МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВОЛИНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЛЕСІ УКРАЇНКИ
Кафедра комп'ютерних наук та кібербезпеки

На правах рукопису

РУБІК АНАСТАСІЯ ДМИТРІВНА

РОЗРОБКА ОДНОСТОРІНКОВИХ ВЕБЗАСТОСУНКІВ З ВИКОРИСТАННЯМ API

Спеціальність: 122 Комп'ютерні науки
Освітньо-професійна програма: Комп'ютерні науки та інформаційні технології
Кваліфікаційна робота на здобуття освітнього ступеня «бакалавр»

Науковий керівник:
МАМЧИЧ ТЕТЯНА ІВАНІВНА,
кандидат фізико-математичних наук, доцент
кафедри комп'ютерних наук та кібербезпеки

РЕКОМЕНДОВАНО ДО ЗАХИСТУ
Протокол No_____
засідання кафедри комп'ютерних наук
та кібербезпеки
від _____ 2024 р.
Завідувач кафедри
(_____) Гришанович Т. О.

Луцьк – 2024

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AJAX – Asynchronous JavaScript And XML

API – Application Programming Interface

DOM – Document Object Model

JS – JavaScript

JSON – JavaScript Object Notation

SEO – Search Engine Optimization

SPA – Single Page Application

MPA – Multi Page Application

URL – Uniform Resource Locator

HTTP – Hypertext Transfer Protocol

HTTPS – Hypertext Transfer Protocol Secure

ВСТУП

Актуальність теми. Створення вебдодатків у сучасну цифрову епоху потребує нового підходу, орієнтованого на користувача. Позитивний досвід користування в Інтернеті має вирішальне значення для впізнаваності бренду. Як результат, з'явилася ідея створити онлайн-платформу, яка передбачатиме безперебійну взаємодію з користувачем. Саме ця потреба підштовхнула розробників до пошуку інноваційних способів створення вебзастосунків. Односторінкові вебзастосунки (SPA) є новою тенденцію, яка змінює підхід до створення вебдодатків.

Але що таке односторінкова програма і чому вона стала необхідною? У минулому перегляд вебсторінок був набагато повільнішим, ніж сьогодні. Кожен клік означав завантаження нової сторінки з сервера, що призводило до неприємних затримок. SPA змінюють цей сценарій. Вони лише оновлюють певні дані, які потрібні користувачеві, значно пришвидшуючи весь процес.

Мета роботи – дослідження та реалізація односторінкових вебзастосунків, аналіз зв'язку між односторінковими вебзастосунками та API, зосереджуючись на тому, як API підтримують динамічний характер SPA.

Для досягнення цієї мети потрібно виконати такі завдання:

- проаналізувати принципи та технології створення односторінкових вебзастосунків;
- розглянути технології використання API та сфери їх застосування;
- визначити переваги використання API в створенні SPA;
- розглянути існуючі приклади односторінкових вебзастосунків, на основі яких, визначити вимоги до кінцевого програмного продукту;
- проєктування вебсайту в SPA–форматі;

- створити інтерфейс та розробити функціональну частину вебсайту, реалізувати роботу з API для оновлення даних;
- забезпечення хостингу вебсайту;
- провести тестування кінцевого продукту;

Об’єкт дослідження – односторінкові вебзастосунки(SPA) реалізовані з API.

Предмет дослідження – розробка вебзастосунка прогнозу погоди, який використовує реальні даних про погодні умови з віддаленого сервера.

РОЗДІЛ 1. ДОСЛІДЖЕНННЯ ІСНУЮЧИХ ПІДХОДІВ ДО СТВОРЕННЯ ОДНОСТОРІНКОВИХ ВЕБЗАСТОСУНКІВ, ЇХ ЗВ’ЯЗОК З API.

1.1. Визначення односторінкових вебзастосунків

Вебсайт – це набір логічно пов'язаних між собою вебсторінок, розташованих у мережі під єдиним ім'ям – доменом. Вебсторінки – це документи, які написані на мові HTML і доступні в Інтернеті. Вони пов’язані між собою за допомогою гіперпосилань і гіпертексту, та мають спільний інтерфейс і дизайн. Вебсторінки містять певний контент, наприклад, статті, фото чи відео. Файли коду, хостинг, доменне ім'я, функції, дизайн та контент є основними компонентами сайту.

Сайти створюються для вирішення різноманітних проблем. Існують сайти-візитки, інформаційні сайти, інтернет-магазини та блоги. Наприклад, мета сайту-візитки – представити людину чи компанію, а ціль інтернет-магазину – продати товари.

Також існує багато різних підходів до створення вебсайтів.

Односторінковий вебсайт (SPA, або Single-Page Application) , як випливає з назви, функціонує на одній вебсторінці й не вимагає багаторазового оновлення сторінки під час взаємодії з користувачем. На відміну від традиційних вебдодатків, SPA використовують динамічні оновлення HTML і асинхронні методи отримання даних, щоб підвищити свою продуктивність[1].

Для того щоб краще розуміти односторінкові застосунки, розглянемо їх у порівнянні з багатосторінковими(рис. 1.1.1).

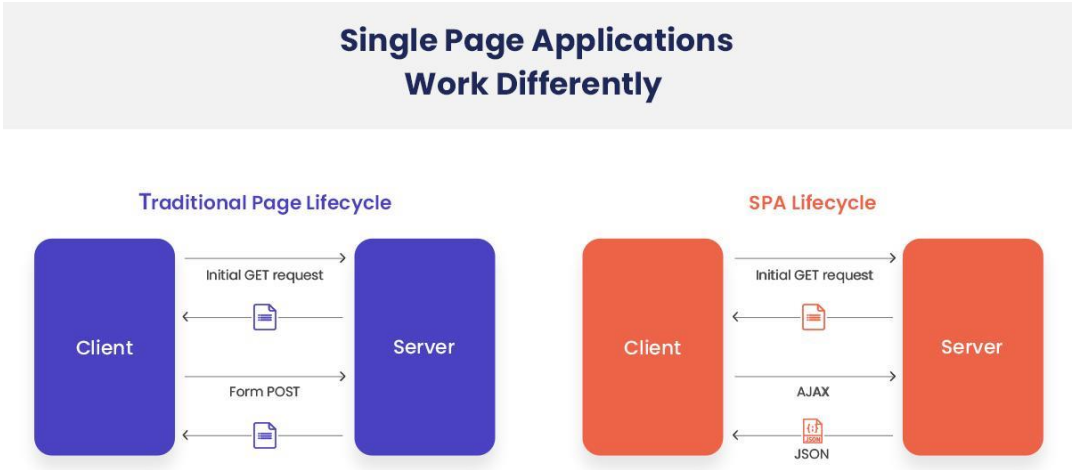

Рисунок 1.1.1 – Принципи роботи SPA і MPA

Кожного разу, коли користувач хоче скористатися певним вебзастосунокм, браузер надсилає відповідний запит на сервер(GET request). В результаті цього запиту сервер надсилає нову сторінку HTML. На цьому етапі в нас є два варіанти розвитку подій: односторінкова архітектура(SPA) і багатосторінкова архітектура(MPA). Коли ми виконуємо певну дію в MPA (наприклад, натискаємо на кнопку чи гіперпосилання), браузер надсилає новий запит до сервера(POST request), в результаті чого сервер, в залежності від тих даних, які він отримує від клієнта, генерує повністю нову HTML сторінку. Браузер в свою чергу завантажить цю нову сторінку з нуля. Тобто ми бачимо, що в цьому випадку сторінка повністю завантажується два рази.

З другого боку, коли ми працюємо з SPA, браузер використовує JavaScript для динамічного оновлення даних на сторінці. В цьому випадку при взаємодії з елементами сторінки JS скрипт надсилає певний запит до сервера (AJAX POST request), але, на відміну від звичайного POST запиту, сервер генерує не цілу сторінку, а лише дані, які потребує користувач. Ці дані зазвичай зберігаються у JSON-форматі(JSON – JavaScript Object Notation)(рис. 1.1.2).

```
{
  "results": [
    {
      "bill_id": "s1917-113",
      "chamber": "senate",
      "congress": 113,
      "number": 62,
      "question": "On Passage of the Bill S. 1917",
      "required": "1/2",
      "result": "Bill Passed",
      "roll_id": "s62-2014",
      "roll_type": "On Passage of the Bill",
```


Рисунок 1.1.2 – Приклад JSON-об’єкта

Після надсилання JSON-об’єкта сервер виконав своє завдання. Тепер він не генерує так багато коду, натомість клієнт має набагато більше роботи. Скрипт, який надіслав запит перебуває в очікуванні відповіді, і коли він отримає цю відповідь, запускається в дію певний JavaScript код, який може поступово оновлювати секції HTML сторінки.

Підсумовуючи, якщо користувач взаємодіє з сторінкою, працюючи з MPA, це провокує створення зовсім нової сторінки, що сильно відрізняється від SPA, де перезавантажуються лише ті частини сайту, з якими взаємодіє користувач, зберігаючи всі інші елементи сторінки.

1.2. Переваги і недоліки застосування SPA та MPA

Такі інформаційні гіганти, як Meta, YouTube і Netflix, перейшли від багатосторінкових до односторінкових застосунків. SPA забезпечують більш плавну роботу користувача та вищу продуктивність. Розглянемо переваги використання односторінкової архітектури[2]:

– Швидкодія: У односторінкових вебдодатках після завантаження сторінки сервер більше не надсилає HTML або CSS. SPA завантажують лише значно менший в розмірах файл JSON, а не нову сторінку, що передбачає кращу

швидкість завантаження та ефективність. Такий підхід забезпечує миттєвий доступ до всіх функцій сторінки без затримок.

– Менше навантаження на сервер: Односторінкова програма виконує лише один запит до сервера під час початкового завантаження та зберігає всі отримані від нього дані для подальшого використання.

– Покращений користувацький досвід: За допомогою SPA користувачі отримують доступ до сторінки, яка миттєво відображається з усім вмістом. Це зручніше, оскільки це надає можливість зручно та безперервно користуватися застосунком. Крім того, користувачі можуть отримати бажану інформацію, не натискаючи кілька посилань, як у MPA.

– Швидкий процес розробки: Під час процесу розробки фронтенд та бекенд частини SPA можна розділити, що дозволяє двом або більше розробникам працювати паралельно.

Незважаючи на всі переваги, односторінкові застосунки ще доволі новітня розробка, яка має свої проблеми і недоліки:

– Пошукова оптимізація(SEO): SPA виявилися неефективними в контексті SEO. Асинхронне завантаження даних, робота на JavaScript та одна URL адреса для кожної сторінки заважають пошуковим систем як Google або Yahoo сканувати односторінкові сайти. Додатково, процес налаштування SEO на готовому сайті – складний і дорогий процес.

– Повільне початкове завантаження: Час початкового завантаження сторінки може збільшитися через її складність, що негативно впливає на користувацький досвід.

– Складна реалізація аналітики сайту: SPA ускладнює визначення того, які секції вебсайту є більш популярними, враховуючи, що весь процес взаємодії з користувачем відбувається на одній сторінці.

– Проблеми безпеки: SPA більш схильні до злому. Вони дозволяють споживачам завантажувати весь застосунок, надаючи їм більше можливостей для пошуку вразливостей за допомогою зворотного проектування.

Не дивлячись на всі переваги SPA, підхід MPA все ще залишається популярним варіантом для деяких видів застосунків, через ряд своїх переваг[3]:

– Простота розробки для невеликих додатків: для невеликих додатків, де кожна сторінка має окремий вигляд або функцію, MPA можуть забезпечити менш складний процес розробки, що полегшує керування структурою та функціональністю додатка.

– SEO: MPA мають кращий підхід до пошукової оптимізації. Оскільки кожна сторінка має власну URL-адресу та вміст, пошукові системи можуть легко сканувати та індексувати сторінки, покращуючи видимість програми серед результатів пошуку.

– Сумісність із браузерами: MPA, як правило, більш сумісні з браузерами та різними пристроями, оскільки вони менше покладаються на JavaScript для основної функціональності. Це забезпечує кращу доступність для користувачів зі старими пристроями або браузерами з обмеженою підтримкою JavaScript.

– У MPA, якщо JavaScript не завантажується або не виконується належним чином, основні функції програми все ще можуть бути доступні. Це дозволяє користувачам взаємодіяти з програмою, навіть якщо певні функції обмежені або недоступні.

MPA мають і свої недоліки:

– Повільна робота: MPA часто вимагають багаторазового перезавантаження, що призводить до повільної та менш зручної роботи порівняно з SPA.

– Складність розробки: багатосторінкова програма зазвичай вимагає більшу кількість функцій порівняно з односторінковими програмами, тому їх створення вимагає більше зусиль і ресурсів.

– Гірший користувацький досвід: постійне перемикання між сторінками під час тривалих дій або складних процесів може призвести до втрати уваги користувача.

Отже, коли слід обирати SPA, а коли звертатися до більш традиційних підходів до розробки? Односторінкова архітектура ідеально підходить для соціальних мереж, платформ SaaS або деяких застосунків, де роль SEO не має великого значення.[4] Також, якщо вебсайт потребує хорошого та насиченого зв'язку з користувачами, то підхід SPA є обов'язковим. При цьому односторінкова архітектура програми найкраще підійде, якщо сайту потрібні оновлення в реальному часі на сторінці.

1.3. Архітектура односторінкових застосунків

Як було вказано вище, сторінка SPA-застосунка завантажується лише один раз і служить відправною точкою для решти програми. Це єдине повне завантаження сторінки, яке відбувається в SPA. Подальші частини програми завантажуються динамічно, без повної перезагрузки сайту, створюючи у користувача враження, що він переміщується між різними сторінками.

SPA поділяє додаток на безліч невеликих та незалежних блоків, які можуть бути модульно розроблені та повторно використані.[5] Такі компоненти представляють певний функціонал чи інтерфейс, їх можна комбінувати для створення більш складних додатків.

Після початкового завантаження сторінки всі інструменти, необхідні для створення та відображення компонентів сайту, завантажуються та готові до використання. Якщо потрібно відобразити новий компонент, він генерується локально в браузері та динамічно приєднується до DOM сторінки за допомогою JavaScript. Завдяки цьому сторінка не потребує перезавантаження.

Односторінкова програма працює на клієнтській стороні, що означає, що весь код виконується в браузері користувача, що робить програму динамічною і

швидкою. Всі запити та обробка даних відбуваються на стороні клієнта. Для роботи такої структури важлива маршрутизація. HTML такого вебсайту містить певні поля призначені для даних, отриманих з сервера. Щоб реалізувати подібний шаблон HTML, потрібна інтеграція з вхідними даними і управління всім процесом передачі даних з допомогою стороннього JavaScript файлу.

У SPA можна використовувати кілька підходів для отримання даних із сервера. Один із більш старомодних підходів це часткова візуалізація на стороні сервера, коли фрагменти HTML поєднуються з даними у відповіді сервера. Наразі найбільш поширеним варіантом є візуалізація на клієнтській стороні, що завжди виконується асинхронно з використанням API. Зв'язок із сервером здійснюється через AJAX, а дані зазвичай передаються в форматі JSON(JavaScript Object Notation).

1.4. Огляд API та їх роль в створенні SPA

У веб розробці API(Application Program Interface) зазвичай описують як набір різних запрограмованих функцій, які розробники можуть використовувати в своїх застосунках для взаємодії з компонентами інтерфейсу користувача або іншими вебсайтами чи сервісами[12].

Розглянемо принцип роботи API(рис 1.4.1). Допустимо, клієнт надсилає запит на сервер API, зазвичай через Інтернет або локальну мережу. Запит виконується за допомогою певного протоколу (наприклад, HTTP) і містить інформацію про операцію, яку хоче виконати клієнт(наприклад, отримання даних).

Сервер API отримує запит і обробляє його. Він може підтверджувати запит, автентифікувати клієнта, авторизувати запит або виконувати інші необхідні операції. Далі сервер API надсилає відповідь клієнту, яка може містити дані, код, що вказує на результат операції або повідомлення про

помилку. Після того як клієнт отримує відповідь і він може утилізувати її у своїй програмі.

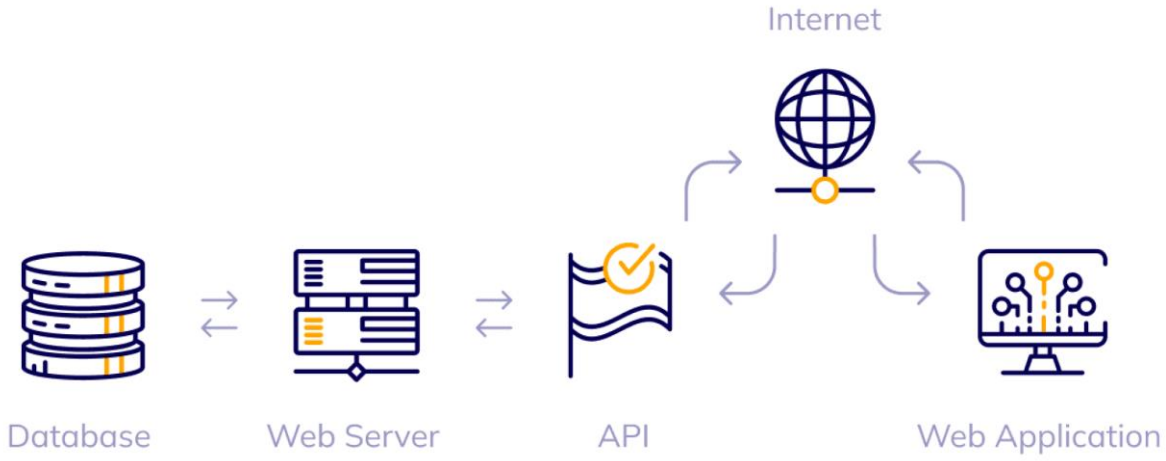

Рисунок 1.4.1 – Принцип роботи API

Запит API – це виклик до API для доступу до даних або функцій. Клієнт здійснює виклик API і надсилає запит на сервер API, а сервер надсилає відповідь. Запит і відповідь використовують певний формат і структуру та передаються за допомогою певного протоколу (наприклад, HTTP).

Щоб здійснити API запит, необхідно мати доступ до документації API. Ця документація містить інформацію про можливості API, структуру запитів і відповідей, а також будь-які вимоги до автентифікації чи авторизації.

API – це інструмент, який часто використовуються майже в усіх цифрових взаємодіях в сучасному світі, наприклад, під час використання мобільних додатків для телефонів, оплати через Інтернет або під час підключення до інших пристроїв або сервісів.

У випадку з однострінковими додатками, API виконують роль посередника між вебдодатком та сервером. В результаті, SPA значною мірою покладаються на API для отримання даних із сервера. Потім ці дані використовуються для оновлення вмісту вебсторінки без повного перезавантаження.

SPA можуть робити запити через API для різних цілей, наприклад для створення нових записів, оновлення наявної інформації або видалення даних. API реагує на ці запити, виконуючи необхідні операції на внутрішньому сервері. Це може передбачати взаємодію з базою даних для зберігання нових даних, зміни існуючих записів або видалення даних згідно з інструкціями SPA.

JavaScript на стороні клієнта має багато доступних API. Зазвичай їх можна поділити на дві категорії[12]:

- Браузерні API – вже вбудовані в веббраузер та забезпечують глибший доступ до браузера і взаємодію з його середовищем. Прикладами браузерних API є DOM(об'єктна модель документа), fetch API, web storage API та інші.
- Сторонні API – не вбудовані в браузер, вони розроблені зовнішніми організаціями та дозволяють користуватися сервісами цих організацій у власній програмі. Існує велика кількість сторонніх API, популярними прикладами є Twitter API, Google Maps API та Facebook API.

SPA є доволі гнучкими в своєму підході до серверних технологій, які вони можуть використовувати, і, також, до типу API, який вони використовують. Розглянемо кілька поширених типів API, які використовуються в розробці SPA.

RESTful API(REST – Representational State Transfer):

- В RESTful API запити надсилаються через браузер, а деталі кожного запиту вбудовано в URL-адресу.
- REST передбачає наявність клієнт-серверної архітектури.
- Системи RESTful підтримують обмін даними в різних форматах, таких як простий текст, HTML, YAML, XML і JSON.
- Ці API використовують HTTP методи для роботи з ресурсами: GET, PUT, HEAD, POST, PATCH, CONNECT, TRACE, OPTIONS і DELETE.

RESTful API популярні завдяки своїй простоті, гнучкості та широкій підтримці інструментів в різних мовах програмування.

GraphQL API:

- GraphQL – це мова запитів для API, яка дозволяє відразу вказати лише потрібні дані в одному запиті.
- Запитуючи лише необхідні дані, GraphQL зменшує обсяг даних, що передаються між вебсайтом та сервером.

gRPC API:

- gRPC — це універсальна структура API із відкритим кодом, яка також класифікується як RPC(Remote Procedure Calls, з англ. “виклики віддалених процедур”).
- За допомогою gRPC клієнт може безпосередньо викликати метод, функцію чи процедуру в іншій програмі у мережі.

WebSocket API:

- WebSocket API дозволяє відкривати двосторонній інтерактивний зв’язок між клієнтом та сервером[13].
- І клієнт, і сервер можуть надсилати і отримувати дані через таке з’єднання. Ці дані можуть мати різні формати, наприклад текстовий, JSON або двійкові дані. З’єднання залишається відкритим, доки одна сторона (клієнт або сервер) не закриє його.

API контролює доступ до своїх ресурсів через автентифікацію (визначення особи) і авторизацію (визначення, до чого дозволено доступ). Головною формою автентифікації, яка використовується з API, є ключ API. Ключ API – це ідентифікатор, вбудований у API запити, який визначає, чи ми маємо доступ до сервера.

Існують різні типи ключів API. Відкритий ключ API – це той, який може використовувати кожен без додаткової ідентифікації. Відкриті ключі часто надають обмежений доступ до ресурсів сервера. Щоб отримати доступ до повного вмісту ресурсів потрібно отримати зареєстрований ключ API. Тип ключа зазвичай визначає кількість запитів, які можна зробити до сервера протягом певного проміжку часу.

1.5. Інструменти для розробки односторінкових застосунків

Односторінкові програми покладаються на рендеринг на стороні клієнта, тому під час розробки SPA основна увага буде зосереджена на фронтенд розробці. Щоб створити односторінковий застосунок потрібно працювати з JavaScript.

JavaScript відіграє фундаментальну роль у розробці SPA. Завдяки своїй взаємодії з DOM, він може динамічно додавати, видаляти та змінювати елементи HTML сторінки. JS відстежує взаємодію користувача з елементами інтерфейсу (наприклад, натискання кнопок, надсилання форм) і виконує певний код у відповідь – це забезпечує інтерактивність і оперативність у SPA. Крім цього JS може обробляти деяку логіку програми на стороні клієнта без постійного зв'язку з сервером.[16] Можна використовувати чистий JavaScript або звернутися до JS фреймворків, як Vue.js, Angular чи React.js. Кожен JavaScript фреймворк має свої сильні та слабкі сторони.

React.js став одним із найпопулярніших інструментів для SPA. React спеціально розроблений для створення інтерактивних користувацьких інтерфейсів[21]. Він відмінно підходить для створення інтерактивних і динамічних інтерфейсів. З його допомогою можна створювати компоненти, які легко масштабуються, перевикористовуються та оновлюються динамічно без необхідності перезавантаження сторінки. React використовує JSX, додаткове синтаксичне розширення для JavaScript, яке дозволяє писати HTML-подібний код у JavaScript. Це полегшує написання компонентів, які поєднують структуру інтерфейсу користувача з логікою JS. Вебсайти створені на основі React.js можна перетворити на високопродуктивні мобільні вебдодатки, які виглядають точно так само, як нативні додатки.

Angular – популярний фреймворк, який в основному використовується для створення сучасних вебдодатків, включаючи SPA. Він володіє потужною

системою компонентів, зручними інструментами для маршрутизації та обробки даних. Angular був створений на основі Typescript, завдяки чому має більш чітку типізацію та структуру коду. Ще однією особливістю є його HTTP сервіси, які надають інструменти для асинхронних викликів API і отримання даних. Він також пропонує інструменти для тестування та налагодження додатків.

Vue.js. – простий та гнучкий JavaScript-фреймворк, який є чудовим вибором для створення SPA. Vue.js пропонує простий у використанні API для розробки компонентів та керування станом та маршрутизації. Vue.js підходить для створення додатків, які містять багато динамічних елементів. Він також має чудовий баланс між продуктивністю і функціональністю.

Ember.js – ще один популярний варіант для створення односторінкових додатків. Ember.js надає великій набір інструментів для розробки, наприклад інспектор стану додатка та засоби для тестування. Він пропонує надійний спосіб керування даними з використанням Ember Data, що спрощує взаємодію з API та базами даних. Ember.js також підтримує автоматичне зв'язування даних (data-binding).

Односторінкова програма не вимагає обов’язкової розробки серверної частини та використання баз даних. Такий застосунок може використовувати сховище на стороні клієнта для зберігання даних користувача та сторонні API для отримання динамічних даних(рис 1.5.1). Однак такий підхід не завжди працює, тому можливість створення серверної частини і використання бази даних не відкидається. У таких випадках поширеним є використання Node.js для розробки серверної частини і надається перевага реляційним базам даних (наприклад, PostgreSQL).

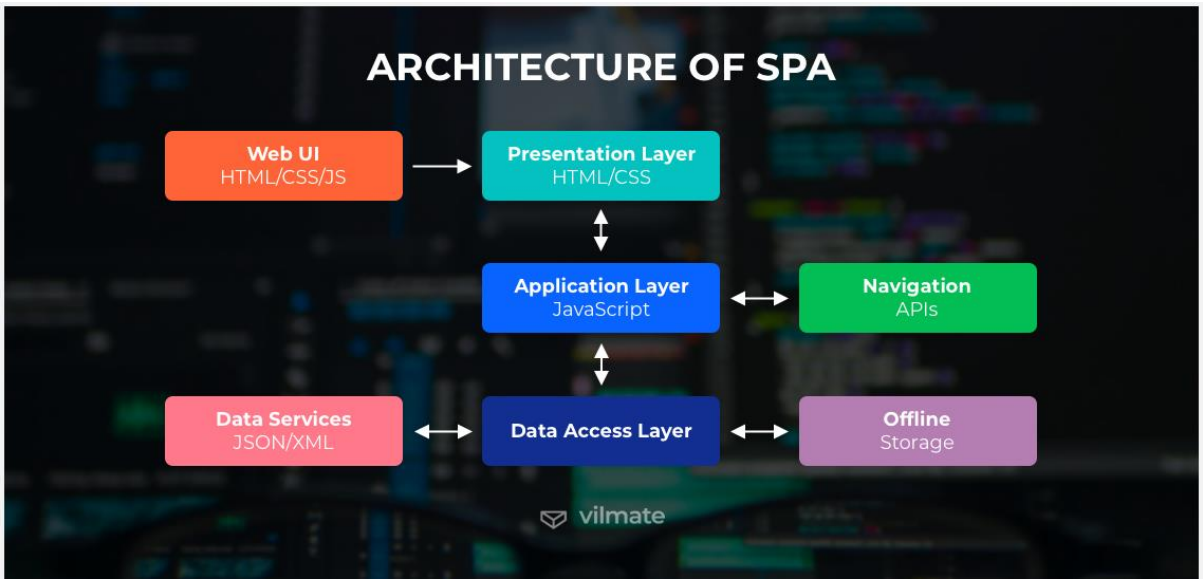

Рисунок 1.5.1 – Архітектура SPA

Як вказано вище, для ефективної роботи сайту SPA часто покладаються на зовнішні API, що дозволяють робити AJAX запити до сервера. Браузери надають API під назвою XMLHttpRequest (або XHR), який забезпечує саме цю функціональність, але він досить низькорівневий, тому краще використовувати бібліотеку-обгортку. На додаток до цього бібліотеки зазвичай додають корисні функції, як-от виконання запитів на основі Promise.

Існує безліч варіантів на вибір: jQuery.ajax(), axios, fetch, $http в Angular тощо. Як завжди, кожен має свої плюси та мінуси, але врешті-решт усі вони виконують свою роботу. Останнім часом найбільш використовуваними є axios і fetch.

Сам AJAX працює шляхом обміну даними з сервером, дозволяючи вебдодаткам надсилати та отримувати дані асинхронно, не втручаючись у процес відображення даних та поведінку існуючої сторінки. SPA використовує AJAX для отримання інформації з сервера, надсилання форм, оновлення частин сторінки та інших взаємодій[15]. AJAX використовує комбінацію технологій, зокрема: вбудований у браузер об’єкт XMLHttpRequest, JavaScript, HTML DOM, а також HTML/CSS для представлення інформації.

Для більш комплексних SPA застосунків можна використовувати Redux, JavaScript бібліотеку для керування станом програми, що працює з великою кількістю даних. Це забезпечує простий та прозорий спосіб керування потоком даних між різними компонентами інтерфейсу користувача.

Redux працює на основі концепції однорівневого стану, де усі дані додатка зберігаються в єдиному об'єкті[20]. Коли до стану надходить дія(action), його обробляють редуктори (reducers). Редуктори беруть об'єкт стану компонента, що змінився, та дію, і на їх основі генерують новий об'єкт стану. Він отримує, зберігає і за потреби передає одним компонентам дані інших. До нього можна звернутися, щоб дізнатися, чи кнопка натиснута, яке значення зараз у змінної, чи вибрана умова. Це зручніше та простіше, ніж звертатися безпосередньо від компонента.

1.6. Огляд та аналіз аналогічних програмних розробок

Перш ніж перейти до другого розділу, розглянемо інші програмні розробки, які створили успішні продукти в рамках архітектури SPA.

SPA – один з найпопулярніших способів створення вебдодатків, він типічно використовується для розробки наступних застосунків:

- Вебдодатки: SPA відмінно підходять для створення складних вебдодатків, де потрібна максимальна взаємодія користувача з інтерфейсом.
- Інтернет магазини: SPA дозволяють оновлювати інформацію про товари та кошик у режимі реального часу, без перезавантаження сторінки.
- Соціальні мережі: інтерактивність є важливим компонентом у соціальних мережах, де користувачі знайомляться, обмінюються повідомленнями та діляться контентом. Використання SPA дозволяє відразу бачити нові повідомлення, коментарі або оновлення без необхідності перезавантаження сторінки.

Розглянемо існуючі приклади застосунків у форматі SPA.

Gmail(https://mail.google.com/?hl=en) – безкоштовна служба електронної пошти від компанії Google.

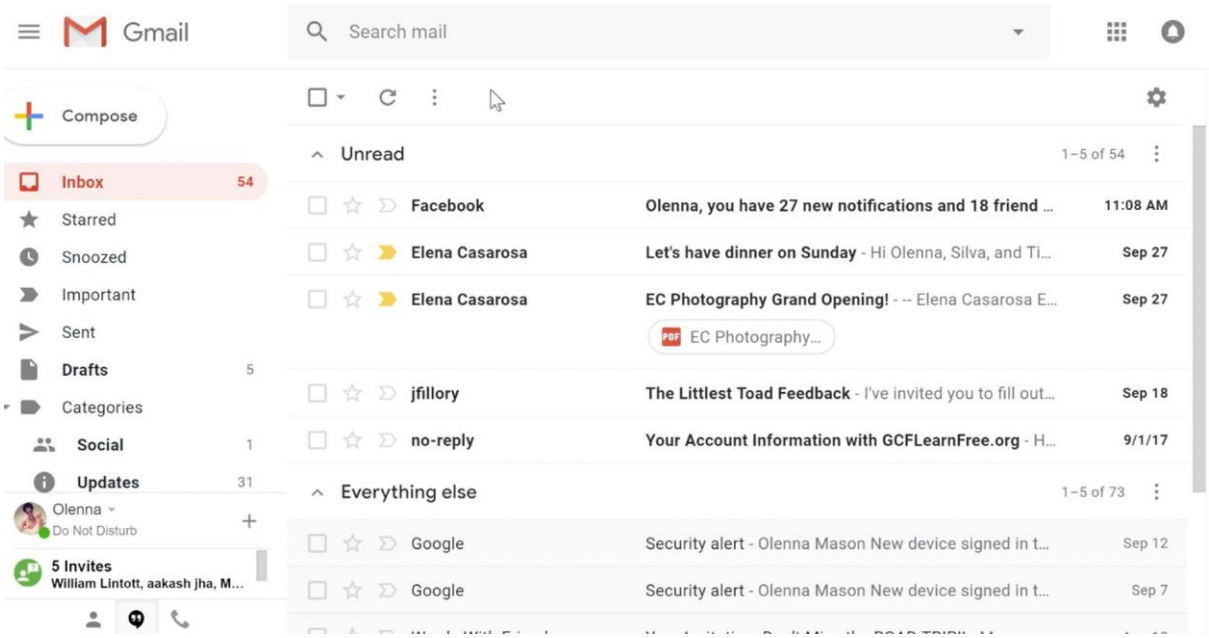

Рисунок 1.6.1 – Безкоштовна служба електронної пошти Gmail

Поштова скринька має кілька основних секцій, які сортують повідомлення за різними пріоритетами. Їх можна відкрити з допомогою навігаційного меню: вхідні, із зірочкою, відкладені, важливі, відправлені, чернетки, заплановані, вся почта, спам, кошик. В Gmail вбудована пошукова система Google, яка дозволяє шукати конкретні розмови серед своїх листів. Також є функція створення нового повідомлення, яка відкриває в правому нижньому кутку вікно з конструктором нового листа.

Користувачам не потрібно оновлювати сторінку своєї поштової скриньки, щоб отримувати нові повідомлення електронної пошти. SPA архітектура, яка керує Gmail автоматично представляє новий вміст скриньки, щойно він надходить із сервера.

Вебсайт компанії Formation Stone Surfaces (https://formationstone.com/).

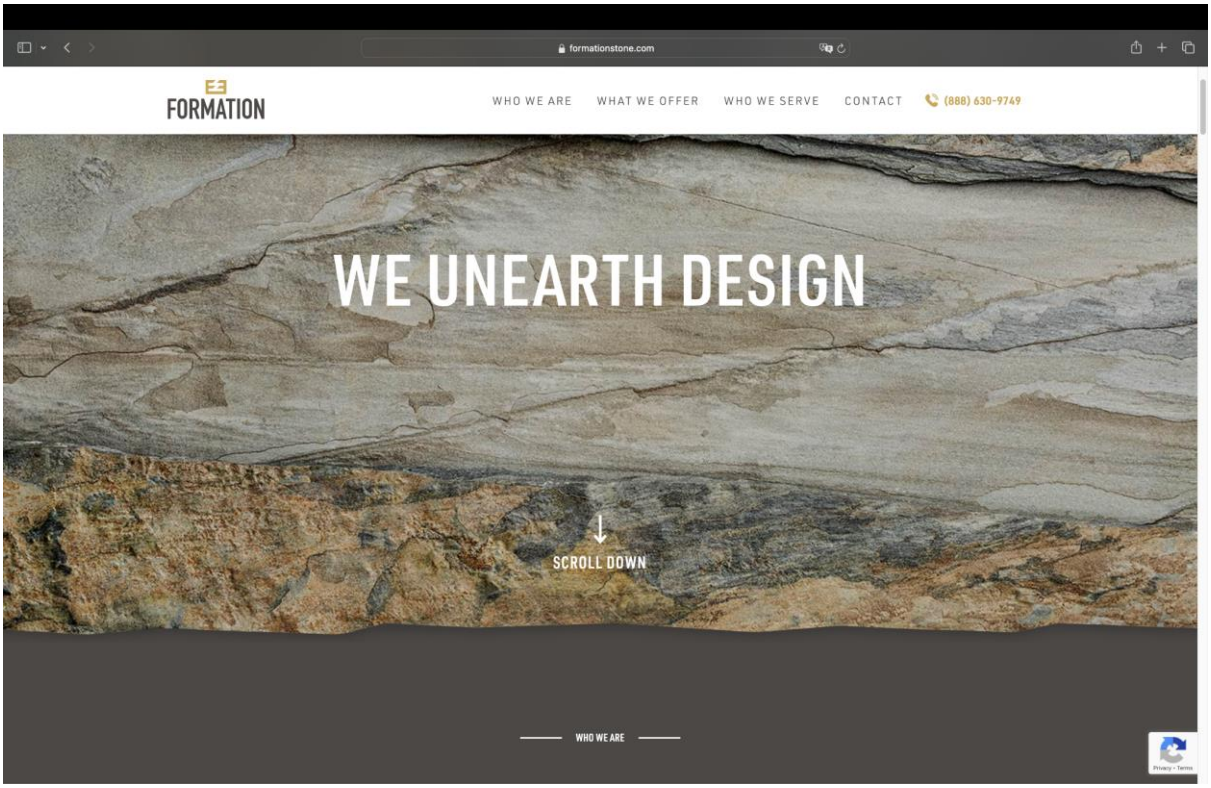

Рисунок 1.6.2 – Сайт компанії Formation Stone Surfaces

Першою є вступна частина – частина із заголовком. На даному сайті можна виділити кілька основних секцій. Всі вони розміщені в шапці сайту та мають відповідні посилання.

“WHO WE ARE” – невелика секція для ознайомлення з компанією Formation. Візитка сайту.

“WHAT WE OFFER” – інтерактивний каталог товарів та сервісів, які пропонує компаніє.

“WHO WE SERVE” – секція, де можна подивитися відгуки від клієнтів Formation. Відгуки зображені у вигляді каруселі, яку можна переключати використовуючи слайдер.

“CONTACT” – заключна частина. Тут знаходяться форма з надсилання електронного листа для зв’язку з компанією. Також розділ з чотирма офісами компанії та їх адресами і контактами. Додатково до адреси, використовується

інтерактивний елемент від Google Maps, що показує розташування офісів на карті.

Портал з інформацією про персонажів і комікси всесвіту MARVEL (https://alexshatokhin.github.io/MarvelReact/).

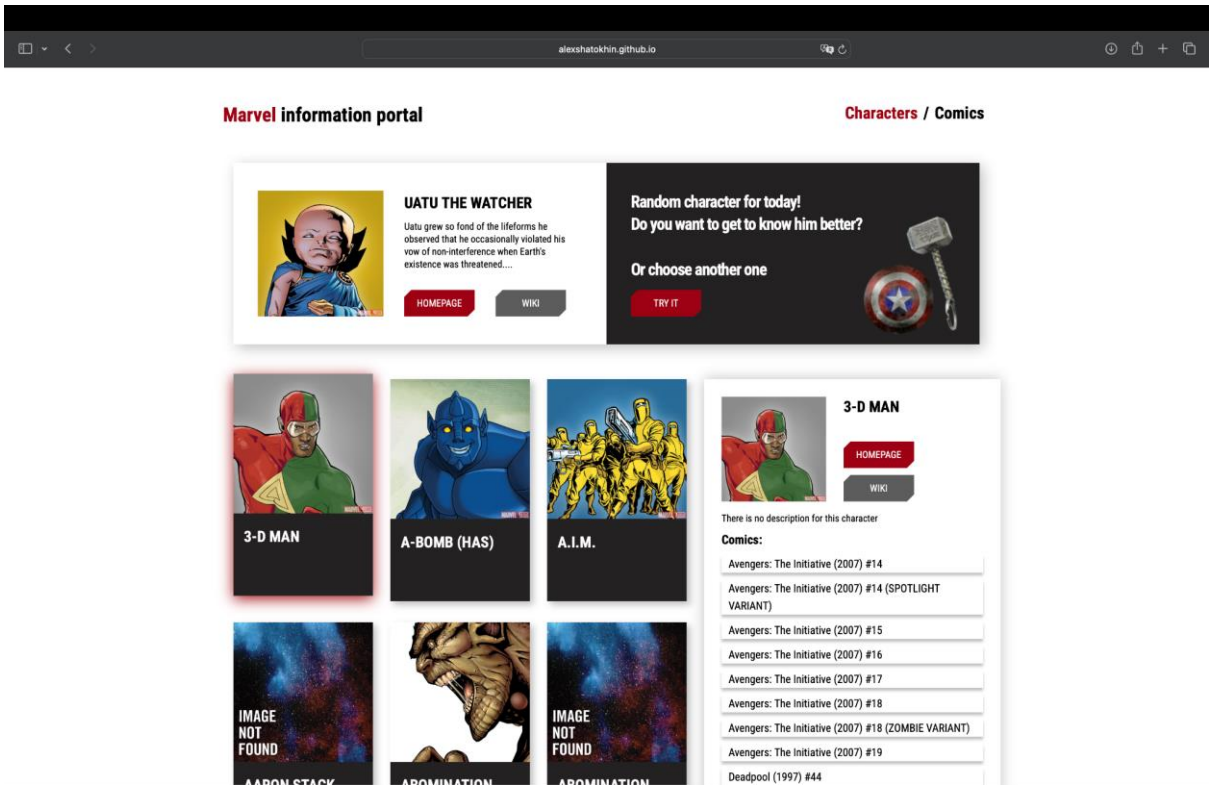

Рисунок 1.6.3 – Портал про персонажів і комікси всесвіту MARVEL

Даний застосунок має дві секції: розділ з персонажами і розділ з коміксами.

У розділі з персонажами MARVEL можна ознайомитись з персонажем, вибраним із наданої бази даних Marvel або випадково за допомогою кнопки “TRY IT”. Після вибору персонажа в правій частині сторінки буде відображатися його короткий опис. Користувач може відкрити більше персонажів після натиску на кнопку “LOAD MORE”. Також вебсайт має функцію пошуку персонажа по імені.

Наступний розділ присвячений коміксам MARVEL. На сторінці відображається перелік різних коміксів, який, ідентично до минулого розділу,

можна розширити за допомогою кнопки “LOAD MORE”. Після вибору комікса користувач переходить на сторінку з більш детальною інформацією про нього.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ОДНОСТОРІНКОВОГО САЙТУ З ВИКОРИСТАННЯМ API.

2.1. Постановка задачі, призначення та вимоги до програмного продукту

Основним завданням даної дипломної роботи є створення прототипу односторінкового вебзастосунку, використовуючи браузерні та сторонні API. Даний сайт буде вебдодатком для показу прогнозу погоди, який буде використовувати різні технології для гарантування практичного користувацького досвіду.

Головним призначенням веброзробки є демонстрація переваг односторінкових застосунків та функціональності, яку їм надають API.

Кінцевий продукт має забезпечувати такі функції:

- динамічний користувацький інтерфейс;
- пошукова система, яка дозволить користувачам знаходити погоду за назвою міста;
- API інтеграція;
- використання реальних даних про погодні умови з стороннього сервера;
- використання сервісів геолокації для визначення розташування користувача;
- забезпечення хостингу сайту з безпечним HTTPS з'єднанням;

Крім цього застосунок повинен мати простий та зрозумілий дизайн з зручною навігацією.

2.2. Вибір моделі розробки програмного засобу

Створення програмного забезпечення завжди передбачає певний структурований цикл роботи, слідуючи якому команда розробників може

досягти бажаного результату. Для організації цього циклу використовують різні моделі розробки такі як – каскадна модель, спіральна модель, ітеративна модель, модель хаоса та інші. В процесі написання дипломної роботи було прийнято рішення використовувати ітеративну модель розробки.

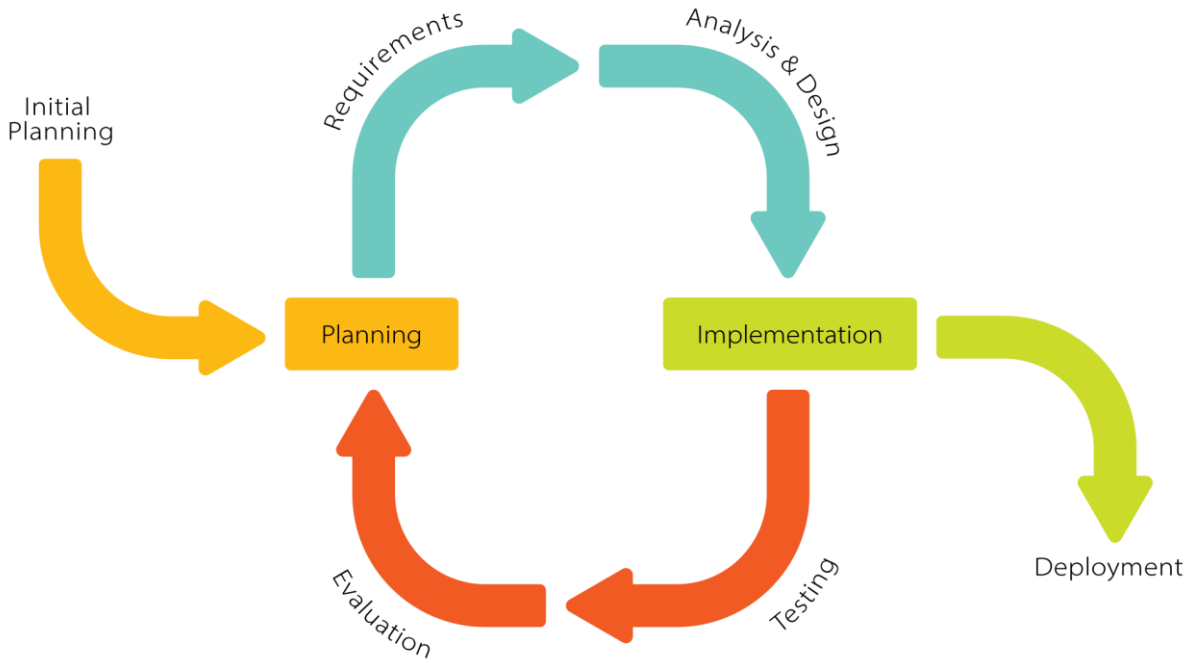

Рисунок 2.2.2 – Ітеративна модель розробки

При використанні ітеративної моделі розробки, весь процес розбитий на певну кількість етапів(ітерацій). Після кожного етапу команда отримує нову версію програмного продукту, яка проходить тестування і потім за потреби зазнає змін та вдосконалюється на наступній ітерації. Цей цикл продовжується до моменту, коли замовник та команда досягли бажаного результату, і не бачать потреби продовжувати роботу над продуктом[18].

Розробку за ітеративною моделлю можна поділити на 6 окремих стадій:

1. Збір вимог та їх аналіз, планування: на цьому етапі визначаються функціональні та нефункціональні вимоги до продукту, встановлюються цілі та плани роботи для команди.
2. Дизайн: ця стадія передбачає створення дизайну продукту.

3. Реалізалізація: головними процесами під час цього етапу є кодування функцій та компонентів для програми.
4. Тестування: після завершення реалізації типічно проводяться різні види тестування, такі як модульне, інтеграційне та функціональне тестування.
5. Огляд: на цьому етапі проводиться огляд роботи та валідності розробленого продукту, якщо буде знайдено будь-яку помилку або буде вирішено, що продукт потребує додаткових функцій, процес повертається до стадії планування.
6. Реалізація: програмне забезпечення розгортається в робочому середовищі.

Основними перевагами ітераційної моделі розробки є:

– тестування та налагодження в момент ітерації проходить швидше;
– можливість легко пристосовуватися до мінливих вимог для фінальної розробки;
– проблеми та ризики ідентифікуються і вирішуються під час ітерацій;

З другого боку така модель не підходить для проектів, які мають чіткі терміни і бюджет, так як розробка проходить в потоковому форматі і передбачити часові рамки та об'єм потрібних ресурсів буває дуже важко, а інколи просто неможливо.

2.3. Загальний опис проєкту

2.3.1. Вимоги до дизайну сайту

Дизайн односторінкових додатків має допомагати зберегти баланс між зручністю використання та масштабністю. Усі ключові компоненти та елементи меню мають знаходитись на початковій сторінці. Важливо переконатися, що користувачі не витрачатимуть час на прокручування до потрібного компонента програми чи інформації. Чим швидше користувачі зрозуміють, як виконати

певну дію, тим більше позитивного досвіду вони отримають під час використання SPA-додатка.

Всі інтерактивні компоненти застосунка повинні мати відповідні лейбли для чіткого розуміння їх призначення і зручної навігації.

Сайт повинен мати стабільну стилістику, яка буде естетичною, приємною та відповідною тематиці сайту.

Візуали. Знімок для фону вебсайту було взято з Unsplash(сайт, що пропонує безкоштовні фотографії для персонального чи комерційного використання). Також застосунок використовує піктограми погодніх умов від API сервісу OpenWeather.

Типографія. Сайт буде використовувати такі шрифти: Albert sans, “sans serif” з висотою кеглю 50px, 48px, 38px, 22px, 20px і 27px. Для реалізації цих шрифтів використовується безкоштовна бібліотека Google Fonts.

2.3.2. Вимоги до структури сайту

На ранніх етапах роботи було розроблено макет основного вікна для інтерфейсу сайту(рис 2.3.2.1). Для виготовлення макету використовувався застосунок InVision.

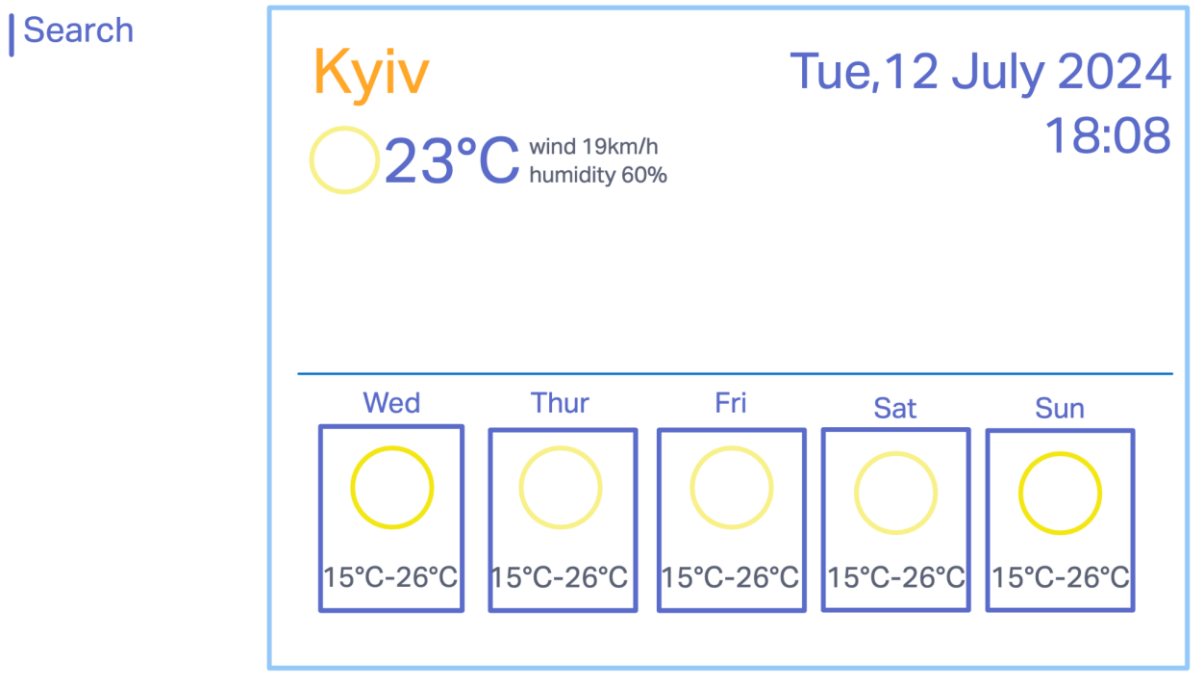

Рисунок 2.3.2.1 – Макет основного вікна сайту

Основним компонентом сторінки є головне вікно, яке відповідає за зображення прогнозу погоди. На верхній частині цього вікна буде відображатися інформація про погоду в поточний день. З лівого боку можна побачити назву міста(локацію), піктограму, температуру(C°), швидкість вітру(в км/год) і вологість(%), а з правого боку – поточний час.

В нижній частині знаходиться секція з іконками прогнозу погоди на п’ять днів вперед. Кожна іконка підписана відповідним днем тижня, який вона описує, та містить мінімальну і максимальну температуру на день і піктограму, яка описує погодні умови.

Також за межами головного вікна розташована панель пошуку і кнопка “Current Location”.

Нижче від основного вікна розташовані ще дві панелі – “Description” та “Map”. На першій можна побачити короткий опис погоди на день, включаючи як відчувається температура, дані про стан неба, мінімальна і максимальна температура на день та хмарність. На другій половині зображено інтерактивну мапу з позначкою в місті, яке задав користувач.

Для розробки інтерфейсу(рис.2.3.2.2) використовувалися мови програмування HTML5/CSS3 з використання бібліотеки Bootstrap.

Цей сайт використовуватиме архітектуру односторінкового вебзастосунка (SPA), створену за допомогою мови програмування JavaScript. Такий підхід забезпечує динамічні оновлення та плавну роботу користувача без повного перезавантаження сторінки.

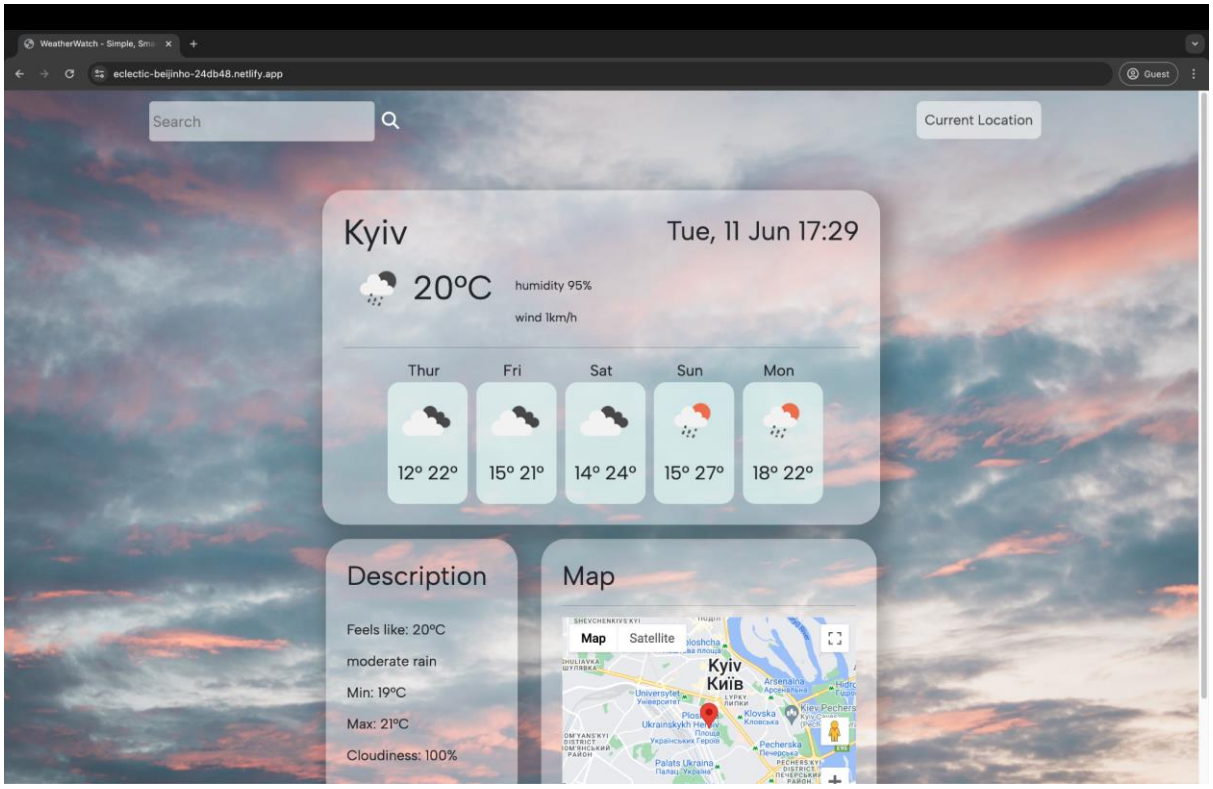

Рисунок 2.3.2.2 – Інтерфейс вебсайту WeatherWatch

2.3.3. Вимоги до функціоналу сайту

Задля того, щоб розпочати роботу з застосунком і отримати погодні дані, користувач може використати пошукове поле або кнопку “Current Location”, за умови надання доступу до своїх геоданих. Для роботи цих двох опцій потрібно забезпечити API інтеграцію.

OpenWeatherMap API буде використовуватися для отримання даних про погоду. Коли користувач вводить назву міста і натискає “ENTER”, щоб побачити відповідний прогноз, цей запит буде оброблятися певним JavaScript кодом. Цей JavaScript код буде виконувати API запит, специфічний для OpenWeatherMap API, для цього ми будемо використовувати бібліотеку для виконання HTTP запитів Axios. Цей запит має обов’язково включати локацію(назву міста), для якої зробили запит, та унікальний ключ API.

Після цього сервер повертає відповідь на наш запит. В таких випадках сервер зазвичай надсилає відповідь в формі JSON-об’єкта(рис.2.3.3.1). Тоді код

JavaScript має опрацювати дані об’єкта і оновити інтерфейс сайту отриманою інформацією про погоду. Це може включати динамічну зміну текстового вмісту, відображення піктограм погодних умов та створення іконок прогнозу.

Рисунок 2.3.3.1 – Приклад JSON-об’єкта від OpenWeatherMap API

2.4. Обґрунтування вибору інструментальних засобів розробки

При виборі середовища для розробки кінцевого програмного продукту було прийнято рішення використовувати Visual Studio Code. Visual Studio Code є популярним вибором серед багатьох розробників через ряд своїх переваг. VS Code має такі сильні сторони:

– кросплатформеність – VS Code працює на Windows, macOS і Linux;
– наявність ефективного редактора коду з вбудованим підсвічуванням синтаксису, зіставленням дужок, засобами навігації, автоматичним відступом і доповнення коду;
– надає доступ до багатьох засобів налаштування, включаючи теми інтерфейсу та інші інструменти, що дозволяють адаптувати функціональні можливості для своїх конкретних потреб;

– велика бібліотека розширень, які дозволяють додавати нові мови програмування, інструменти та засоби для підтримки процесу розробки;
– зручний та інтуїтивно простий у розумінні інтерфейс.

Також для автоматизації форматування коду використовувалося розширення Prettier. Це розширення забезпечує узгоджений стиль форматування за власними правилами. Воно працює з різними мовами програмування включаючи JavaScript, TypeScript, CSS, HTML та інші.

Під час розробки було використано розширення Live Server, що дозволяє запускати локальний сервер для розробки безпосередньо з середовища програмування. Після збереження HTML, CSS або JavaScript файлів він автоматично оновлює вебсторінку в браузері, відображаючі внесені в файли зміни.

Для контролю версій використовувався Git. Git – це розподілена система контролю версій, яка дозволяє розробникам співпрацювати над одним проектом і відстежувати зміни у файлах. Він відомий своєю швидкістю, простим дизайном, підтримкою нелінійної розробки, повною децентралізацією та ефективністю роботи з великими проектами. Git — це потужна й універсальна система контролю версій, яка пропонує численні переваги, зокрема відкритість, легкість в використанні, швидкість, надійність для резервного копіювання даних, а також має прості й ефективні можливості розгалуження.

Також в процесі розробки використовувався GitHub. GitHub – це платформа для хостингу та спільної роботи над кодом. Вона дозволяє розробникам зберігати, керувати та спільно працювати над проектами з відкритим вихідним кодом. GitHub пропонує широкий спектр інструментів для управління проектами, включаючи пул-реквести, репозиторії, завдання, коментарі та вікі, а також інструменти для спільної роботи, такі як дошки завдань та чати. Платформа дозволяє легко ділитися проектами з іншими розробниками та отримувати зворотний зв'язок та підтримку у розробці.

Для організації хостингу вебсайту було прийнято рішення використовувати Netlify. Netlify – це хмарна платформа, яка надає розробникам спосіб створювати, розгортати та керувати вебдодатками. Велика кількість функцій платформи приваблює розробників усіх рівнів досвіду: хостинг для статичних та динамічних вебсайтів, безсерверні функції, які виконуються без розгортання окремого сервера, а також автоматичне розгортання.

Netlify має такі сильні сторони[23]:

- простота використання;
- висока продуктивність – можливість обробляти мільйони запитів за секунду без шкоди для продуктивності;
- безпека – використання SSL/TLS для шифрування трафіку, а також сканування на наявність вразливостей та захист від DDoS-атак.

Для програмування логіки вебсайту використовується JavaScript. JavaScript – це динамічна, об'єктно-орієнтована прототипна мова програмування. Вона є дуже різносторонньою і використовується в різних сферах розробки, але найбільш суттєвим є її вплив на веб розробку. Майже на кожному сучасному вебсайті використовується код, написаний на JS. Крім цього, JavaScript забезпечує повну інтеграцію з HTML/CSS.

Ще однією особливістю JavaScript є те, що вона відноситься до інтерпретованих мов програмування. Тобто це означає, що її код не потребує компіляції. Його пишуть і одразу передають програмі-інтерпретатору, яка його виконує. Також JS не має чіткої типізації, натомість використовується динамічна типізація – в змінній можна зберігати будь-які типи даних.

В браузері JavaSript може[17]:

- додавати та видаляти наявний HTML-код на сторінці, змінювати існуючий вміст сайту, стилі;
- реагувати на дії та запити користувача, такі як натискання миші, рух курсора, натискання клавіш;

– використовуючи технології AJAX і COMET, надсилати запити до віддалених серверів і скачувати та надсилати файли;
– отримувати та надсилати куки, ставити запитання користувачам, показувати повідомлення;
– запам’ятовувати дані на стороні клієнта (“local storage”), які будуть доступні в майбутніх сесіях на цьому вебсайті.

Щоб організувати зв’язок з стороннім сервером використовується HTTP-клієнт Axios. Це досить популярна бібліотека, яка дозволяє робити HTTP-запити та отримувати дані з сервера. Axios надає стислий та інтуїтивно зрозумілий API для створення HTTP-запитів. Для запиту потрібно вказати URL-адресу, метод (GET, POST, PUT, тощо), а також додаткові дані або параметри конфігурації і все це в зрозумілому читабельному форматі.

Axios дозволяє перехоплювати та змінювати запити чи відповіді перед їх надсиланням або обробкою, що може бути корисним для таких завдань, як автентифікація та зберігання запитів/відповідей. Додатково axios має вбудовану підтримку обробки помилок і надає докладні повідомлення про помилки.

Axios використовує Promise – основну концепцію JavaScript для обробки асинхронних операцій. Promise дозволяють визначити, що відбувається після успішного виконання запиту (завершеного запиту) або виявлення помилки (відхиленого запиту)[11].

Як було згадано раніше, ми використовуємо OpenWeatherMap API для отримання даних про погоду. Цей API надає поточну інформацію про погоду, прогнози та історичні дані для місць у всьому світі. Він використовує підхід RESTful, що дозволяє робити запити з різними HTTP методами та параметрами для отримання даних. У відповідь він може надсилати дані у форматах JSON, XML і HTML.

В даному проекті буде використовуватися безкоштовний план One Call API 3.0. Він дозволяє робити 2000 API запитів на день. На офіційному вебсайті

OpenWeatherMap можна знайти документацію з інструкціями використання даного API та отримати персональний API-ключ для свого проекта.

```
https://api.openweathermap.org/data/3.0/onecall?lat={lat}&lon=
{lon}&exclude={part}&appid={API key}
```

Рисунок 2.4.1 – Приклад API запиту для OpenWeatherMap API

Для реалізації інтерактивної карти використовується Google Maps API. В даний час платформа Google Maps надає велику кількість API для різних аспектів своїх послуг. Деякі з доступних для використання API включають Static API Maps для виконання простих вставок з Google Maps, Maps JavaScript API для створення інтерактивних і настроюваних карт, Places API для доступу до даних про точки інтересу та Directions API для надання маршрутів розташування[26].

На даному сайті використовується Maps JavaScript API. Він пропонує широкі можливості налаштування, щоб адаптувати вигляд карти до дизайну вебсайту: можна налаштувати кольори, мітки та інші візуальні елементи.

На сайті Google Map Platform в відкритому доступі є документація з вказівками до впровадження даного API та інструкція для отримання API-ключа. Щоб додати карту до свого проєкта потрібно обв’язково додати скрипт Maps JavaScript API.

```
<script>
  (g=>{var h,a,k,p="The Google Maps JavaScript API",c="google",l="importLibrary",q="
    key: "AIzaSyDUrkbxWwN4nbIYxb9m6HswHBFfgvQsCfw",
    v: "weekly",
    // Use the 'v' parameter to indicate the version to use (weekly, beta, alpha, et
    // Add other bootstrap parameters as needed, using camel case.
  });
</script>
```

Рисунок 2.4.2 – Скрипт для впровадження Maps JavaScript API

2.5. Особливості програмної реалізації

Для того, щоб розпочати роботу з застосунком і отримати погодні дані, ми в першу чергу використовуємо пошукове поле, тому розглянемо головні процеси, які зумовлюють його роботу.

Перша функція, яка отримує дані з поля це searchButton()(рис. 2.5.1). Під час роботи цієї функції програма перевіряє введені дані, після чого у випадку їх валідності передає їх далі у функцію searchCity()(рис. 2.5.2).

```
function searchButton(event) {
  event.preventDefault();
  let searchInput = document.querySelector("#search-text-input");
  console.log(searchInput.value);
  let newCity = document.querySelector("#real-location");
  if (searchInput.value) {
    searchCity(searchInput.value);
  } else {
    newCity.innnerHTML = null;
    alert("Please type a city");
  }
  searchInput.value = "";
}
```

Рисунок 2.5.1 – Функція searchButton()

```
function searchCity(city) {
  let apiKey = "7b3a77a5c1a8ebaa302785b7cb6888c7";
  let apiUrl = `https://api.openweathermap.org/data/2.5/weather?q=${city}&units=metric&appid=7b3a77a5c1a8ebaa302785b7cb6888c7`;
  axios.get(apiUrl).then(showWeather);
}
```

Рисунок 2.5.2 – Функція searchCity()

У функції searchCity ми звертаємося до OpenWeatherMap API. Для того, щоб отримати доступ до погодніх даних від OpenWeather використовуємо бібліотеку Axios. В результаті цього запиту, ми отримаємо JSON-об’єкт, який містить в собі погодні дані про місто, яке задає користувач. Надалі ми зможемо застосовувати їх на сайті.

У функції showWeather()(рис. 2.5.3) ми використовуємо погодні дані, щоб оновити відповідні їм секції на сторінці. Таким чином на сторінці з'являться нова температура, вологість, швидкість вітру, піктограма погодніх умов та поточний час.

```
function showWeather(response) {
  console.log(response.data);
  let temperature = Math.round(response.data.main.temp);
  let currentDegree = document.querySelector("#weather-number");
  currentDegree.innerHTML = temperature;
  let realLocation = document.querySelector("#real-location");
  let humidity = document.querySelector("#humidity");
  humidity.innerHTML = `humidity ${response.data.main.humidity}%`;
  realLocation.innerHTML = response.data.name;
  let windSpeed = document.querySelector("#wind");
  windSpeed.innerHTML = `wind ${Math.round(response.data.wind.speed)}km/h`;
  let fullDate = document.querySelector("#real-date");
  fullDate.innerHTML = formatDate(response.data.dt * 1000);
  let currentIcon = document.querySelector("#icon");
  currentIcon.setAttribute(
    "src",
    `http://openweathermap.org/img/wn/${response.data.weather[0].icon}@2x.png`
  );
  currentIcon.setAttribute("alt", `${response.data.weather[0].description}`);

  searchForecast(response.data.coord);
  initMap(response.data.coord);
  displayDescription(response.data)
}
```

Рисунок 2.5.3 – Функція showWeather()

На даному етапі важливо правильно вказати шлях до потрібних даних. Наприклад, щоб отримати температуру потрібно вказати таку послідовність: response.data.main.temp. Для визначення правильного шляху можна використовувати документацію до API або орієнтуватися по відповідному JSON-об'єкту.

За схожим принципом ми оновлюємо дані в прогнозі на наступні п'ять днів використовуючи функцію displayForecast()(рис. 2.5.4). Ця функція

відповідає за створення п’яти іконок прогнозу. Кожна іконка буде показувати тільки мінімальну і максимальну температуру дня, а також відповідну піктограму погодніх умов. Щоб зобразити всі п’ять іконок використовується метод forEach().

```
function displayForecast(response) {
  let dailyForecast = response.data.daily;

  let forecast = document.querySelector("#dayly-forecast");
  let forecastHTML = `<div class = "row">
  <div class="col-1"></div>`;
  dailyForecast.forEach(function (forecastDay, index) {
    if ((index > 0) & (index < 6)) {
      forecastHTML =
        forecastHTML +
        `<div class="col-2">
                  <span class="week">${formatForecastDate(forecastDay.dt)}</span>
                  <div class="day-forecast">
                    <img
                      src="http://openweathermap.org/img/wn/${
                        forecastDay.weather[0].icon
                      }@2x.png"
                      alt="${forecastDay.weather[0].description}"
                    />
                    <span class="forecast-temp-min-max"> ${Math.round(
                      forecastDay.temp.min
                    )}° ${Math.round(forecastDay.temp.max)}°</span>
                  </div>
                </div>`;
    }
  });
  forecastHTML =
    forecastHTML +
    `<div class="col-1"></div>
                </div>`;
  forecast.innerHTML = forecastHTML;
}
```

Рисунок 2.5.4 – Функція displayForecast()

Щоб додати інформацію до панелі “Description” використовується функція displayDescription()(рис. 2.5.5). Вона отримує параметр data, в якому збережені погодні дані, та використовуючи їх динамічно оновлює інформацію панелі.

```
function displayDescription(data){
  console.log(data);
  let temp = Math.round(data.main.feels_like);
  let feelTemp = document.querySelector("#desc-temp");
  feelTemp.innerHTML = `Feels like: ${temp}°C`;

  let weatherDesc = document.querySelector("#desc-main");
  weatherDesc.innerHTML = `${data.weather[0].description}`;

  let minTemp = document.querySelector("#desc-min");
  minTemp.innerHTML = `Min: ${Math.round(data.main.temp_min)}°C`;

  let maxTemp = document.querySelector("#desc-max");
  maxTemp.innerHTML = `Max: ${Math.round(data.main.temp_max)}°C`;

  let curClo = document.querySelector("#desc-clo");
  curClo.innerHTML = `Cloudiness: ${data.clouds.all}%`;
}
```

Рисунок 2.5.5 – Функція displayDescription()

Додаємо інтерактивну мапу з функцією initMap()(рис. 2.5.6) використовуючи Google Maps API. Як параметр задаємо координати місцезнаходження(широту і довготу) і потім за допомогою API генеруємо карту та додаємо маркер(рис. 2.5.7).

```
async function initMap(coordinates) {
  let position = { lat: coordinates.lat, lng: coordinates.lon };
  const { Map } = await google.maps.importLibrary("maps");
  const { AdvancedMarkerElement } = await google.maps.importLibrary("marker");

  map = new Map(document.getElementById("map"), {
    zoom: 12,
    center: position,
    mapId: "DEMO_MAP_ID",
  });

  const marker = new AdvancedMarkerElement({
    map: map,
    position: position,
  });
}
```

Рисунок 2.5.6 – Функція initMap()

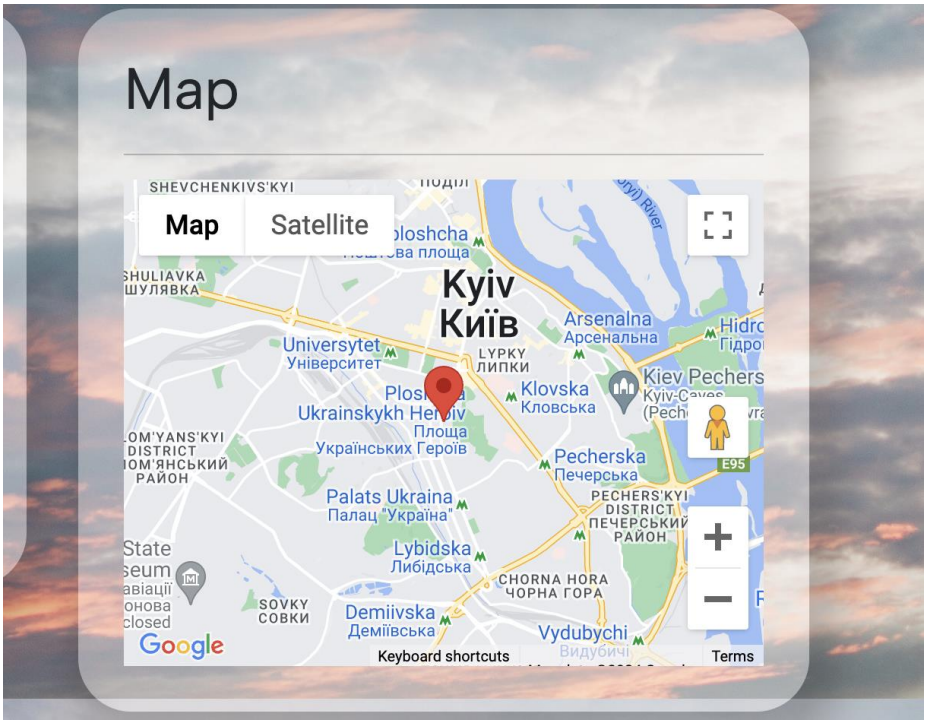

Рисунок 2.5.7 – Елемент Map на сайті WeatherWatch

Крім цього, є опція отримати прогноз за своїм місцезнаходженням за допомогою кнопки CurrentLocationButton(рис. 2.5.8). Це свою чергу викликає функцію getCurrentPosition(), яка використовує Navigation API для визначення точної геолокації користувача. Потім ці дані геолокації використовуються для звернення до OpenWeatherMap API. Даний API може використовувати не тільки назву міста, а й географічні координати для знаходження погодніх даних.

```
//current location button
function currentWeather(position) {
  console.log(position.coords.latitude);
  let apiUrl = `https://api.openweathermap.org/data/2.5/weather?lat=${position.coords.latitude}
  &lon=${position.coords.longitude}&units=metric&appid=7b3a77a5c1a8ebaa302785b7cb6888c7`;
  axios.get(apiUrl).then(showWeather);
}

function getCurrentPosition() {
  navigator.geolocation.getCurrentPosition(currentWeather);
}

let currentLocationButton = document.querySelector("#device-location");
currentLocationButton.addEventListener("click", getCurrentPosition);
```

Рисунок 2.5.8 – Кнопка Current Location

Також, слід зазначити, що в JavaScript немає автоматичного форматування часу, тому це робиться з використанням функцій formatDate()(рис.2.5.9) та formatForecastDate(рис. 2.5.10).

```
function formatForecastDate(timestamp) {
  let now = new Date(timestamp * 1000);
  let days = ["Sun", "Mon", "Tue", "Wed", "Thur", "Fri", "Sat"];
  let day = days[now.getDay()];
  return `${day}`;
}
```

Рисунок 2.5.9 – Функція formatForecastDate()

```
function formatDate(timestamp) {
  let now = new Date(timestamp);

  let date = now.getDate();
  let hours = now.getHours();
  if (hours < 10) {
    hours = `0${hours}`;
  }
  let minutes = now.getMinutes();
  if (minutes < 10) {
    minutes = `0${minutes}`;
  }
  let days = ["Sun", "Mon", "Tue", "Wed", "Thur", "Fri", "Sat"];
  let day = days[now.getDay()];
  let months = [
    "Jan",
    "Feb",
    "Mar",
    "Apr",
    "May",
    "Jun",
    "Jul",
    "Aug",
    "Sept",
    "Oct",
    "Nov",
    "Dec",
  ];
  let month = months[now.getMonth()];

  return `${day}, ${date} ${month} ${hours}:${minutes}`;
}
```

Рисунок 2.5.10 – Функція formatDate()

2.6. Організація тестування та налагодження програмного засобу

Головною метою тестування програмного засобу є перевірка на наявність дефектів чи помилок в роботі програми.

Під час роботи над даним програмним засобом було проведено такі види тестування:

– Тестування функціональності: потрібно було переконатися, що основні функції, як відображення прогнозу, знаходження геолокації, робота пошукового поля працюють належним чином; потрібно протестувати інтеграцію з обраним API;

– Тестування точності даних: для того, щоб переконатися, що програма отримує правильні дані про погоду, слід порівняти ці дані з прогнозами довірених джерел;

– Тестування продуктивності: порахувати час завантаження вебсайту в умовах різної якості інтернет-з’єднання;

– Тестування кросплатформеності: перевірити функціональність сайту на різних браузерах.

Під час тестування використовувався інструмент PageSpeed Insights, який надає детальний звіт про якість користувацького досвіду вебсайту[24].

На першому раунді тестування(рис. 2.6.1) було помічено значні проблеми з ефективністю роботи застосунка(70/100) та використанням оптимальних методів(78/100).

Вебсайт мав високий Speed Index (одиниця, яка відображає час завантаження всього вмісту сайту у вікні перегляду користувача), він становив 2,3 секунди. Час завантаження Largest Contentful Paint(час рендерингу найбільшого зображення чи текстового блоку) становив 3,6 секунд. Також незадовільною була оцінка Cumulative Layout Paint(показник найбільшої кількості несподіваних змін макету сторінки), яка становила 0,137.

Основною причиною низькою оцінки за використання оптимальних методів були проблеми зі з’єднанням, а саме випадок змішаного контенту. За принципами PageSpeed Insight, всі сайти мають бути захищені протоколом HTTPS, навіть якщо вони не обробляють чутливі дані. В свою чергу змішаний контент передбачає, що деякі ресурси завантажуються через HTTP, незважаючи на те, що початковий запит було надіслано через HTTPS. В нашому випадку змішаним контентом були піктограми погодніх умов, які використовували ненадійне HTTP з’єднання.

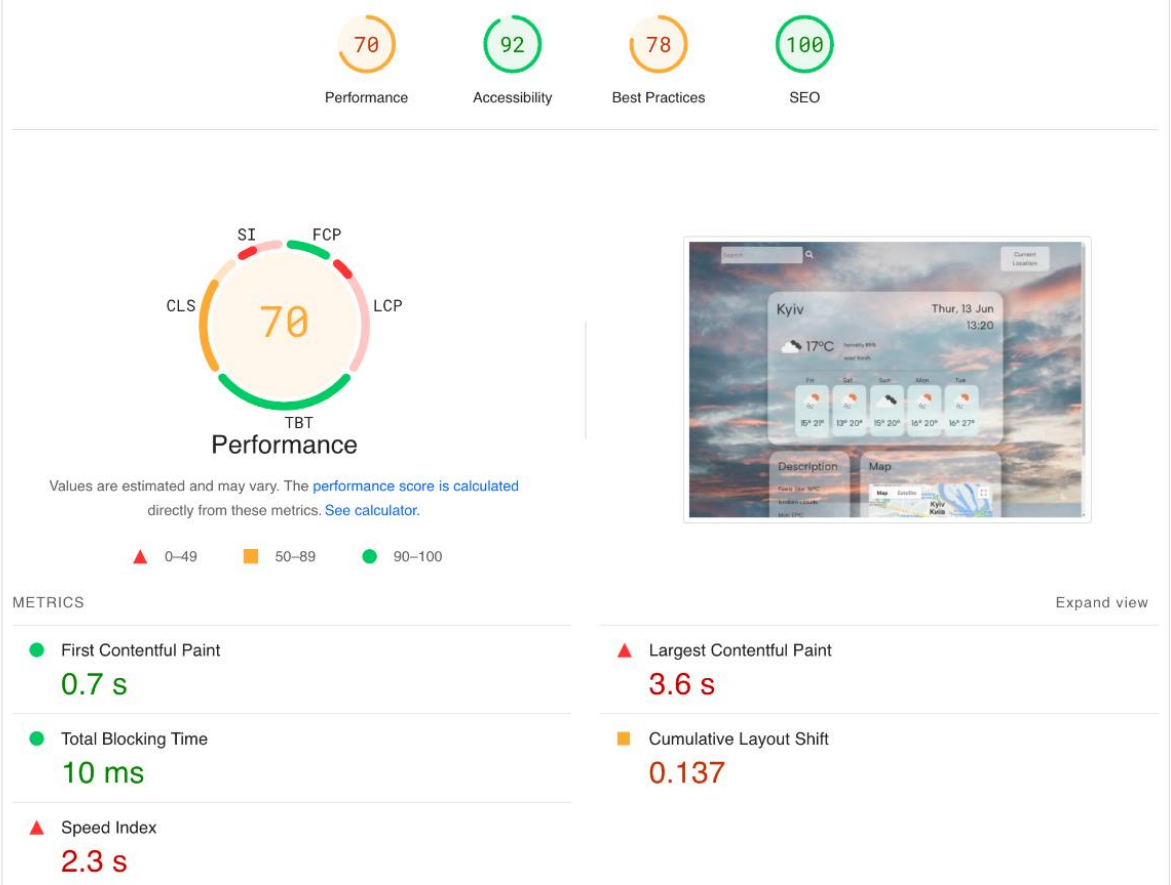

Рисунок 2.6.1 – Початковий звіт від PageSpeed Insights

Усі проблеми наведені вище було ліквідовано в процесі налагодження застосунка, як результат було проведено наступний звіт роботи сайту(рис. 2.6.2).

Зважаючи на остаточний звіт для даного вебсайту можна сказати, що застосунок WeatherWatch має високий рівень продуктивності(95/100). Його

Speed Index становить 1,2 секунди, так само як і Largest Contentful Paint – 1,2 секунди. Показник Cumulative Layout Paint зменшився до 0,09. Сайт має хороший ступінь доступності для осіб з спеціальними можливостями(92/100). Також для вебсайту добре налагоджена пошукова оптимізація – SEO(100/100). Покращене використання оптимальних методів(96/100).

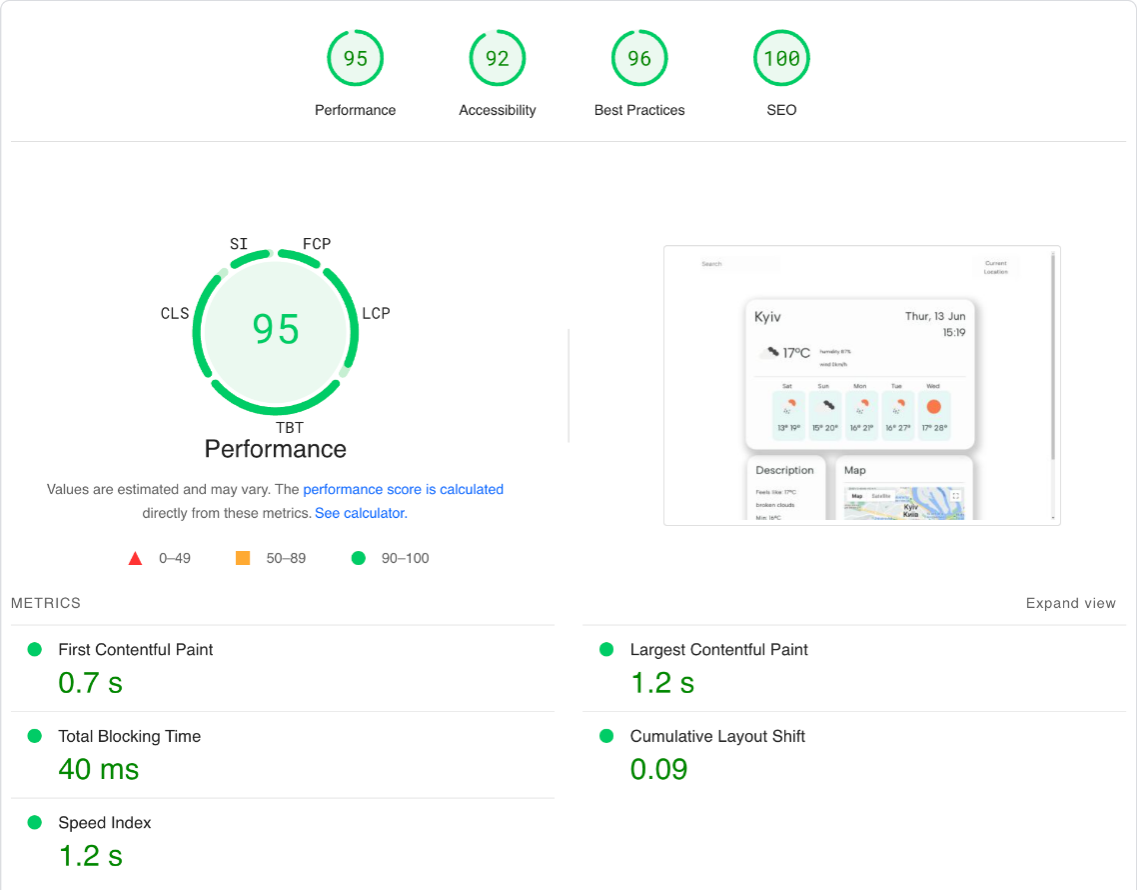

Рисунок 2.6.2 – Остаточний звіт від PageSpeed Insights

2.7. Рекомендації по використанню та впровадженню програмного засобу

Програмний застосунок призначений для щоденного практичного використання. Його основна функція – надання детальної погодної інформації на день та коротких прогнозів на наступні п’ять днів для користувачів в їх особистих цілях.

Даний застосунок функціонує в браузері з супроводом JavaScript. В майбутньому передбачається впровадження мобільної версії застосунка. Також, з огляду на потреби сайту, можливе розширення мовної підтримки.

ВИСНОВОК

Метою кваліфікаційної роботи було дослідження технологій та принципів створення односорінкових застосунків з допомого API. Для досягнення даної мети було: проведено аналіз наявних інструментів розробки, які характерні для SPA; визначено переваги та недоліки односторінкових застосунків; встановлено роль використання API у їх розробці.

Здійснено огляд сфер використання SPA у веб індустрії та проведено аналіз існуючих застосунків, які використовують дану архітектуру.

В результаті роботи було спроєктовано і розроблено прототип односторінкового застосунка. Процес розробки мав два основні етапи: розробка інтерфейсу користувача та розробка функціоналу програми. Для кінцевого продукту було проведено різні види тестування, включаючи тестування функціоналу, продуктивності та інші. Була проведена оцінка користувацького досвіду для сайту. Також, реалізований хостинг вебсайту на хмарному сервері.

Розроблений продукт вдало виконує усі поставлені завдання та може бути утилізований для практичного застосування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Choose Between Traditional Web Apps and Single Page Apps (SPAs) – Learn Microsoft URL: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/choose-between-traditional-web-and-single-page-apps#when-to-choose-spas (дата звернення: 25.05.2024).
2. Mikowski, Michael, and Josh Powell. Single page web applications: JavaScript end-to-end. Simon and Schuster, 2013.(дата звернення: 25.05.2024).
3. Single-Page Application Vs. Multiple-Page Application – Scand URL: https://scand.com/company/blog/single-page-application-vs-multi-page-application/ (дата звернення: 25.05.2024).
4. What Are Single Page Applications? – Netguru URL: https://www.netguru.com/blog/what-are-single-page-applications (дата звернення: 25.05.2024).
5. How to Build Single Page Application and Succeed in 2024 – Clockwise Software URL: https://clockwise.software/blog/single-page-applications-are-they-a-good-choice-for-your-project/ (дата звернення: 27.05.2024).
6. What Is a Single-Page Application? Architecture, Benefits, and Challenges – Spiceworks URL: https://www.spiceworks.com/tech/devops/articles/what-is-single-page-application/ (дата звернення: 27.05.2024).
7. What Is A Single Page Application? Meaning, Pitfalls & Benefits – Excellent Webworld URL: https://www.excellentwebworld.com/what-is-a-single-page-application/ (дата звернення: 27.05.2024).
8. Козлов, Сергей Валерьевич, і Павел Леонидович Ильин. "Средства разработки современных одностраничных веб-приложений." “ОБЩЕСТВО, ОБРАЗОВАНИЕ, НАУКА В СОВРЕМЕННЫХ ПАРАДИГМАХ РАЗВИТИЯ”, прошедшей (2022): ст. 39. (дата звернення: 28.05.2024).

9. Design Approach for Single-Page Apps: Tips and Examples – Lemberg Solutions URL: https://lembergsolutions.com/blog/design-approach-single-page-apps-tips-and-examples (дата звернення: 30.05.2024).
10. Получение данных с сервера – MDN Web Docs URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Fetching_data (дата звернення: 31.05.2024).
11. Axios API – Axios Docs URL: https://axios-http.com/uk/docs/api_intro (дата звернення: 31.05.2024).
12. Introduction to web APIs – MDN Web Docs URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction (дата звернення: 31.05.2024).
13. The WebSocket API – MDN Web Docs URL: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API (дата звернення: 31.05.2024).
14. What is API: Definition, Types, Specifications, Documentation – Altexsoft Software r&d Engineering URL: https://www.altexsoft.com/blog/what-is-api-definition-types-specifications-documentation/ (дата звернення: 31.05.2024).
15. Single-Page Apps & AJAX calls – DataDome URL: https://docs.datadome.co/docs/protect-singlepage-app (дата звернення: 31.05.2024).
16. JavaScript Development – Radix URL: https://radixweb.com/javascript-development (дата звернення: 03.06.2024).
17. Вступ до JavaScript – JavaScript.Info URL: https://uk.javascript.info/intro (дата звернення: 03.06.2024).
18. Итеративная разработка программного обеспечения – Web creator URL: https://web-creator.ru/articles/iterative_development (дата звернення: 03.06.2024).
19. Ключові методології розробки програмного забезпечення: робота команди зсередини – Wezom URL:

https://wezom.com.ua/ua/blog/metodologija-razrabotki-programmnogo-obespechenija (дата звернення: 03.06.2024).

20. Getting Started with Redux – Redux URL: https://redux.js.org/introduction/getting-started (дата звернення: 03.06.2024).
21. React – JavaScript-бібліотека для створення користувацьких інтерфейсів. React.js URL: https://uk.legacy.reactjs.org (дата звернення: 03.06.2024).
22. OpenWeather: Weather forecasts, nowcasts and history in a fast and elegant way – OpenWeather URL: https://openweathermap.org (дата звернення: 05.06.2024).
23. Особливості платформи Netlify – Foxminded URL: https://foxminded.ua/netlify-shcho-tse/ (дата звернення: 06.06.2024).
24. PageSpeed Insights – Google for Developers URL: https://pagespeed.web.dev/ (дата звернення: 06.06.2024).
25. Angular – Angular Dev URL: https://angular.dev/ (дата звернення: 07.06.2024).
26. Maps JavaScript API – Google Maps Platform URL: https://developers.google.com/maps/documentation/javascript/overview (дата звернення: 07.06.2024).
27. What Is API and How Does API Work? Quick introduction – Flatlogic URL: https://flatlogic.com/blog/what-is-api-and-how-api-works/ (дата звернення: 07.06.2024).

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

Даний сайт буде прототипом односторінкового вебзастосунку, який використовує браузерні та сторонні API.

Вступ

Результатом технічного завдання буде вебсайт для показу прогнозу погоди.

1. Підстави для розробки

Документ, на підставі якого ведеться розробка: Завдання наукового керівника на кваліфікаційну роботу за спеціальністю комп'ютерні науки.

Організація, що затвердила документ: Волинський національний університет імені Лесі Українки.

2. Призначення розробки

Головним призначенням веброзробки є демонстрація переваг односторінкових застосунків та функціональності, яку їм надають API.

3. Вимоги до програми чи програмного продукту

3.1. Вимоги до функціональних характеристик

Кінцевий продукт має забезпечувати такі функції:

- динамічний користувацький інтерфейс;
- пошукова система, яка дозволить користувачам знаходити погоду за назвою міста;
- API інтеграція;
- використання реальних даних про погодні умови з стороннього сервера;
- використання сервісів геолокації для визначення розташування користувача;

Застосунок повинен мати простий та зрозумілий дизайн з зручною навігацією.

3.2. Вимоги до надійності

Для забезпечення надійності потрібно гарантувати хостинг сайту з безпечним HTTP з'єднанням. Крім цього, необхідно забезпечити приватність коритувача під час збору даних для геолокації.

3.3. Умови експлуатації

Вебсайт має працювати на різних пристроях (настільних комп'ютерах, ноутбуках, смартфонах, планшетах) і веб-браузерах (Chrome, Firefox, Safari, Edge), щоб відповідати різноманітним технологічним умовам користувачів. Також він повинен функціонувати за різних умов мережі, включаючи ненадійні з'єднання.

3.4. Вимоги до складу і параметрів технічних засобів

Для даного вебсайту необхідний супровід таких технологій, як HTML5, CSS3 і Bootstrap та JavaScript і її бібліотек.

Програма має бути сумісною з API, який використовуються для отримання даних про погоду. Це передбачає забезпечення того, що програма може робити запити та правильно інтерпретувати відповіді. Формат даних (в цьому випадку JSON), що застосунок отримує з сервера, має бути сумісний із технологіями для обробки та відображення даних.

4. Стадії і етапи розробки

1. Визначення цілей та вимог.
2. Вибір технологій для розробки.
3. Створення дизайну застосунку.
4. Розробка та стилізація інтерфейсу.
5. Впровадження пошукової системи.
6. Реалізація API інтеграції.
7. Тестинг і оптимізація.
8. Розгортання на хмарному сервері.

5. Порядок конролю і приймання

По закінченню розробки, необхідно провести наступні види тестування сайту:

- перевірка функціональності;
- тестування швидкодії;
- сумісність з різними браузерами та пристроями;
- перевірка дотримання всіх вимог.

ДОДАТОК Б

ІНСТРУКЦІЯ КОРИСТУВАЧУ

1. Загальні відомості

Назва вебсайту – “WeatherWatch”.

2. Функціональне призначення

Програмний застосунок призначений для надання детальної погодної інформації на день та коротких прогнозів на наступні п’ять днів для користувачів в їх особистих цілях.

3. Умови застосування програми

Застосунок функціонує в браузері з підтримкою JavaScript.

4. Опис роботи програми

Після повного завантаження вебсайту на сторінці буде відображатися дефолтне значення міста – Київ та повний прогноз для нього.

Щоб почати роботу користувач може використати пошукове поле та ввести туди назву потрібного міста або використати кнопку “Current Location”, за умови надання доступу до своєї геолокації.

Після задання значення міста сторінка динамічно обновить дані та буде показувати прогноз за запитом користувача.

АНОТАЦІЯ

Рубік А. Д. Розробка односторінкових вебзастосунків з використанням API.

Кваліфікаційна робота на здобуття освітнього ступеня «бакалавр» за спеціальністю 122 Комп'ютерні науки. Волинський національний університет імені Лесі Українки, Луцьк, 2024 р.

Дипломна робота присвячена вивченню односторінкових вебзастосунків(SPA). В процесі роботи було описано принципи роботи односторінкових вебзастосунків та проведено порівняння з багатосторінковими застосунками, також було визначено переваги і недоліки використання кожної з цих акрітектур. У роботі було проведено визначення API та описано роль використання API в розробці SPA. Проведено аналіз типового інструментарію потрібного для розробки SPA.

Кінцевий програмний продукт є прототипом односторінкового застосунка та використовує різні практики для забезпечення позитивного користувацького досвіду. Розроблений застосунок "WeatherWatch" є вебсайтом для показу прогнозу погоди, який забезпечує такі функції як динамічне оновлення даних, реалізація інтерактивної мапи, наявність пошукової системи, використання сервісів геолокації, отримання даних із стороннього сервера, API інтеграція. Дана робота описує всі стадії розробки застосунка від планування до розробки і тестування.

Ключові слова: односторінковий вебзастосунок, SPA, багатосторінковий вебзастосунок, MPA, архітектура, API, JavaScript, пошукова оптимізація.