**РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА СУЧАСНИХ WEB ДОДАТКІВ**

2.1 Специфікація вимог до системи

Специфікація вимог є ключовим етапом у розробці програмного забезпечення, що визначає функціональні та нефункціональні вимоги до системи. Вона дозволяє уточнити очікування від програмного продукту і забезпечити відповідність розробленої системи бізнес-потребам та вимогам користувачів.

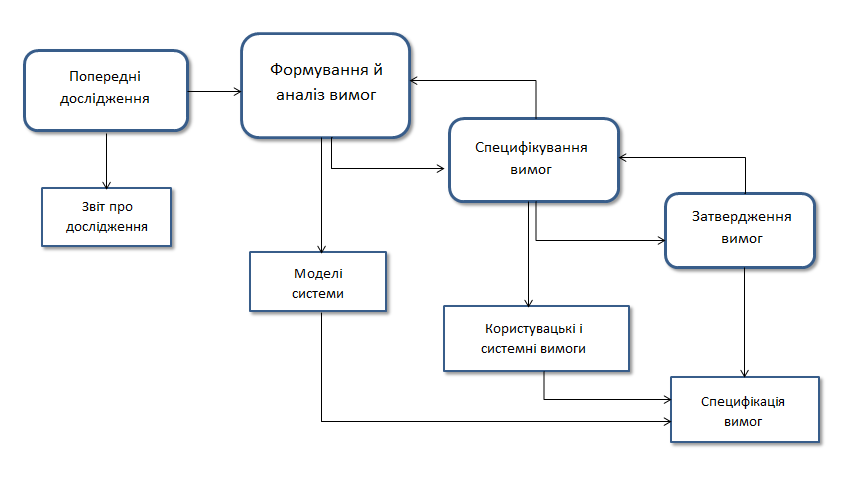


Рис. 2.1 – Специфікація вимог до системи.

1. Системні вимоги:

- Мінімальні вимоги до обладнання: Для використання додатка необхідний комп'ютер або мобільний пристрій з встановленим веб-браузером або підтримкою Node.js для виконання TypeScript коду.

- Мінімальні вимоги до програмного забезпечення: Підтримка сучасних веб-браузерів (Chrome, Firefox, Safari тощо) або встановлення Node.js на мобільний пристрій для виконання серверного коду.

- Вимоги до мережі: Для завантаження додатка та обміну даними з сервером потрібний доступ до Інтернету.

2. Функціональні вимоги:

- Керування даними: Додавання, редагування та видалення даних користувачами з різними ролями та правами доступу.

- Відображення даних: Створення інтерфейсів для відображення геодезичних даних, збережених у базі даних, та їхнього оновлення у реальному часі

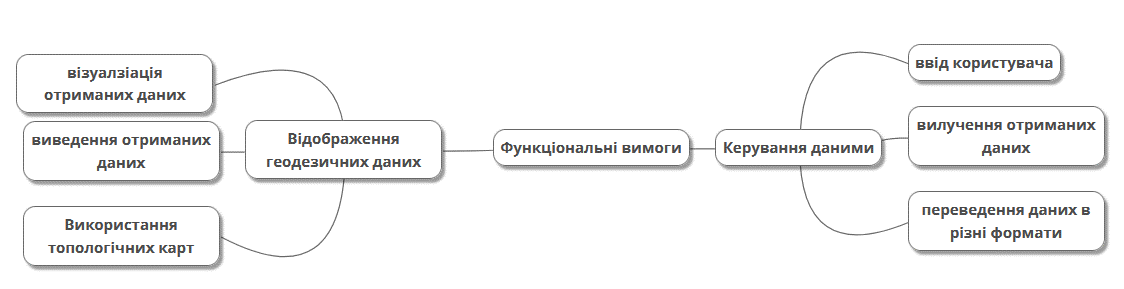


Рис. 2.2 – Функціональні вимоги до системи.

3. Безпека та доступність:

- Шифрування даних: Забезпечення конфіденційності інформації за допомогою шифрування.

- Захист від атак: Проведення заходів щодо захисту від XSS та CSRF атак, а також валідація введених даних користувачем.

- Обмеження доступу до даних: Налаштування прав доступу до функцій та ресурсів залежно від ролей користувачів.

- Аудит доступу: Відстеження та аналіз дій користувачів для забезпечення безпеки системи.

- Регулярні оновлення: Забезпечення актуальності системи та захисту від відомих вразливостей через регулярні оновлення програмного забезпечення."



Рис. 2.3 – Безпека та доступність.

4. Інтеграція та розширення:

- Розширення функціоналу: Можливість розширення функціоналу додатка за допомогою плагінів або розширень.

- Інтеграція з іншими системами: Забезпечення можливості інтеграції з іншими системами чи API для обміну даними та розширення функціоналу.

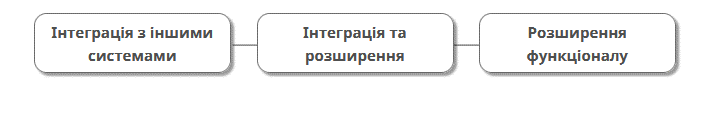


Рис. 2.4 – Інтеграція та розширення.

Додаткові функціональні вимоги:

5. Конвертор координат:

- Користувачам буде доступна можливість конвертувати координати між різними системами (наприклад, з географічної в систему UTM або наоборот). Ця функція дозволить користувачам зручно працювати з географічними даними, надаючи їм можливість переводити координати у формат, зрозумілий для їхнього контексту.

6. Обчислення відстаней між двома точками:

- Додаток надасть користувачам зручний інструмент для визначення відстані між двома точками на мапі. Це може бути корисно для планування маршрутів, визначення відстані між об'єктами або для розрахунку відстаней у геодезичних роботах.

7. Визначення висоти точки:

- Користувачі матимуть можливість отримувати інформацію про висоту конкретної точки на мапі. Це може бути корисно для альпіністів, туристів або для роботи з географічними даними у наукових дослідженнях.

8. Середня точка між двома точками:

- Додаток автоматично розраховуватиме координати середньої точки між двома вказаними точками. Ця функція може бути корисною для планування маршрутів, розташування об'єктів у просторі або для поділу території між декількома точками.

9. Азимут між двома точками:

- Додаток надасть можливість визначити азимут (кут напряму) між двома вказаними точками на мапі. Ця функція може бути корисною для навігації, планування маршрутів або для геодезичних обчислень.

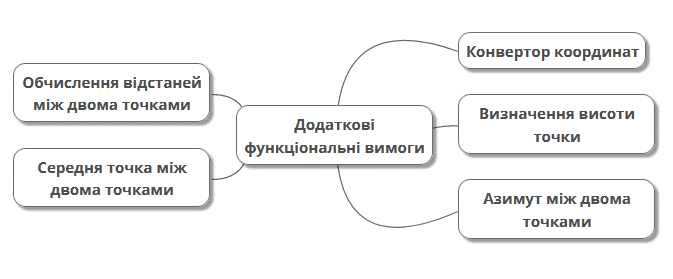


Рис. 2.5 – Додаткові функціональні вимоги.

Ці додаткові функціональні вимоги значно розширюють можливості додатка і забезпечують користувачам зручний та функціональний інструмент для роботи з географічними даними. Вони сприятимуть покращенню досвіду користувача та розширенню його можливостей у використанні системи.

Ця специфікація вимог до системи детально описує всі аспекти програмного забезпечення, враховуючи використання новітніх технологій та особливостей їхнього впровадження.

Зокрема, зазначимо, що використання SSR (Server-Side Rendering) технологій на базі Node.js та TypeScript дозволяє забезпечити швидкий та ефективний рендеринг веб-сторінок на стороні сервера перед їхнім відправленням користувачам. Це покращує продуктивність та швидкодію додатка, зменшуючи час завантаження сторінок та поліпшуючи їхню доступність для користувачів з різних пристроїв та з'єднаннями.

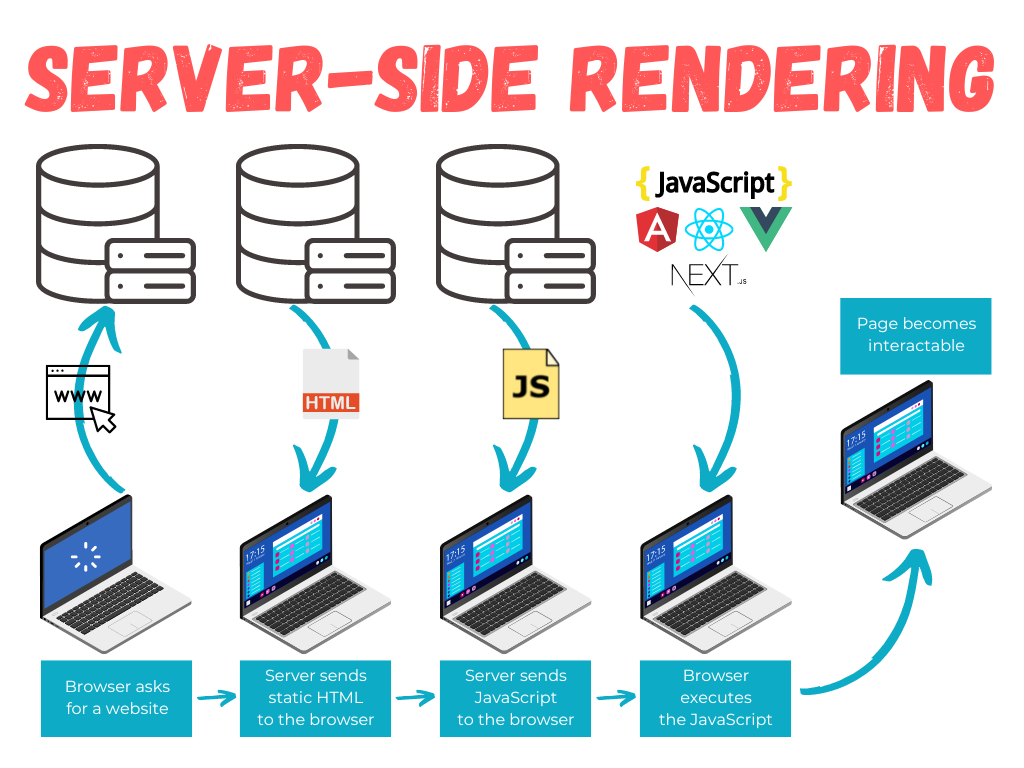


Рис. 2.6 – Використання SSR (Server-Side Rendering) у веб технологіях.

2.2 Розробка UML діаграм

UML використовує різноманітні діаграми як графічні засоби для представлення статичних чи структурних аспектів системи, а також для відображення її динамічної поведінки.

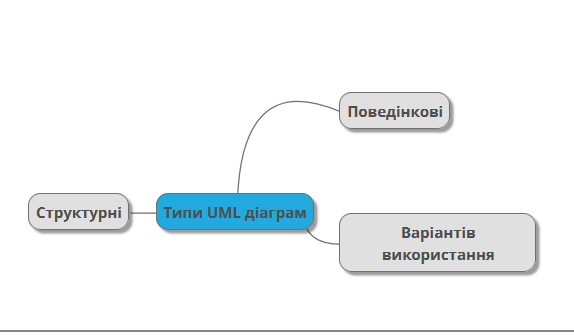


Рис. 2.7 – Типи UML діаграм.

Діаграми UML поділяються на структурні та поведінкові категорії. На початкових етапах проектування їх можна використовувати для ілюстрації бажаних конструкцій, формування міцної архітектури та налагодження зв'язків між компонентами. Під час процесу розробки вони дозволяють візуалізувати взаємодію системи з її оточенням через різні сценарії використання.

Структурні діаграми (Діаграма класів, Компонентна діаграма, Об'єктна діаграма, Складова діаграма структури, Діаграма розгортання, Пакетна діаграма) відображають систему - класи, об'єкти, частини та модулі, фізичні вузли, компоненти та інтерфейси. Вони також демонструють зв'язки між цими елементами - класами, які успадковують від інших класів, об'єктами, які володіють іншими об'єктами, які класи належать до яких пакетів, які вузли й одиниці оточують їх.

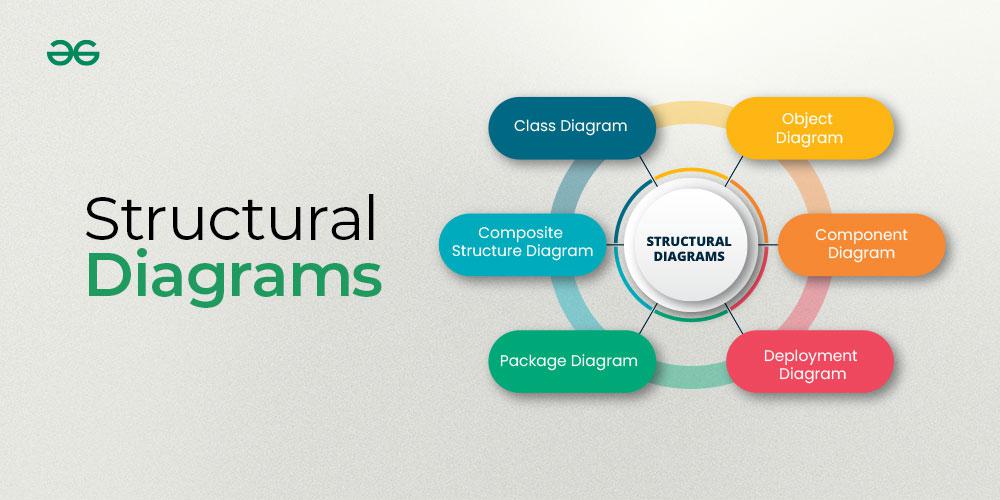


Рис. 2.8 – Структурні UML діаграми.

Діаграми поведінки (Діаграма діяльності, Діаграма послідовності випадків, Діаграма станів, Діаграма взаємодії, Діаграма часу) показують, як система веде себе та взаємодіє з собою та іншими сутностями (користувачами, іншими системами). Вони відображають дані рухаючись через систему, як об'єкти спілкуються одні з одними, яка операція часу виконується на системі та взаємні позиції станів системи.

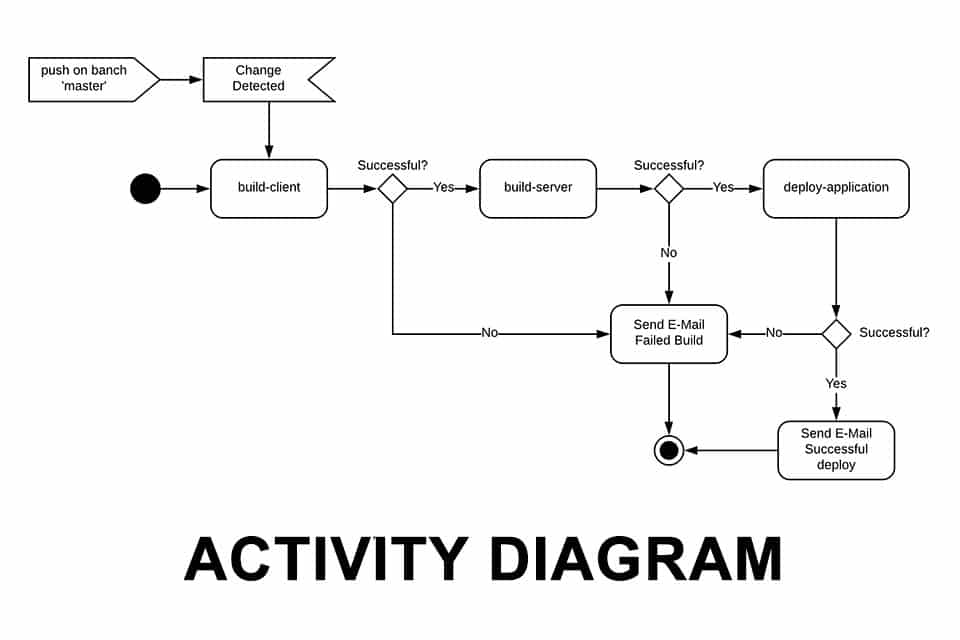


Рис. 2.9 – Поведінкові UML діаграми.

Отже, далі буде представлена діаграма варіантів використання. Діаграма використання - це не перелік кроків, що визначають взаємодію між акторами (іншими діючими, вже існуючими системами) та самою системою. Діаграма використання зображує функції системи. Це допомагає розробникам зрозуміти функціональні вимоги до системи.

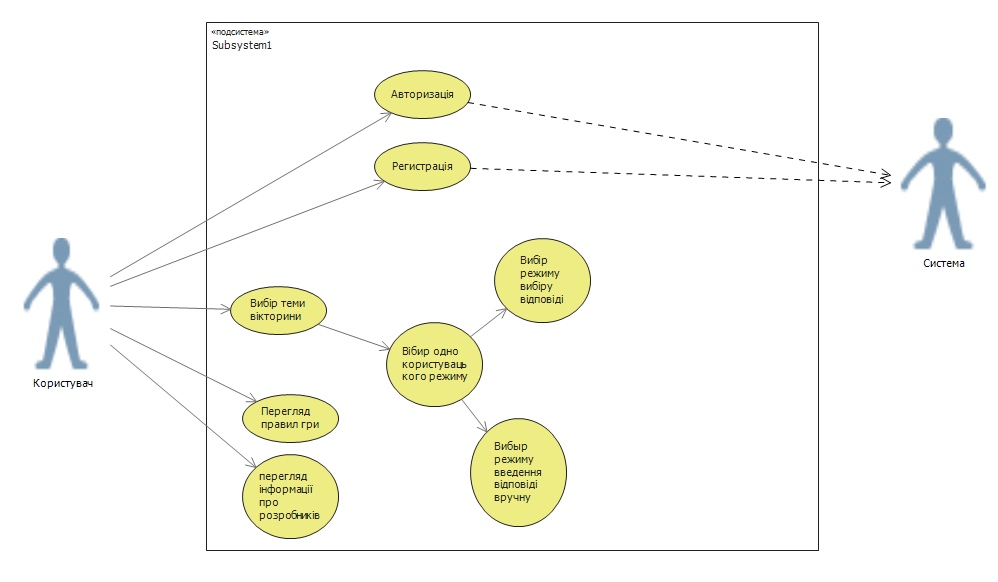


Рис. 2.10. - UML діаграма варіантів використання.

2.3 Вибір технології для реалізації основних модулів системи

При розробці основних модулів системи виникає потреба в обдуманому виборі технологій, що відповідають вимогам ефективності, продуктивності та зручності в розробці. У зв'язку з цим було прийнято рішення використовувати Node.js, TypeScript, API OpenTopoMap та бібліотеку Leaflet для реалізації веб-додатка.

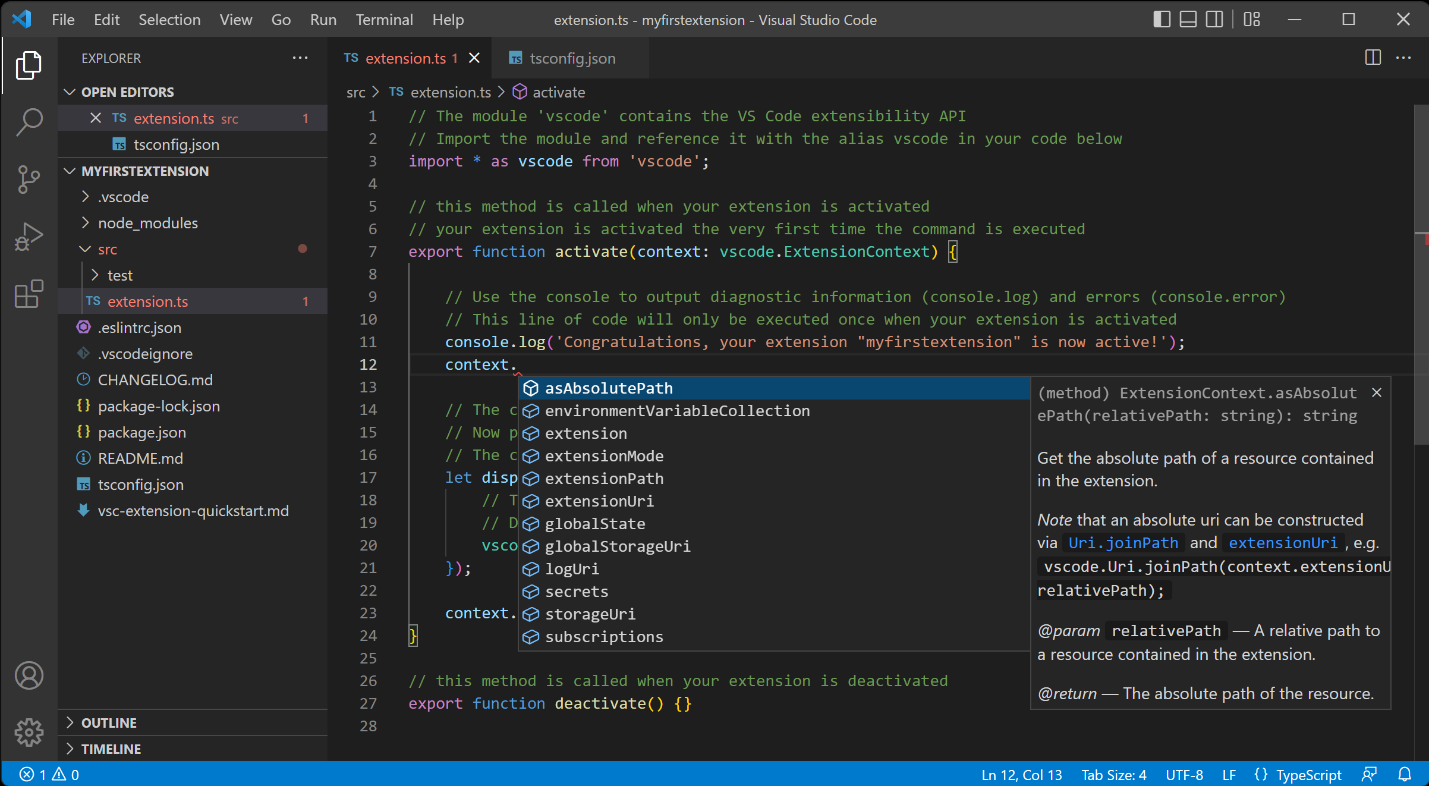


Рис. 2.11. – Приклад синтаксису TypeScript.

Node.js обрано як основу серверної логіки веб-додатка. Використання його асинхронної та подієвої моделі програмування забезпечує високу продуктивність та масштабованість додатку. TypeScript, у свою чергу, використовується як мова програмування для Node.js, надаючи переваги статичної типізації та інші функції, які полегшують розробку, такі як автодоповнення коду та підвищена безпека.

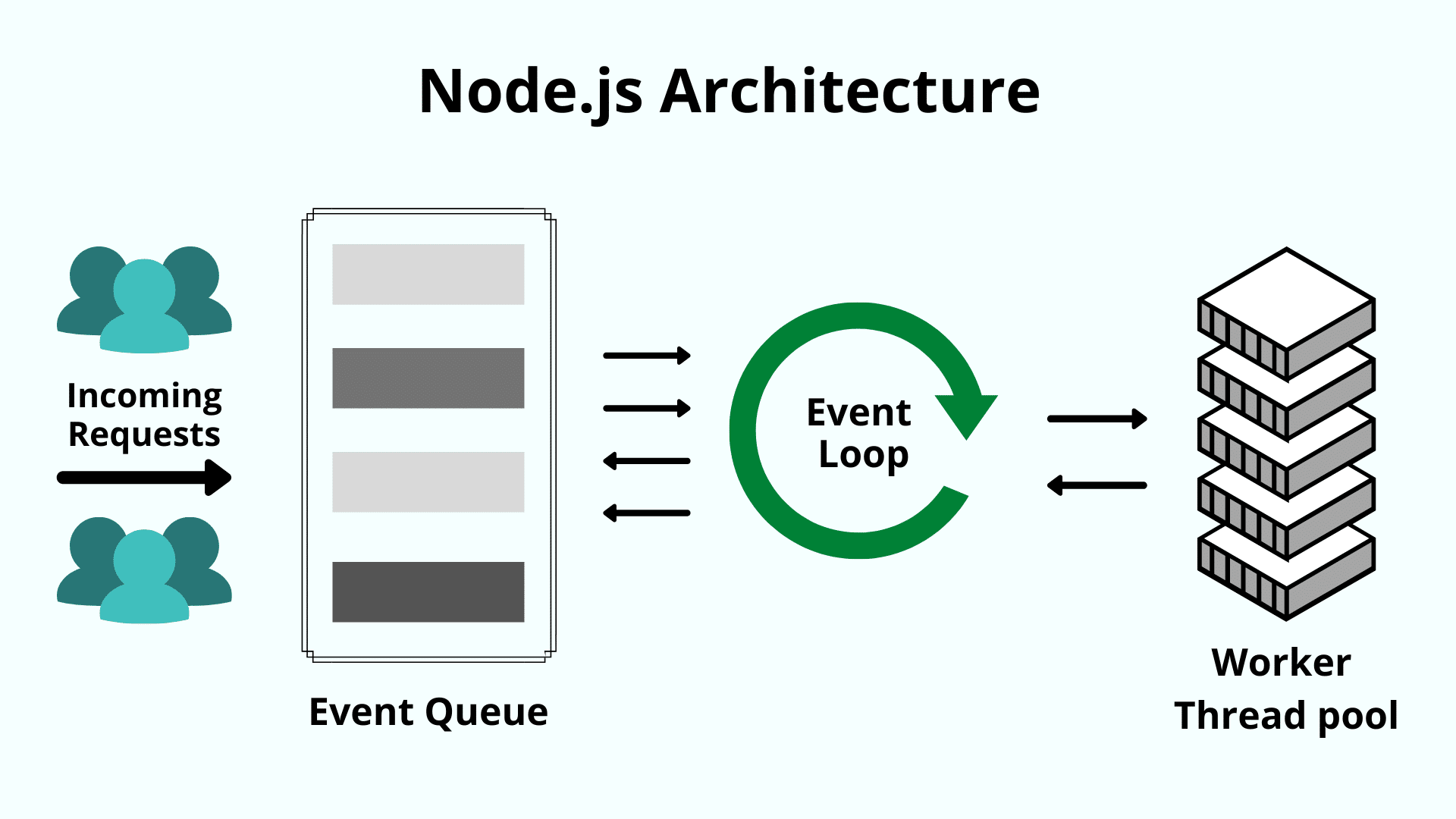


Рис. 2.12. – Архітектура Node.js.

API OpenTopoMap:

Як джерело географічних даних обрано API OpenTopoMap. Це дозволяє отримувати високоякісні топографічні дані, які можна використовувати для створення інтерактивних карт та візуалізації географічних об'єктів.

Для відображення географічних даних та створення інтерактивних карт веб-додатка використовується бібліотека Leaflet. Вона надає потужні інструменти для роботи з географічними даними та можливості налаштування відображення карти згідно з потребами проекту.

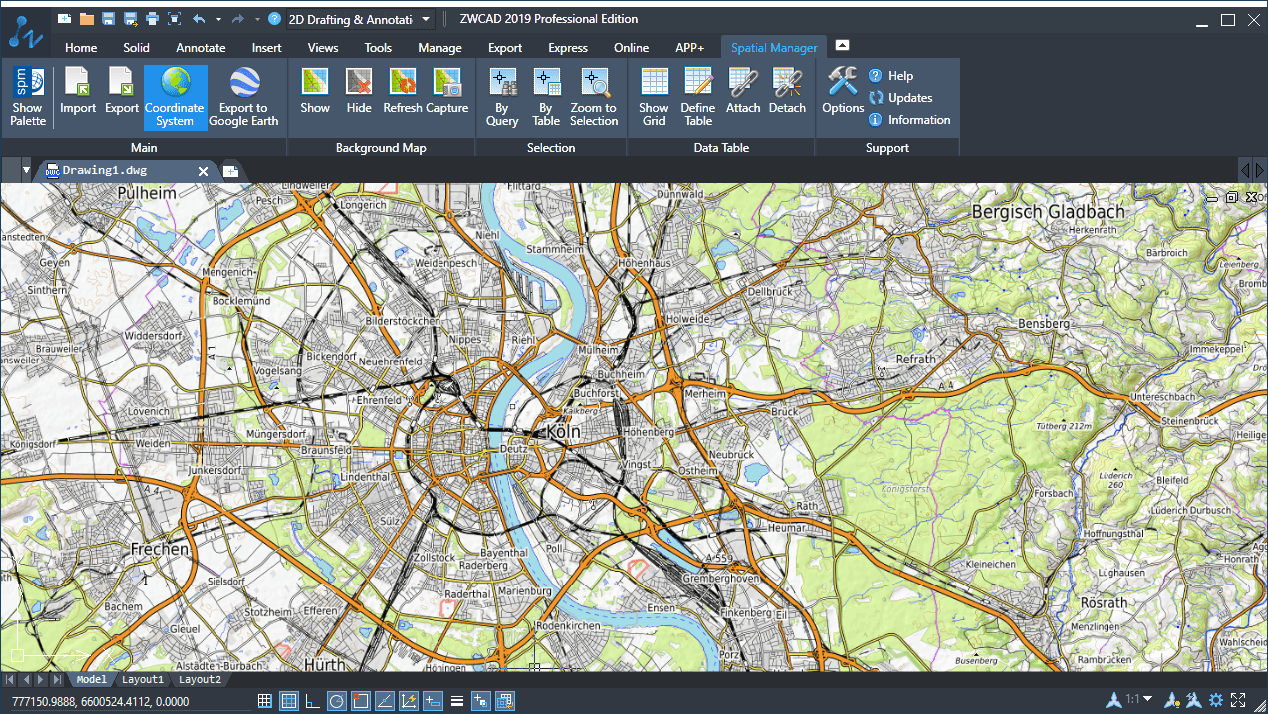


Рис. 2.13. – API OpenTopoMap.

Цей набір технологій надає широкі можливості для реалізації веб-додатка з високоякісним географічним функціоналом. Використання API OpenTopoMap та бібліотеки Leaflet дозволяє відобразити та взаємодіяти з географічними даними у зручний та ефективний спосіб, що робить веб-додаток більш привабливим для користувачів.

Обраний підхід до вибору технологій забезпечить якість та надійність веб-додатка, забезпечуючи при цьому високу продуктивність та зручність в розробці.

2.4 Розробка графічного інтерфейсу Web-орієнтованого сервісу

Для розробки графічного інтерфейсу веб-орієнтованого сервісу було використано комплекс технологій, який включає в себе Next.js, Node.js, TypeScript, а також api OpenTopoMap та OpenTopoData разом з бібліотекою Leaflet.

Next.js є реактивним фреймворком для розробки веб-додатків на основі React. Він дозволяє створювати швидкі, масштабовані та SEO-оптимізовані веб-додатки. Особливості Next.js включають в себе автоматичне підтримання рендерингу на стороні сервера, вбудовану оптимізацію зображень та сторінок, а також підтримку TypeScript для розширення безпеки та продуктивності в розробці.

Node.js - це середовище виконання JavaScript, що дозволяє виконувати код JavaScript на сервері. TypeScript - це розширення JavaScript, яке додає статичну типізацію та інші сучасні функції до мови. Використання Node.js разом з TypeScript надає можливість писати безпечний та підтримуваний код, що сприяє якості та надійності веб-додатку.

API OpenTopoMap та OpenTopoData використовуються для отримання геоданих та топографічних карт. Ці дані можуть бути використані для відображення топографічної інформації на мапах у веб-додатку.

Leaflet - це відкрите програмне забезпечення для відображення мап та геоданих у веб-додатках. Вона надає простий та легкий у використанні API для відображення різноманітної географічної інформації на картах.



Рис. 2.14. – Leaflet API.

Особливості використання Next.js в розробці інтерфейсу web-орієнтованих сервісів:

Next.js надає ряд переваг для розробки інтерфейсу веб-орієнтованих сервісів, зокрема:

- Рендеринг на стороні сервера (SSR): Дозволяє отримувати повний HTML-код сторінки на стороні сервера, що покращує SEO та робить веб-додаток більш доступним для пошукових систем.

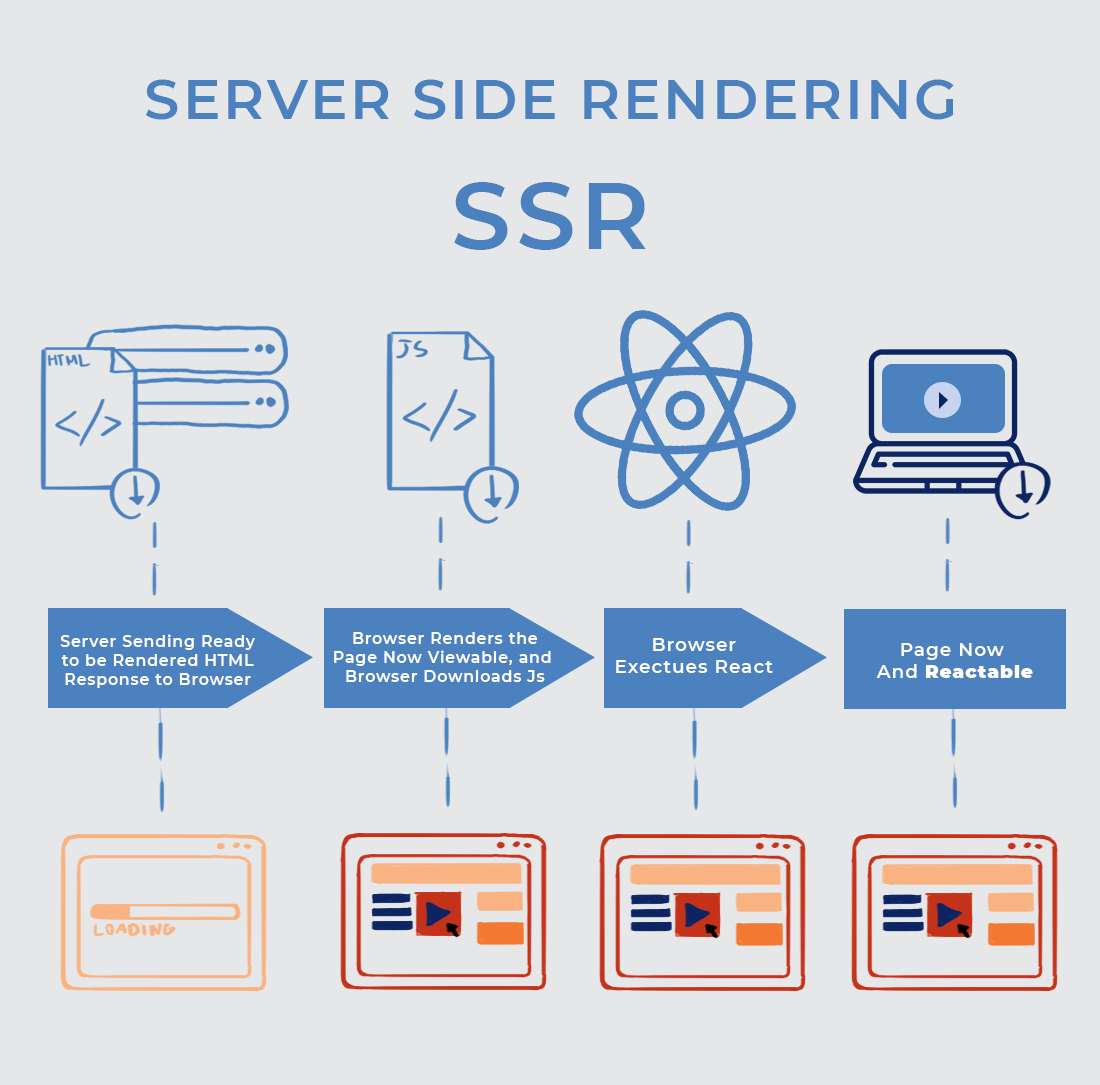


Рис. 2.15. – Використання SSR (Server-Side Rendering) у веб технологіях.

- Статичне згортання (SSG): Надає можливість попереднього рендерингу статичних сторінок під час збірки проекту, що поліпшує швидкодію та продуктивність додатку.

- Підтримка TypeScript: Дозволяє писати безпечний та підтримуваний код за допомогою статичної типізації та інших функцій TypeScript.

- Розширюваність: Next.js має розширений екосистему плагінів та модулів, які спрощують розробку та розширення функціональності веб-додатку.

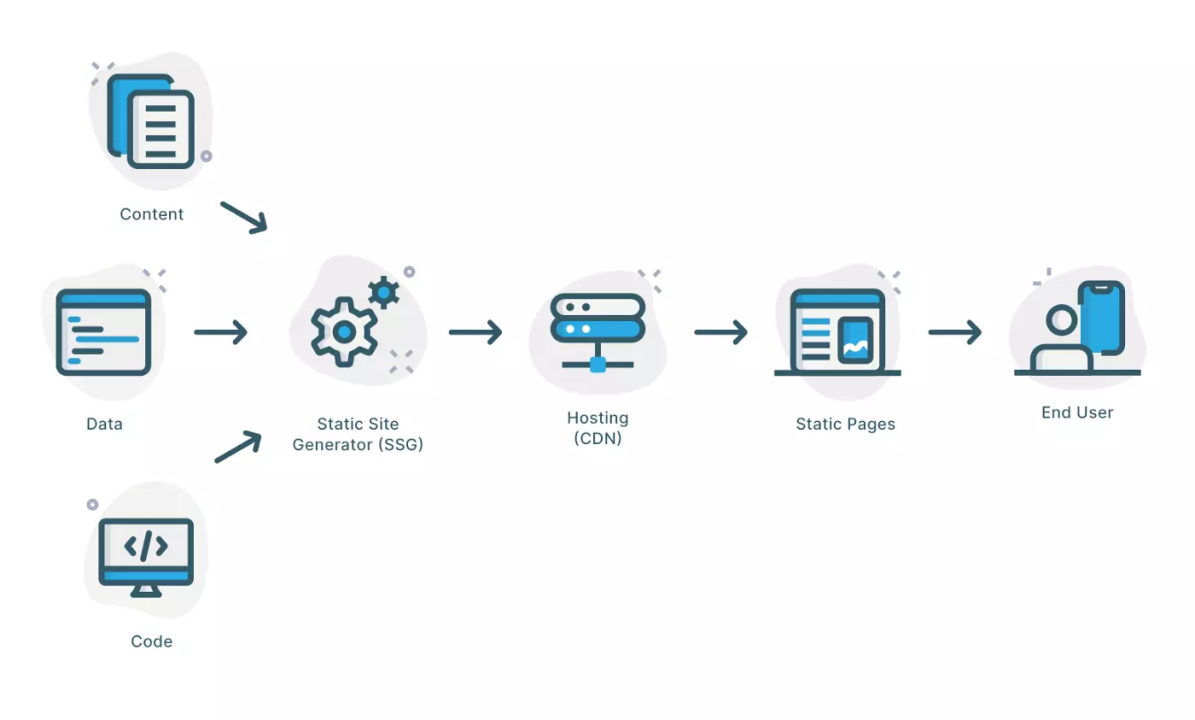


Рис. 2.16. – Використання SSR (Server-Side Rendering) у веб технологіях.

Отже, використання Next.js разом з Node.js, TypeScript, api OpenTopoMap, OpenTopoData та бібліотекою Leaflet дозволяє створити потужний та ефективний графічний інтерфейс для веб-орієнтованого сервісу, забезпечуючи якість, продуктивність та швидкодію веб-додатка.

Висновки до розділу 2.

Висновки до розділу 2

У другому розділі нашої курсової роботи ми зосередилися на специфікації вимог до системи "Geo-Helper", розробці UML діаграм, виборі технологій для реалізації основних модулів та розробці графічного інтерфейсу веб-сервісу. Цей розділ визначає технічні та функціональні аспекти нашого проекту, що будуть визначальними для подальшої реалізації.

Основні висновки розділу:

1. Специфікація вимог до системи: Визначення функціональних та нефункціональних вимог є ключовим етапом у розробці будь-якої системи. В цьому розділі було детально описано вимоги до функціональності та характеристики, які має мати розроблена система.

2. Розробка UML діаграм: Використання UML діаграм дозволяє систематизувати та візуалізувати архітектурні та структурні аспекти системи. На основі UML діаграм було розроблено моделі взаємодії та структури системи, що сприяє кращому розумінню її функціональності та логіки роботи.

3. Вибір технології для реалізації основних модулів системи: Під час вибору технологій для реалізації модулів системи були враховані вимоги до функціональності, ефективність, масштабованість та інші критерії. В результаті було обрано найбільш підходящі технології для кожного модуля.

4. Розробка графічного інтерфейсу Web-сервісу: Створення зручного та привабливого графічного інтерфейсу є важливим аспектом успішної реалізації веб-додатку. В даному розділі було описано процес розробки інтерфейсу користувача для розробленої системи , з урахуванням зручності використання та естетичних аспектів.

Цей розділ є ключовим для розуміння технічних та функціональних характеристик нашого проекту і служить основою для подальшої реалізації та впровадження веб-додатку.