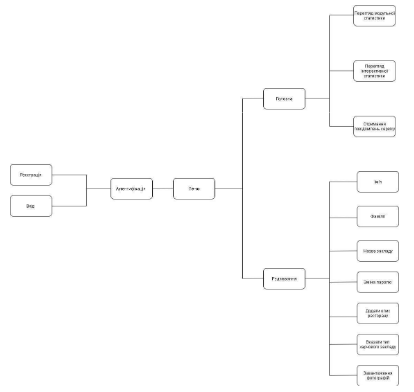
**РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ РОЗРОБЛЕНОГО WEB-ДОДАТКУ**

3.1 Опис модулів та компонентів розробленого Web-сервісу

Розроблений веб-сервіс призначений для надання комплексного рішення для конвертації координат та функцій картографування.

Отже, нижче (рис. 3.1) наведено структурну схему розробленої системи.



Розроблений сервіс має такий функціонал:

1. Конвертація географічних координат між різними системами, така як перетворення форматів UTM, MGRS, та WGS84.

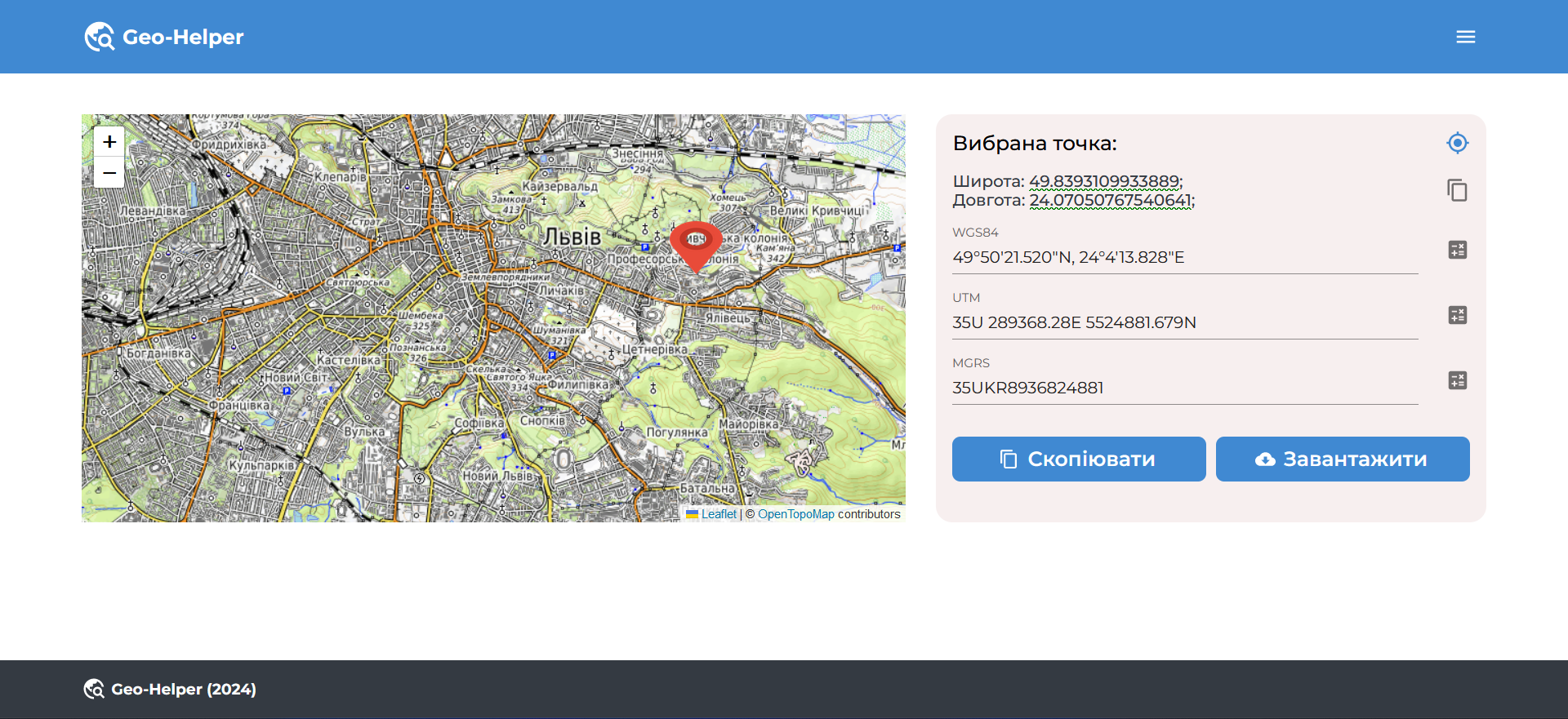


Рис. 3.2 – Конвертація географічних координат в різних системах.

2. Точне обчислення відстаней між двома точками на поверхні Землі за допомогою формули Гаверсінуса, виміряних у метрах.

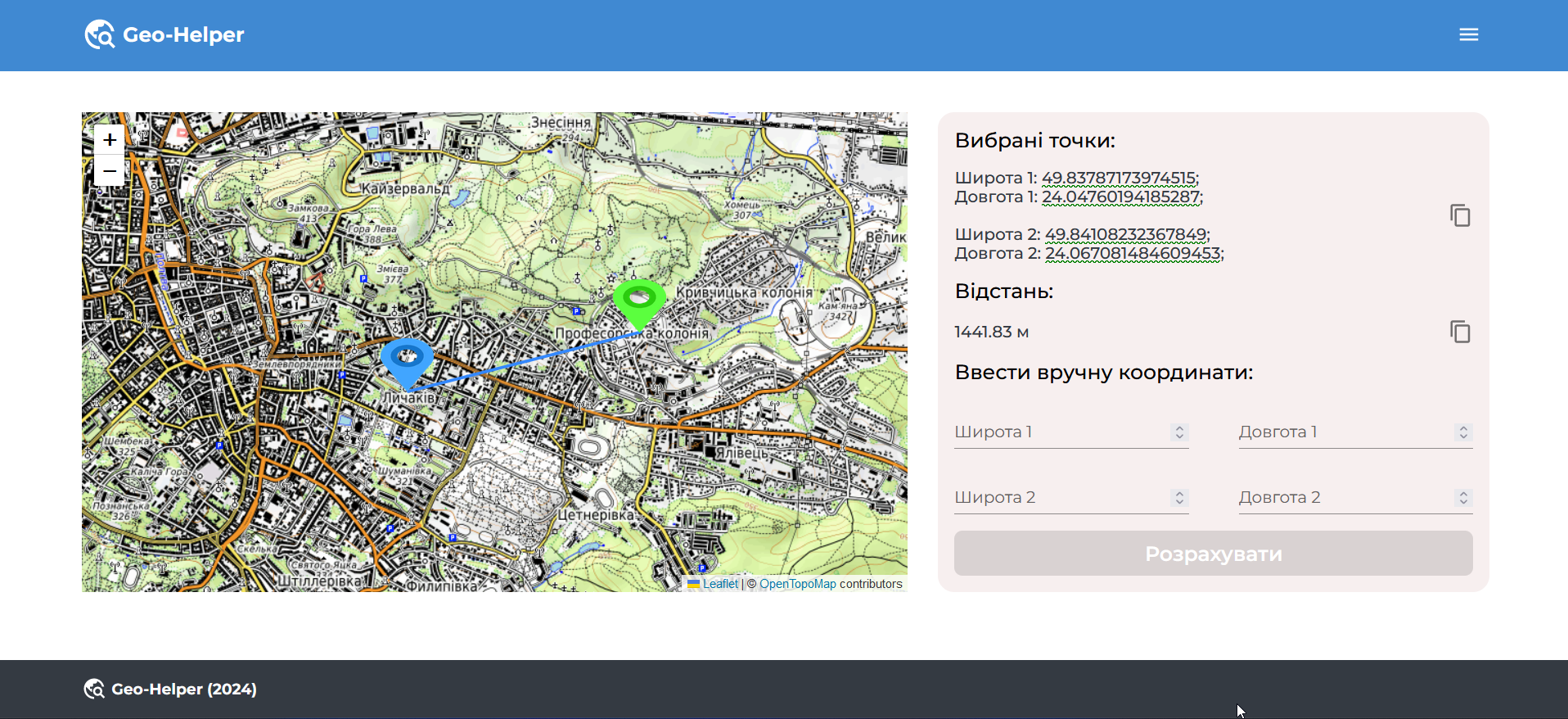


Рис. 3.3 – Обчислення відстані між точками.

3. Визначення висоти заданої точки шляхом застосування алгоритмів, які враховують геоїд та еліпсоїд Землі.

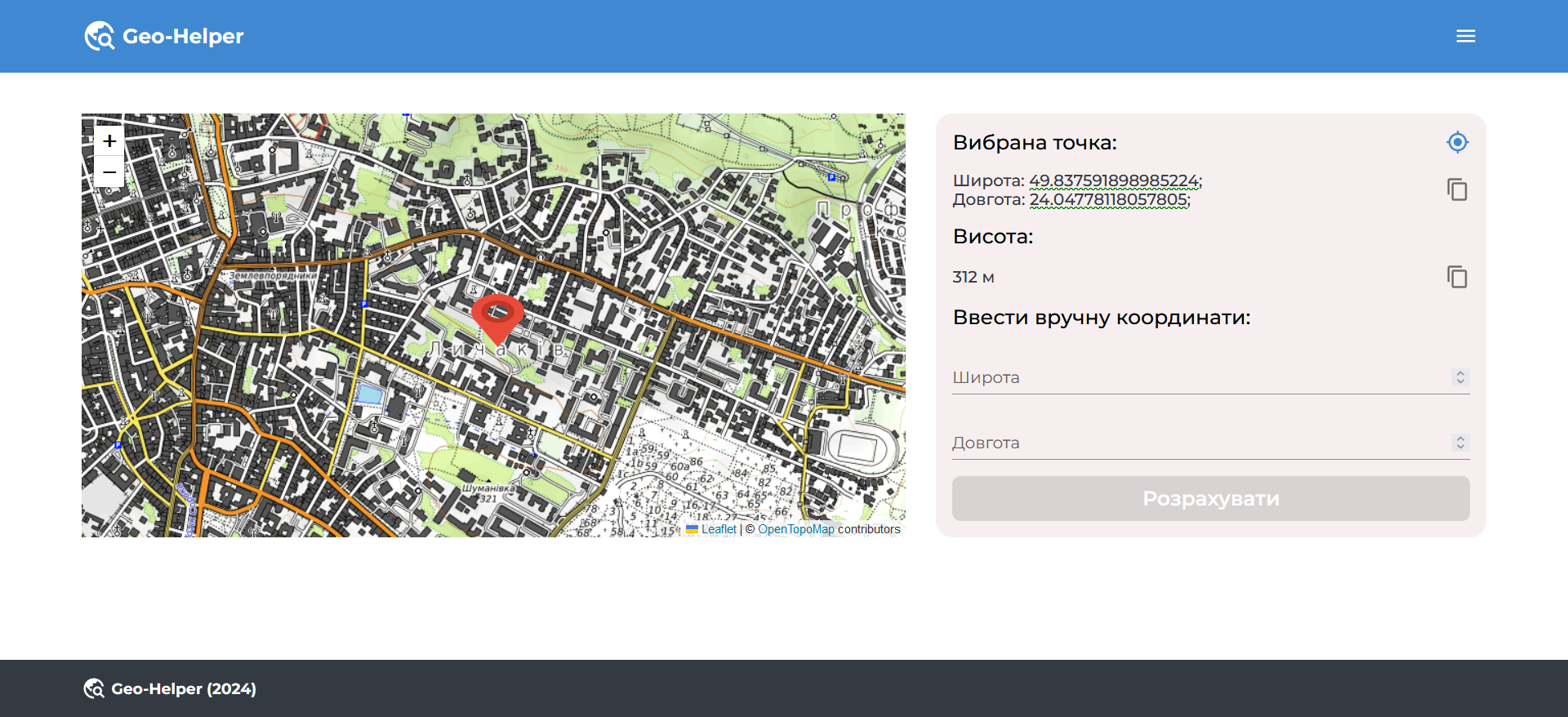


Рис. 3.4 – Визначення висоти заданої точки над рівнем моря.

4. Знаходження середньої точки між двома координатами шляхом усереднення їхніх відповідних широт і довгот.

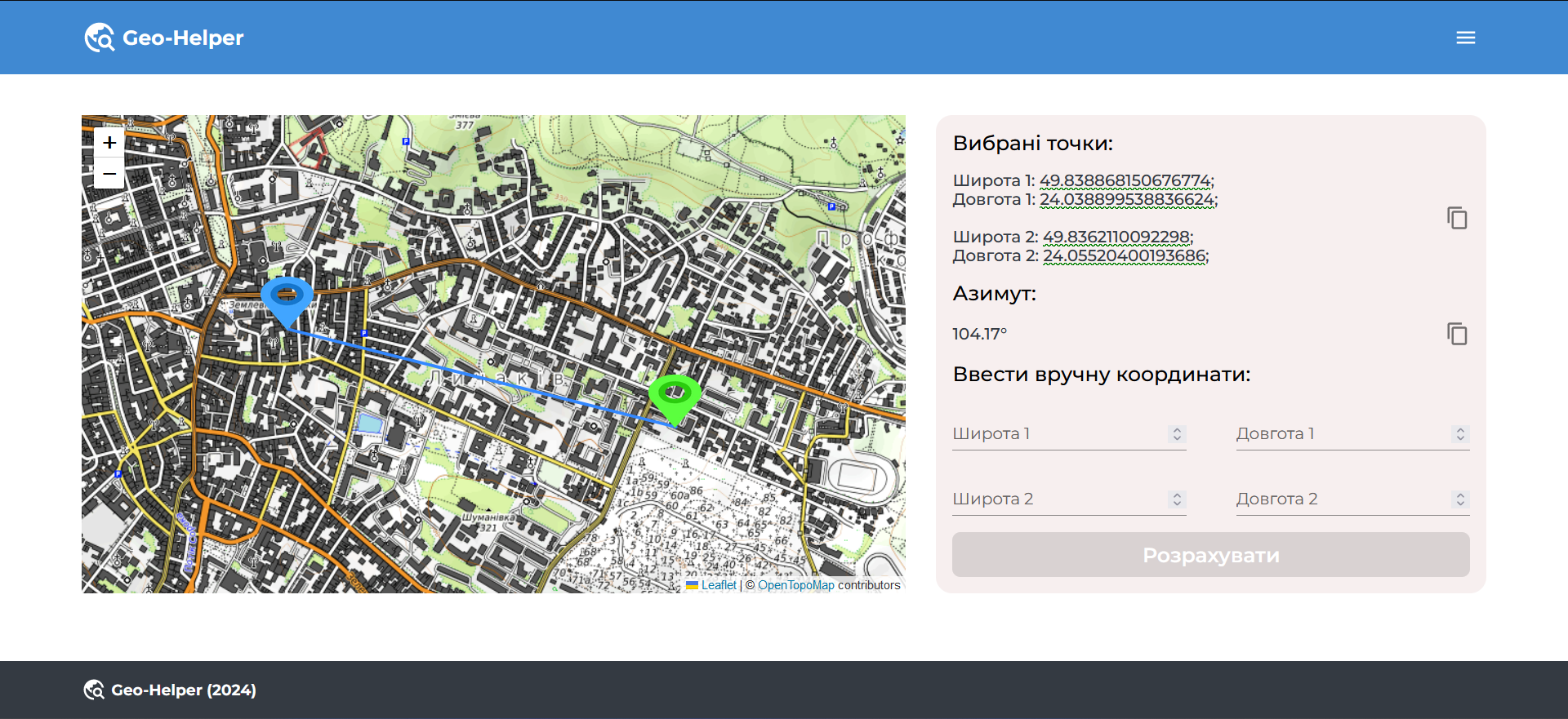


Рис. 3.5 – Знаходження середньої точки між двома координатами.

5. Обчислення азимуту між двома точками за допомогою тригонометричних функцій та алгоритмів.

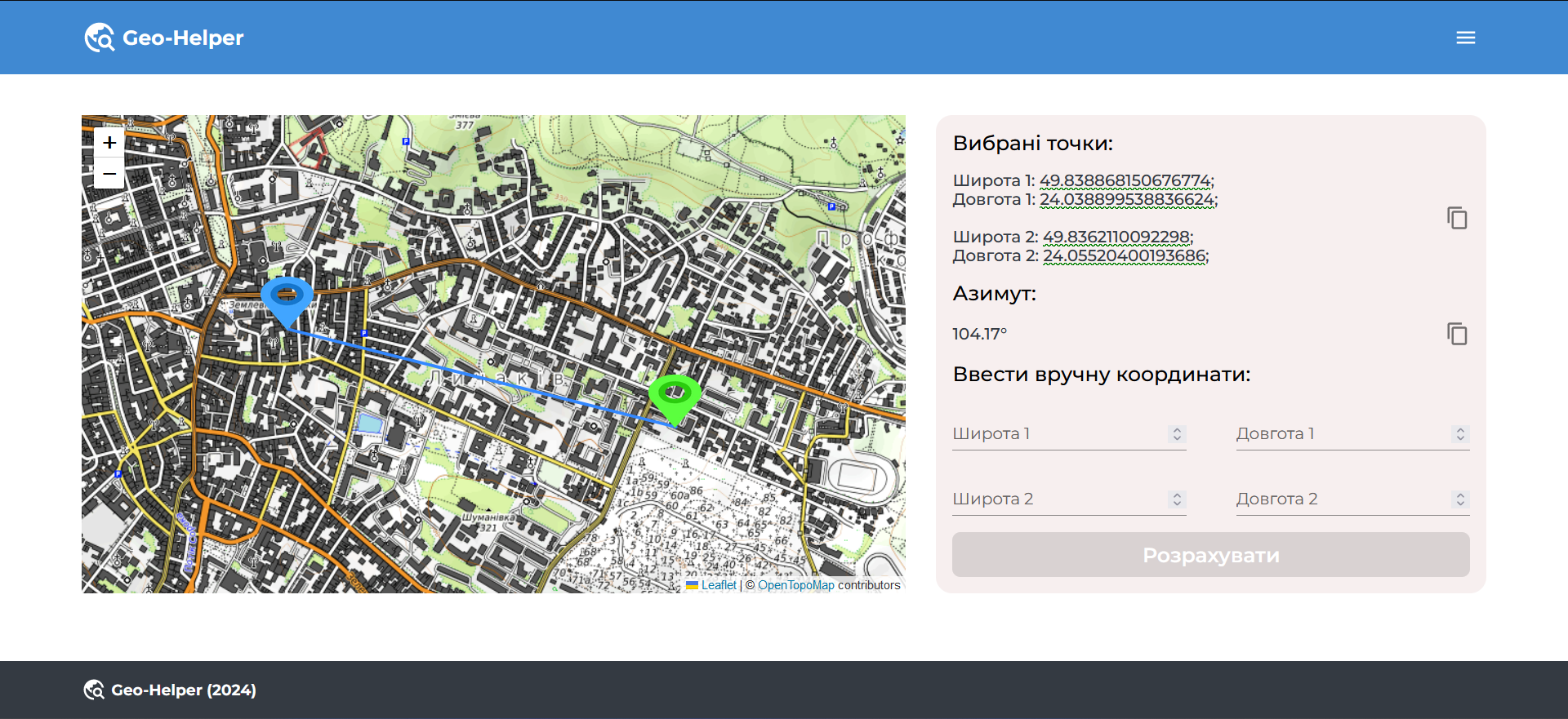


Рис. 3.6 – Обчислення азимуту між двома точками.

Модуль "Конвертор координат" є важливою складовою систем геопросторового аналізу та обробки географічних даних. Основна його мета - забезпечити можливість ефективної конвертації між різними форматами координат, що дозволяє користувачам працювати з даними в їхніх улюблених системах відліку. Давайте розглянемо детальніше, як працює цей модуль та його функціонал:

1. Підтримка різних систем координат: Модуль може опрацьовувати координати у різних системах, таких як Universal Transverse Mercator (UTM), Military Grid Reference System (MGRS) та World Geodetic System 1984 (WGS84). Це означає, що він здатен працювати з широтою та довготою (у форматі WGS84), а також з граничними координатами відносно сітки UTM або MGRS.

2. Пошук інформації про системи координат: Модуль може включати базу даних або таблицю, що містить інформацію про різні системи координат. Ця інформація включає параметри проекції, зони UTM, відомості про геодезичний датум тощо.

3. Автоматизація конвертації: Модуль може надавати можливість автоматичної конвертації координат. Наприклад, користувач може ввести координати у будь-якому форматі, і модуль автоматично визначатиме цей формат і перетворить його в потрібний.

4. Підтримка різних форматів виведення: Модуль може забезпечувати можливість виводу координат у різних форматах. Наприклад, результат може бути представлений у вигляді тексту, таблиці або графічного відображення на карті.

5. Перевірка валідності введених даних: Модуль може включати перевірку на валідність введених координат. Він може перевіряти, чи відповідають вони вимогам певної системи координат або чи не містять помилок.

Ці функції дозволяють модулю "Конвертор координат" забезпечувати зручність та точність при роботі з різними системами координат, спрощуючи процес обробки та аналізу географічних даних.

Модуль "Обчислення відстаней між двома точками" є інструментом, спеціально розробленим для точних розрахунків відстаней між географічними точками на земній поверхні. Основний метод, за яким працює цей модуль, - це використання формули Гаверсінуса, яка дозволяє обчислити відстань між двома точками на сфері за допомогою їхніх географічних координат.

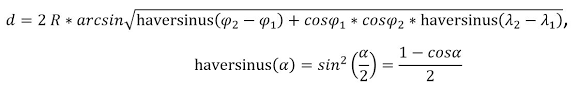


Рис. 3.7 – Формула Гаверсінуса для визначення відстані між точками.

Розглянемо детальніше, як працює цей модуль:

1. Формула Гаверсінуса: Це математична формула, яка використовується для обчислення відстаней між двома точками на сфері за їхніми географічними координатами (широта і довгота). Вона базується на законі сферичної тригонометрії і забезпечує високу точність обчислень відстаней.

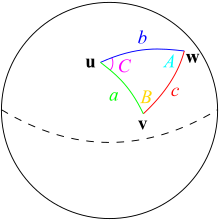


Рис. 3.8 – Приклад використання методу Гаверсінуса.

2. Врахування радіуса Землі: Модуль враховує радіус Землі в своїх обчисленнях. Це важливо, оскільки земна поверхня не є ідеальною сферою, і радіус може змінюватися залежно від місцевих умов. Точне значення радіуса використовується для забезпечення точності обчислень.

3. Врахування широти та довготи: Модуль бере в обробку географічні координати обох точок (широту і довготу) і використовує їх у формулі Гаверсінуса для обчислення відстані між ними. Це дозволяє враховувати кривизну земної поверхні та виконувати точні розрахунки.

4. Високоточність обчислень: Завдяки використанню формули Гаверсінуса та врахуванню радіуса Землі, широти та довготи, модуль забезпечує високоточні обчислення відстаней між двома точками на земній поверхні.

Цей модуль є корисним інструментом для будь-яких застосувань, де потрібно точно визначити відстань між географічними точками, таких як планування маршрутів, навігація, геологічні та геодезичні роботи та багато інших.

Модуль "Визначення висоти заданої точки" розроблений для надання можливості точного визначення висоти будь-якої точки на земній поверхні. Його основна функція - забезпечення точності та надійності вимірювання висоти в межах заданої точки.

Цей модуль зазвичай використовує дані з геодезичних датумів та висотних систем для обчислення висоти точки над еліпсоїдом, геоїдом або іншими референцними поверхнями. Він може використовувати різноманітні методи, включаючи геодезичні рівняння та геодезичні прилади, для досягнення високої точності у визначенні висоти.

Крім того, модуль може бути інтегрований з іншими геопросторовими системами для забезпечення комплексного аналізу географічних даних та вимірювань. Його можна використовувати у різних галузях, включаючи геодезію, картографію, інженерію, геологію та багато інших.

Модуль "Обчислення азимуту між двома точками" призначений для точного визначення кута азимуту між двома географічними точками на поверхні Землі. Азимут - це кутове відхилення від північного напрямку при вимірюванні напрямку на поверхні Землі. У сферичній геометрії азимут визначається як кут між лінією, що з'єднує точку спостереження з північним полюсом та напрямком на об'єкт чи точку, яку спостерігають. Азимут зазвичай вимірюється в градусах, від 0° (на північ) до 360° (знову на північ, проходячи повний коло). Цей концепт широко використовується в навігації, картографії, геодезії та інших географічних науках для визначення напрямку між точками на земній поверхні.

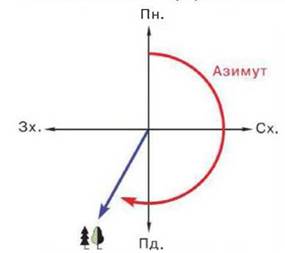


Рис. 3.9 – Азимут.

Основні функції модуля включають:

1. Визначення географічних координат точок: Модуль приймає на вхід географічні координати двох точок (широту і довготу), які ви хочете порівняти.

2. Обчислення азимуту: З використанням географічних координат введених точок, модуль використовує математичні формули для обчислення азимуту. Ці формули зазвичай базуються на сферичній тригонометрії та враховують кривизну Землі.

3. Врахування магнітного відхилення (необов'язково): Деякі реалізації модуля можуть також враховувати магнітне відхилення для визначення магнітного азимуту. Це корисно для навігації з використанням компаса, оскільки вказує напрямок на північ, який відрізняється від географічного північного напрямку.

4. Представлення результатів: Результати обчислення азимуту можуть бути представлені у вигляді числового значення азимуту, або в графічному вигляді на карті, що показує напрямок між двома точками.

5. Використання в різних галузях: Модуль може бути корисним для навігації, геодезії, картографії, а також для визначення напрямку між точками в географічних дослідженнях та інженерних проектах.

Розробка веб-сервісу використовує наступні технології:

Node.js: Забезпечує міцне та ефективне середовище для серверного програмування та розробки.

TypeScript: Забезпечує безперервний JavaScript код з розширеними функціями, перевіркою синтаксису та підтримкою відладки.

API OpenTopoMap: Надає детальні дані картографування для збагачення веб-сервісу екстенсивною топографічною інформацією.

Бібліотека Leaflet: Відображає інтерактивні та динамічні карти з помітною легкістю для поліпшеного користувацького досвіду.

Бібліотека Formik: Керує та перевіряє введення користувача в формах точно, враховуючи всі вимоги користувачів щодо полів введення.



Рис. 3.10 – Приклад використання бібліотеки Formik.

Важливі досягнення у розробці модулів включають досягнення високої точності у обчисленнях відстаней та висот, а також використання перевірених математичних методів для обчислення азимуту та географічних координат. Однією з важливих проблем було забезпечення безперервної інтеграції з зовнішніми джерелами, такими як OpenTopoMap, без виникнення розбіжностей та затримок у даних.

3.2 Тестування розробленого сервісу

Проведено тестування розробленого веб-сервісу з метою перевірки його функціональності, надійності та ефективності. Тестування є критично важливою частиною процесу розробки програмного забезпечення, оскільки воно допомагає виявити та виправити помилки, забезпечити відповідність вимогам та забезпечити задоволення потреб користувачів.

Основні типи проведених тестувань:

1. Функціональне тестування

Функціональне тестування — це тип тестування програмного забезпечення, під час якого система перевіряється на відповідність вимогам і специфікаціям. Функціональне тестування гарантує, що вимоги або специфікації належним чином задовольняються продуктом. Цей тип тестування, у першу чергу, стосується результату обробки

Під час функціонального тестування було перевірено відповідність розробленого сервісу вимогам та очікуванням користувачів. Були перевірені всі основні функції, включаючи конвертор координат, обчислення відстаней та інші.

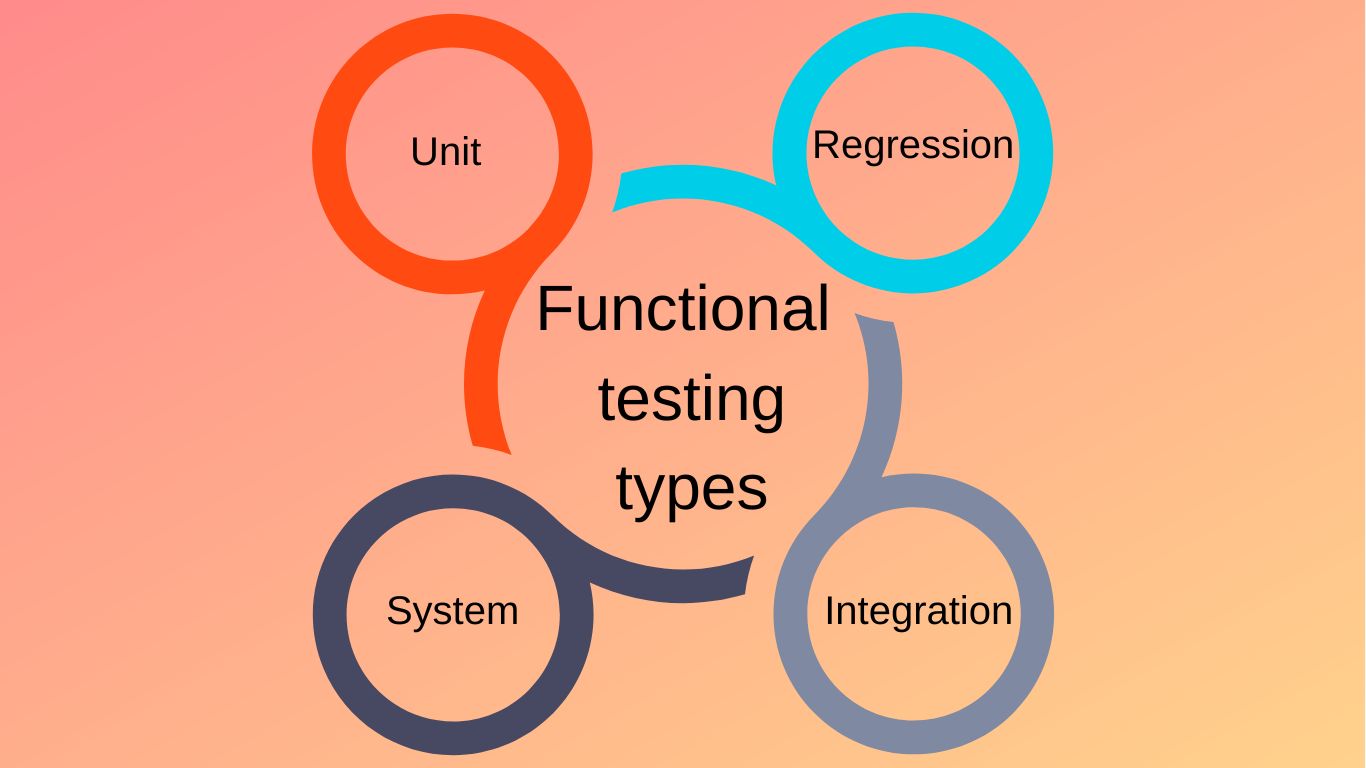


Рис. 3.11 – Функціональне тестування.

2. Тестування на відповідність стандартам безпеки

Тестування на відповідність стандартам безпеки - це процес перевірки програмного забезпечення на відповідність встановленим стандартам, вимогам та нормативам безпеки. Під час цього тестування аналізується, чи дотримується програмне забезпечення встановлених стандартів безпеки, а також чи враховані та реалізовані необхідні заходи для захисту від потенційних загроз та атак.

Це включає в себе такі аспекти, як:

1. Шифрування даних: Перевірка наявності та правильності використання шифрування для захисту конфіденційності даних під час їх передачі та збереження.

2. Захист від атак: Перевірка використання захисних механізмів проти різних типів атак, таких як XSS (Cross-Site Scripting), CSRF (Cross-Site Request Forgery), SQL injection тощо.

3. Аудит доступу: Перевірка наявності системи аудиту доступу, яка веде журнали доступу до системи та дій користувачів для подальшого аналізу та виявлення можливих порушень безпеки.

4. Відповідність нормативам та стандартам: Перевірка відповідності програмного забезпечення встановленим нормативам та стандартам безпеки, таким як ISO 27001, OWASP Top 10, GDPR тощо.

Цей тип тестування допомагає забезпечити високий рівень безпеки програмного забезпечення та захисту конфіденційності, цілісності та доступності даних.

Проведені тестування дозволили переконатися в тому, що система відповідає встановленим стандартам безпеки. Були перевірені заходи безпеки, такі як шифрування даних, захист від атак XSS та CSRF, а також інші заходи.

3. Тестування на навантаження

Тестування навантаження (або навантажувальне тестування) - це вид тестування програмного забезпечення, який проводиться з метою перевірки поведінки системи під час навантаження або в режимі реального часу. Основна ідея полягає в тому, щоб випробувати систему при максимальному або підвищеному навантаженні, щоб виявити можливі проблеми з продуктивністю, масштабованістю та надійністю.

Під час тестування навантаження зазвичай виконуються такі дії:

1. Симуляція реального навантаження: Збільшення кількості користувачів або обсягу запитів, щоб перевірити, як система реагує на велике навантаження.

2. Моніторинг продуктивності: Вимірювання часу відгуку системи та її здатності обробляти запити при різних рівнях навантаження.

3. Тестування меж масштабованості: Визначення максимального обсягу даних або кількості користувачів, яку система може обробляти без значних затримок або відмов.

4. Виявлення проблем з ресурсами: Визначення, як система використовує ресурси, такі як процесор, пам'ять та мережа, під час роботи з великим обсягом даних або користувачів.

Тестування навантаження допомагає виявити слабкі місця в системі, такі як проблеми з продуктивністю, неефективне використання ресурсів або недостатня масштабованість, що дозволяє розробникам вчасно виправити ці проблеми перед впровадженням системи в експлуатацію.



Рис. 3.12 – Тестування навантаження.

Jest - це потужна бібліотека для тестування JavaScript-коду, яка забезпечує зручність та ефективність у створенні, запуску та аналізі тестів. Вона спеціалізується на юніт-тестуванні, тобто тестуванні окремих компонентів програмного забезпечення.

Next.js, як реактивний фреймворк для розробки веб-додатків на основі React, не включає Jest в стандартний комплект поставки, але є дуже сумісним із цією бібліотекою. Next.js дозволяє розробникам безперешкодно використовувати Jest для тестування компонентів та функцій, що забезпечує високу якість коду та його надійність.

Одним з ключових аспектів використання Jest у Next.js є можливість здійснювати автоматичне виявлення та запуск тестів без необхідності встановлення та конфігурування додаткових плагінів чи конфігураційних файлів. Це забезпечує зручність у використанні бібліотеки та прискорює процес розробки шляхом автоматизації тестування.

Завдяки вбудованій підтримці Jest, розробники можуть легко інтегрувати тестування в свій розробний процес без додаткових зусиль. Це дозволяє швидше виявляти та виправляти помилки, підвищує надійність програмного забезпечення та сприяє підтримці високої якості коду.

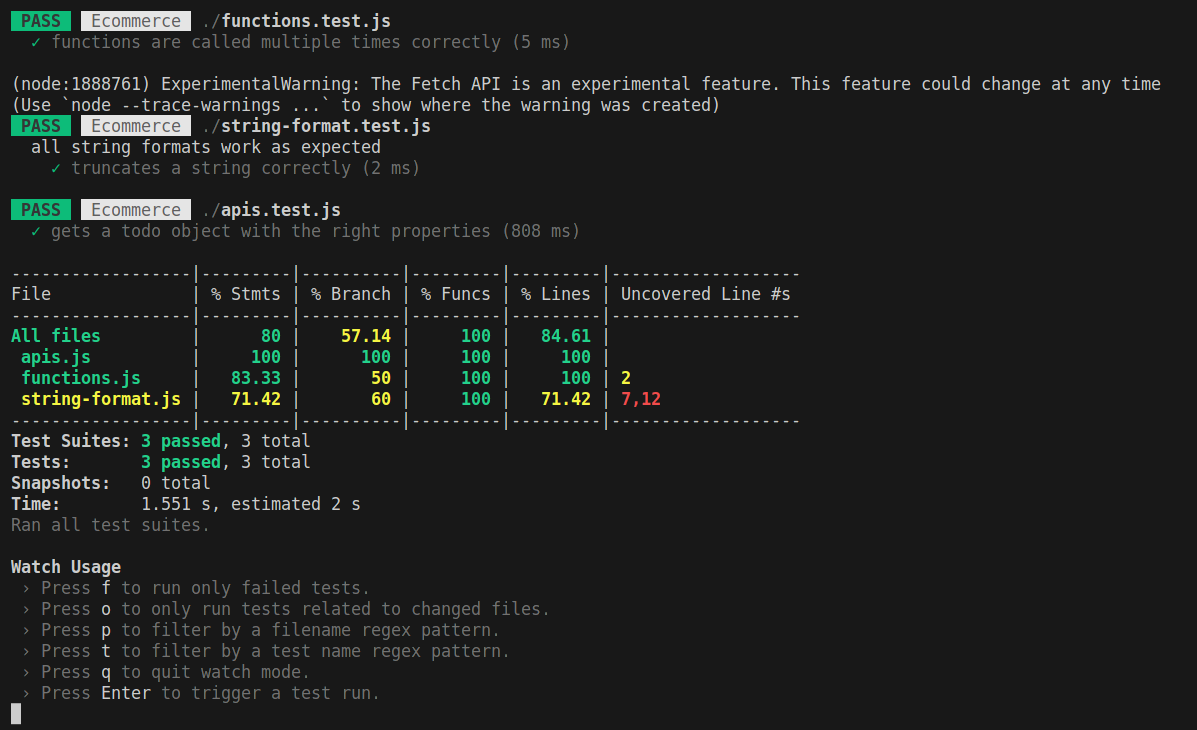


Рис. 3. Тестування системи з використанням бібліотеки Jest

Отже, Jest і Next.js є сумісними технологіями, які разом забезпечують зручне та ефективне тестування веб-додатків на основі React, сприяючи підвищенню якості та надійності програмного забезпечення.

Проведене тестування на навантаження дозволило переконатися в тому, що система може працювати ефективно при різних рівнях навантаження. Були перевірені час відгуку системи та її здатність обробляти одночасні запити великої кількості користувачів.

4. Тестування взаємодії з різними платформами та пристроями: Проведене тестування дозволило переконатися в тому, що веб-сервіс "Geo-Helper" працює на різних платформах та пристроях із різними операційними системами та веб-браузерами без значних проблем.

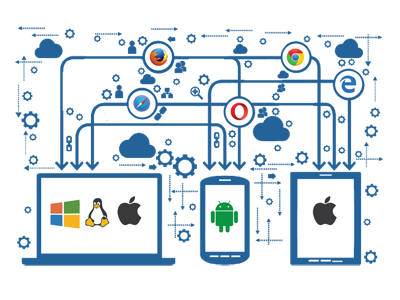


Рис. 3.13 – Тестування взаємодії з різними платформами.

Загальна мета тестування полягала в тому, щоб переконатися в надійності та ефективності розробленого веб-сервісу перед його впровадженням для широкого кола користувачів. Результати тестування дозволили ідентифікувати та виправити потенційні проблеми та забезпечити якість та задоволення вимог користувачів.

3.3 Дейплоймент розробленого Web-додатку

Деплоймент - це процес розгортання розробленого програмного забезпечення на сервері або в хмарному сервісі, щоб зробити його доступним для кінцевих користувачів. Це один з останніх етапів в розробці програмного забезпечення, коли функціонал вже реалізовано та протестовано, і готово до використання.

Процес деплою може включати наступні етапи:

1. Підготовка до деплою: Перш ніж розгорнути програмне забезпечення, необхідно переконатися, що всі необхідні файли та ресурси готові до розгортання. Це може включати компіляцію програмного коду, збирання статичних файлів, підготовку конфігурацій тощо.

2. Вибір середовища для деплою: Вибір середовища для розгортання програмного забезпечення може залежати від потреб проекту та вимог щодо масштабованості, надійності та доступності. Це може бути власний сервер, хмарний сервіс або інший хостинговий сервіс.

3. Конфігурація середовища: Підготовка сервера або хмарного сервісу для прийому програмного забезпечення. Це включає налаштування середовища виконання, баз даних, мережевих налаштувань та інші параметри.

4. Розгортання програмного забезпечення: Завантаження програмного коду та інших необхідних ресурсів на сервер або хмарний сервіс. Цей процес може варіюватися в залежності від вибраного середовища та інструментів для деплою.

5. Тестування після деплою: Після розгортання програмного забезпечення важливо перевірити, чи коректно працює система у новому середовищі. Це може включати тестування функціональності, продуктивності та безпеки.

6. Моніторинг та підтримка: Після успішного деплою необхідно постійно моніторити роботу системи та вчасно реагувати на будь-які проблеми чи помилки. Також може знадобитися регулярне оновлення та підтримка програмного забезпечення.

Отже, деплоймент є важливим етапом в життєвому циклі розробки програмного забезпечення, який забезпечує готовність системи до використання та надійність у реальних умовах експлуатації.

Деплоймент розробленого веб-сервісу є критичною частиною процесу розробки, оскільки визначає доступність та функціональність системи для кінцевих користувачів. В даному випадку для деплою веб-сервісу "Geo-Helper" використовувався сервіс vercel.com, який надає зручні інструменти для швидкого та ефективного розгортання веб-додатків.

Після завершення розробки та тестування функціональності, було прийнято рішення про розгортання сервісу на платформі vercel.com. Цей сервіс забезпечує простий та швидкий процес деплою, дозволяючи розробникам зосередитися на основних аспектах розробки.

Деплоймент на платформі Vercel — це процес розгортання веб-додатків та статичних сайтів, який забезпечує їхню доступність у мережі Інтернет. Vercel надає зручний та швидкий інтерфейс для розгортання, автоматично налаштовує середовище та забезпечує високу продуктивність та стабільність веб-додатків.

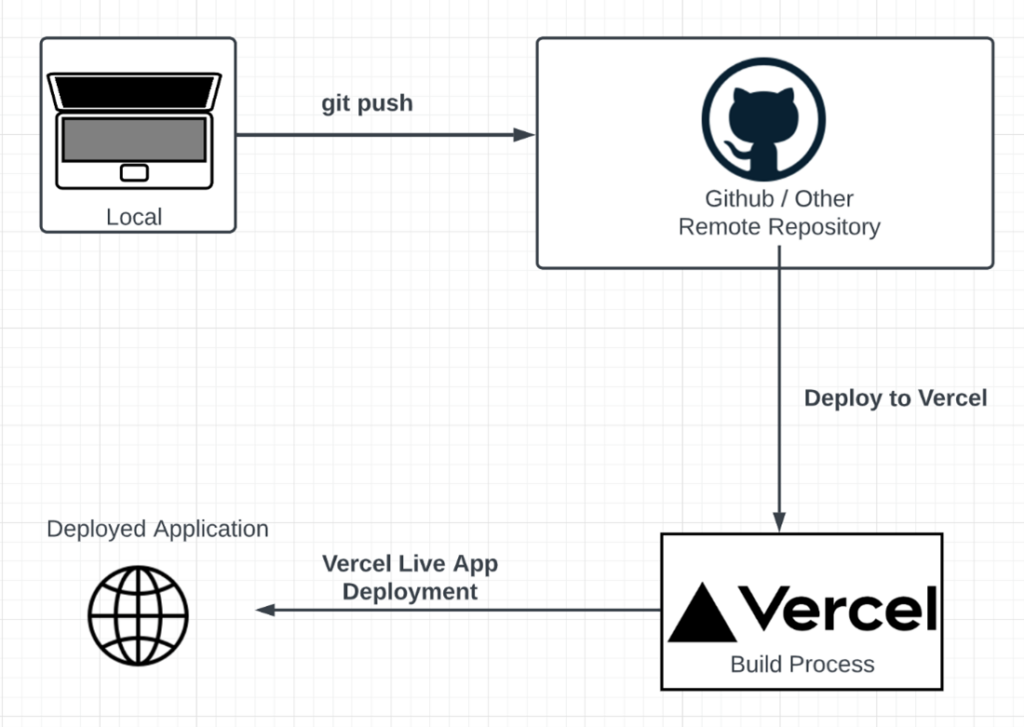


Рис. 3.14 – Тестування взаємодії з різними платформами.

Першим кроком у деплої на Vercel є створення облікового запису на платформі. Після цього необхідно увійти в обліковий запис та створити новий проект. Для цього можна використовувати інтерфейс Vercel або інтегровану команду в терміналі.

Після створення проекту необхідно налаштувати його параметри. Це включає вибір основного каталогу з додатком або сайтом, вибір репозиторію з вихідним кодом (якщо ви використовуєте Git), налаштування середовища виконання (наприклад, Node.js версії), а також налаштування змінних середовища, які можуть знадобитися для деплою.

Після налаштування параметрів проекту можна розпочати процес деплою. Vercel автоматично інтегрується з репозиторієм та може автоматично відслідковувати зміни у вихідному коді. При зміні коду або після вручного запуску деплою, Vercel автоматично стягує оновлений код з репозиторію, компілює або збирає додаток, якщо потрібно, та розгортає його на серверах платформи.

Після успішного завершення деплою Vercel автоматично надає унікальний URL для доступу до додатку або сайту. Крім того, платформа надає різноманітні інструменти для моніторингу та аналізу продуктивності додатків, а також можливість налаштування додаткових параметрів, таких як налаштування домену, HTTPS, автоматичного масштабування та ін.

Отже, деплоймент на платформі Vercel є швидким, зручним та надійним способом зробити ваш веб-додаток або сайт доступним у мережі Інтернет. Платформа надає все необхідне для ефективного розгортання та управління вашими проектами. Процес деплою веб-сервісу "Geo-Helper" на vercel.com був автоматизований та відбувався безперервно, що забезпечило швидке впровадження змін та оновлень в систему. Після успішного завершення деплою, веб-сервіс став доступним для користувачів і готовий до використання.

Отже, завдяки використанню сервісу vercel.com, процес деплою веб-сервісу "Geo-Helper" був простим, ефективним та надійним. Розгортання на цій платформі дозволило швидко зробити систему доступною для користувачів і забезпечило високу стабільність та доступність сервісу.

Висновки до розділу 3

У цьому розділі було проведено реалізацію та тестування розробленого веб-додатку, який має на меті надати користувачам зручні та потужні інструменти для роботи з географічними даними. Реалізація включала розробку модулів та компонентів системи, а також проведення тестування для перевірки функціональності, надійності та продуктивності додатку.

Основні висновки розділу:

1. Опис модулів та компонентів розробленого веб-сервісу: В рамках цього підрозділу було докладно описано всі модулі та компоненти, що входять до складу веб-додатку "Geo-Helper". Кожен модуль був розглянутий окремо, з описом його функціональності та ролі в системі. Цей опис є важливою основою для подальшого розвитку та підтримки системи.

2. Тестування розробленого веб-додатку: Під час тестування було перевірено роботу розробленого веб-додатку з точки зору його функціональності, надійності та продуктивності. Були виконані різноманітні тести, включаючи функціональні тести, тести на безпеку та тести на навантаження. Результати тестування дозволили ідентифікувати та виправити потенційні проблеми та забезпечити високу якість та надійність додатку.

Загалом, розділ 3 був спрямований на опис та виконання всіх необхідних кроків для успішної реалізації та тестування веб-додатку "Geo-Helper". Результати цих процесів будуть використані для подальшого вдосконалення та розвитку системи з метою задоволення потреб користувачів і досягнення їхніх цілей.