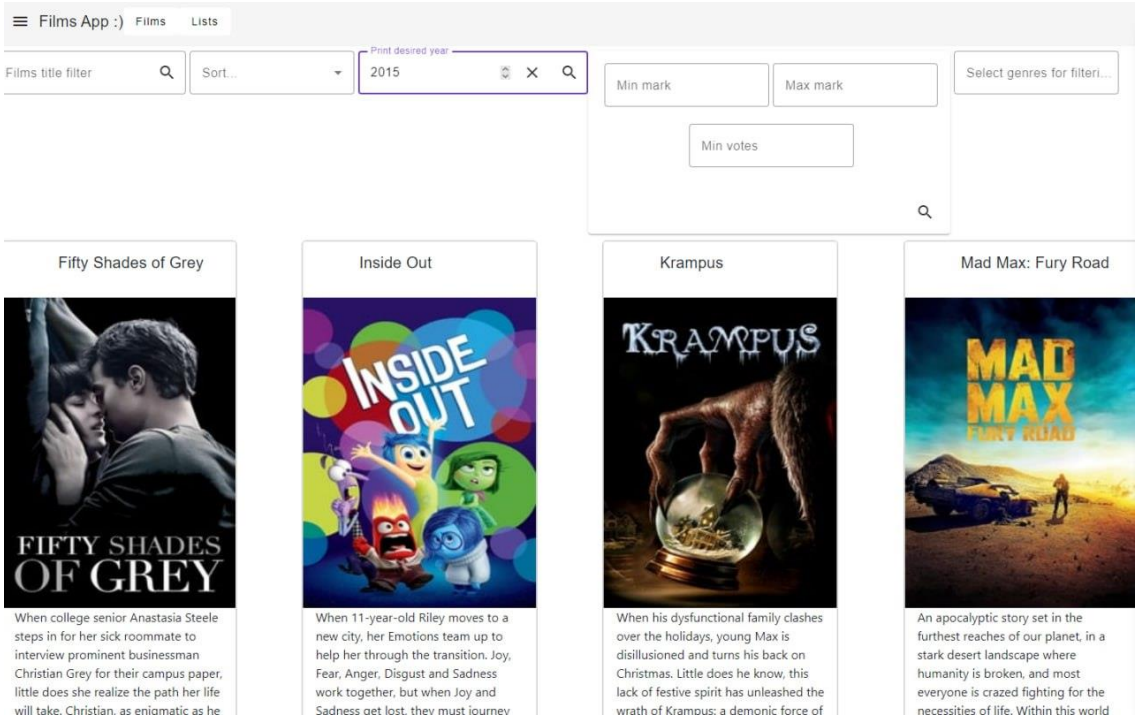

Рис. 2.1. Приклад фільтрації по фільмам, які вийшли у 2015 році

- Мінімальна оцінка від глядачів

  Користувач може встановити фільтр на мінімальну оцінку (наприклад, від 7 і вище за шкалою). Клієнтська реакція:

  Інтерфейс оновлюється і відображає лише ті фільми, що мають рейтинг більше або дорівнює встановленому значенню. Оновлений список підтягується із сервера в реальному часі.

2. Логіка взаємодії клієнта з моделлю

- Клієнт відправляє запит:

  Користувач через елементи інтерфейсу (випадаючі списки, повзунки, кнопки) задає необхідні параметри для фільтрації, пагінації чи встановлює мінімальний рейтинг. Ці дані формують запит до API.
- Сервер взаємодіє з моделлю:

  Сервер виконує пошук у базі даних (використовуючи фільтрацію, сортування чи обмеження кількості записів), формує відповідь і надсилає її клієнту.
- Клієнтська обробка відповіді:
  1) Отримані дані проходять парсинг на клієнті.
  2) За допомогою середовища Angular дані відображаються в інтерфейсі користувача у вигляді списку або карток.
  3) У разі помилки (наприклад, відсутності результатів) виводиться повідомлення "За вашим запитом нічого не знайдено".

Приклад роботи

1. Користувач обирає "жанр: комедія" та "оцінка: від 7".
2. Клієнт формує запит:

{

```
 "page": 1,
 "limit": 10,
 "total": 25,
 "movies": [
  {"title": "Movie 1", "rating": 8.0},
  {"title": "Movie 2", "rating": 7.5}
 ]
}
```


3. Клієнт оновлює інтерфейс, відображаючи відфільтровані фільми з можливістю переходу на наступну сторінку.

Таким чином, клієнт реагує на дані, забезпечуючи зручний користувацький досвід через динамічне оновлення інтерфейсу та інтеграцію з сервером.

2.2 . Оптимізація продутивності

Ледаче завантаження — це техніка, яка дозволяє завантажувати ресурси та модулі лише коли вони необхідні, а не на старті додатка. Це допомагає зменшити початковий час завантаження сторінки і оптимізувати використання пам'яті.

- Модульне ледаче завантаження: Angular підтримує ледаче завантаження через модулі, що дозволяє додавати нові компоненти і сервіси лише тоді, коли користувач переходить до відповідного маршруту.
- Динамічне завантаження компонентів: Якщо компоненти не використовуються на всіх сторінках, вони можуть бути завантажені лише тоді, коли користувач взаємодіє з певним елементом інтерфейсу.

Використання Change Detection Strategy: Angular використовує механізм Change Detection для відстеження змін у даних і оновлення відображення компонентів. За замовчуванням Angular використовує Default Change Detection

Strategy, що може бути неефективно, коли в додатку є велика кількість компонентів, що часто змінюються.

OnPush Change Detection Strategy: Ця стратегія дозволяє оновлювати компонент тільки коли:

- Змінилися вхідні дані (input).
- Відбулася подія (наприклад, клік або асинхронний запит).
- Це значно знижує кількість перевірок, що виконує Angular, і покращує продуктивність.

Віртуалізація дозволяє рендерити лише ті елементи, які видно на екрані, замість рендерингу всього списку відразу. Це значно знижує навантаження при роботі з великими наборами даних.

2.3 . Рекомендації щодо подальшого вдосконалення

В процесі роботи над інтерактивною системою пошуку фільмів з використанням мультимедійного помічника та голосового асистента було виявлено ряд напрямів та функцій, які можуть покращити розроблене ПЗ, а саме:

- Кешування запитів: Впровадити механізм кешування для популярних запитів за допомогою Redis або Memcached. Це дозволить зменшити навантаження на базу даних та скоротити час відповіді для часто використовуваних фільтрів чи сторінок.
- Оптимізація бази даних: Використати індекси для полів, за якими відбувається фільтрація та сортування (наприклад, rating, genre, year). Індексація допоможе пришвидшити пошук записів.

- Реалізація пагінації на стороні сервера: Використовувати пагінацію на сервері з параметрами LIMIT та OFFSET, щоб уникнути обробки великих обсягів даних.
- Додати інформацію про загальну кількість елементів (totalCount) у відповіді для зручного відображення навігації на клієнті.

ВИСНОВКИ

Розроблена клієнтська частина на Angular успішно забезпечує інтуїтивну взаємодію користувача з голосовим асистентом для пошуку фільмів. Завдяки використанню компонентного підходу та ефективному менеджменту стану, досягнуто високої продуктивності та масштабованості. Реалізована функція голосового пошуку, підкріплена технологією розпізнавання мови, значно спрощує процес пошуку фільмів. Інтеграція з серверною частиною забезпечує актуальність даних про фільми та персоналізацію рекомендацій. В процесі розробки було виявлено, що деякі анімації можуть сповільнювати роботу на слабких пристроях. Для вирішення цієї проблеми планується оптимізувати анімації та використовувати більш легкі бібліотеки.

Реалізація голосового пошуку, персоналізація рекомендацій та адаптивний дизайн інтерфейсу під різні розміри екранів сприяли підвищенню задоволеності користувачів. Використання матеріальних дизайну забезпечило сучасний та естетичний зовнішній вигляд додатка.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Angular. (n.d.). *Angular: A platform for building mobile and desktop web applications*. Angular. Retrieved December 16, 2024, from https://angular.io/.
2. Aslam, 54 – 57 Smedinghoff, J., & Dymek, M. (2019). Angular 8: Up and running: Learn modern web development with Angular 8. O'Reilly Media.
3. TypeScript Programming. [Електронний ресурс]. — Режим доступу: https://code.visualstudio.com/docs/languages/typescript
4. Llopis, R. (2020). Mastering Angular: Build enterprise-level applications with modern tools and techniques. Packt Publishing.
5. Riley, M., & Richards, G. (2017). *Learning Angular: A no-nonsense guide to building modern web applications with Angular*. Packt Publishing.
6. SMART: What is this and user manual. [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/SMART.
7. An introduction to Three.js. [Електронний ресурс]. — Режим доступу: https://humaan.com/blog/web-3d-graphics-using-three-js/