

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ ДЛЯ
ТРЕКІНГУ ТУРИСТИЧНИХ МАРШРУТІВ
SOFTWARE IMPLEMENTATION OF THE
TRACKING TOURIST ROUTE APPLICATION

Виконав: студентка 4 курсу, групи КНТ-119сп
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Прокоп'єва Т.А.

(прізвище та ініціали)

Керівник Олійник А.О.

(прізвище та ініціали)

Рецензент Гофман Є.О.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Прокоп'євої Тетяни Андріївни

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація застосунку для трекінгу туристичних маршрутів. Software Implementation of the Tracking Tourist Route Application

керівник проєкту (роботи) Олійник Андрій Олександрович, д.т.н., професор,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “27” квітня 2022 року № 102

2. Строк подання студентом проєкту (роботи) 07 червня 2022 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури програми. 3. Основні рішення щодо реалізації компонентів системи. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Олійник А.О., професор		
Нормоконтроль	Липовець М.В., асистент		

7. Дата видачі завдання “04” квітня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Прокоп'єва Т.А.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Олійник А.О.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
95 с., 55 рис., 8 табл., 4 дод., 15 джерел.

БАЗА ДАНИХ, МОВА ПРОГРАМУВАННЯ, ПРОГРАМНЕ
ЗАБЕЗПЕЧЕННЯ, РЕЙТИНГУВАННЯ МАРШРУТІВ, ТРЕКІНГОВА
СИСТЕМА, ДОПОМОГА МАНДРІВНИКАМ, GIT, REST API, PHP, REACT.

Об'єкт дослідження – процес розробки програмного забезпечення для
трекінгу туристичних маршрутів.

Предмет дослідження – програмне забезпечення для моніторингу та
рейтингування туристичних маршрутів.

Метою роботи є розробка програмного забезпечення для допомоги
мандрівникам.

Матеріали, методи та технічні засоби: мова програмування серверної
частини PHP, мова програмування клієнтської частини JavaScript, React,
система керування базами даних MySQL.

Результати. Розроблено програмне забезпечення для трекінгу
туристичних маршрутів та рейтингування.

Практична цінність роботи полягає у розробці програмного
забезпечення для допомоги мандрівникам у пошуку та відстеженні
туристичних маршрутів.

Висновки. Розроблено програмне забезпечення для трекінгу
туристичних маршрутів.

Галузь використання – користувачі мережі Інтернет, мандрівники та
туристи, фанати активного відпочинку, туристичні гіді.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 95 pages, 55 figures, 8 tables, 4 appendixes, 15 sources.

TRACKING SYSTEM, ROUTE RATING, TRAVEL HELP, PHP, REACT, DATABASE, PROGRAMMING LANGUAGE, SOFTWARE, GIT, REST API.

The object of research is the process of software development for tracking tourist itineraries.

The subject of the research is software for monitoring and rating of tourist routes.

The purpose of this work is to develop software to help travelers.

Materials, methods and technical tools: PHP server-side programming language, JavaScript client-side programming language, React, MySQL database management system.

Results. The software for tracking of tourist routes and rating itineraries have been developed.

The practical value of the work lies in the development of software to help travelers to find and track tourist routes.

Conclusions. The software for tracking tourist routes is developed.

Scope of use – Internet users, travelers and tourists, outdoor activity fans, tourist guides.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	8
1 Аналіз предметної області	9
1.1 Предметна область	9
1.2 Дослідження існуючих аналогів	11
1.2.1 Strava	11
1.2.2 В Похід Карпатами	12
1.2.3 Wikiloc	13
1.2.4 Порівняння існуючих аналогів	14
1.3 Постановка завдання	15
1.3.1 Функціональність системи	16
1.3.2 Вимоги до інтерфейсу	17
1.3.3 Не функціональні вимоги до продукту	17
1.3.4 Усвідомлення середовища користувача	17
1.3.5 Оцінка надійності, продуктивності і підтримки	18
1.4 Висновки за розділом 1	18
2 Розробка архітектури програми	20
2.1 Вибір інструментарію розробки	20
2.1.1 Вибір мови програмування	20
2.1.2 Обґрунтування вибору програмної надбудови	22
2.1.3 Система керування базами даних	23
2.1.4 Розподіл та реалізація бізнес-логіки	24
2.2 Структурна схема розробки	26
2.3 Висновки за розділом 2	27
3 Основні рішення щодо реалізації компонентів системи	28
3.1 Фізичне проєктування	28
3.1.1 Проєктування логіки взаємодії	29
3.2 Проєктування бази даних	33
3.3 Проєктування дизайну	35

3.4 Висновки за розділом 3.....	40
4 Експлуатація, тестування та експериментальне дослідження програми.....	42
4.1 Опис застосунку.....	42
4.2 Умови виконання програми	43
4.3 Інструкція по експлуатації програми	43
4.3.1 Характеристики програми	43
4.3.2 Використані технічні засоби	44
4.3.3 Звернення до програми	44
4.3.4 Вхідні й вихідні дані	45
4.4 Методика та результати тестування	46
4.4.1 Тестування функціоналу	47
4.4.2 Тестування верстки	50
4.4.3 Чек-листи тестування.....	51
4.5 Висновки за розділом 4.....	52
Висновки	54
Перелік джерел посилання	56
Додаток А Технічне завдання	57
Додаток Б Опис програми	63
Додаток В Текст програми	68
Додаток Д Слайди презентації	86

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

SQL – Structured Query Language (мова структурований запитів);

UML – Unified Modeling Language (уніфікована мова програмування);

БД – база даних;

ПЗ – програмне забезпечення;

ОП – оперативна пам'ять;

ОС – операційна система;

СКБД – система керування базами даних.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Предметна область

Тематикою дипломної роботи є вебзастосунок для моніторингу та рейтингування маршрутів, основною метою якого буде допомога мандрівникам.

Наше сьогодення стрімко змінює ставлення громадян до туризму. Сучасна прогресивна молодь активно пропагує здоровий спосіб життя, саморозвиток, заняття спортом, зокрема, туризмом. Охочих подорожувати дедалі стає все більше. При цьому кількість подорожей заради відпочинку (пасивний туризм) зменшується. Натомість отримують поширення подорожі-експедиції, які ставлять метою пізнання культури, історії, природи країни, в якій подорожуєш.

Українці активно долучаються до міжнародного туризму. Але не згасає інтерес до подорожей Україною. Нашій країні є чим привабити допитливих туристів: славетна історія, мальовнича природа, самобутня народна творчість.

Карпати кличуть на зимові розваги, а Асканія-Нова - у заповітні степи. Петриківка запрошує оцінити художній розпис, а Київ і Львів пропонують оглянути об'єкти Всесвітньої спадщини ЮНЕСКО (Собор святої Софії, Києво-Печерська лавра, Львівський ансамбль історичного центру).

Саме для допомоги мандрівникам, побачити свою країну – розроблюється вебзастосунок «Travel Helper». Він допоможе знайти найцікавіші місця та маршрути, а також – розрахувати необхідну кількість провізії на цей шлях.

Велосипедний туризм – це вид туризму, у якому основним чи єдиним засобом пересування є велосипед. Під поняттям «велосипедний туризм» мають на увазі і один із видів активного відпочинку, і різновид спортивних подорожей [1].

Велотуризм слід відрізняти від велопоходів. Різниця в тому, що велотуризм передбачає проходження на велосипеді маршрутів певної

складності (всього їх налічується 6 штук) заради спортивних досягнень і розрядів, а велопоходи – це, скоріше, поєднання приємного з корисним: тут вам і спортивна складова і огляд культурних й природних пам'яток [2].

Існує декілька стилів велотуризму, але всі вони мають одну спільну рису – подолання перешкод за допомогою велосипеда.

Класичний велосипедний туризм – це справжнісінький туризм: з наметами, багаттями та гітарами – тільки маршрут велосипедний, а не піший. Швидкість пересування по маршруту залежить від того, яка зібралася група. Даний вид велосипедного туризму можливий і для походів вихідного дня і для тижневих турів.

Цивілізований велотуризм – найпростіше, що можна придумати. Та, мабуть, найнудніше. Звичайна велосипедна прогулянка на свіжому повітрі на відносно велику відстань. Проїхавши 50-70 кілометрів, можна зупинитися в готелі і присвятити вечір дозвілля. Походів такого роду зараз існує багато, особливо за кордоном, де за своїм бажанням можна підібрати цікавий маршрут. Для тих велосипедистів, котрі не горять бажанням жити в наметі і готувати їжу на багатті.

Експедиційний велосипедний туризм – вигаданий та втілений на практиці людьми, для яких першочергові цілі – подолати перешкоди. Однак про навколишні пейзажі та пам'ятки вони також не забувають. Найкращою винагородою для подібних мандрівників є пройдений маршрут. І чим складніший він був, тим краще.

Швидкий велотуризм – такий напрям в велосипедному туризмі підійде професіоналам, які звикли постійно відчувати своє тіло, він, звісно, припадає до душі. Головне для гонщика в цьому виді велотуризму — подолання відстаней. Велосипедист, який не обтяжений вантажем, за день проходить сотні кілометрів, а ночі проводить в готелях. Такий велопохід відрізняється дійсно величезними відстанями й спеціальними шосейними велосипедами.

Екстремальний велосипедний туризм – цей вид велотуризму пов'язаний в основному з появою МТВ (mountain bike). Як правило, для такого

туризму маршрут вибирається досить складний — в ідеалі це гори, важкопрохідні ліси і болота. Основа екстремального велотуризму — це поєднання рівних доріг, бездоріжжя і різних пам'яток [3].

1.2 Дослідження існуючих аналогів

Наразі, вебзастосунки такого типу дуже поширені, але в основному надають неповний обсяг функціоналу безкоштовно. Основою таких застосунків є пошук маршрутів, але не всі застосунки мають функції фільтрування та рейтингування. Тому час на пошук потрібного маршруту дуже подовжується. Існуючі трекінгові застосунки мають як недоліки, так і переваги.

Недоліки:

- відсутність можливості фільтрації маршрутів;
- відсутність аналітики досвіду користувача;
- відсутність функції розрахунку необхідного продовольства;
- незначна кількість безкоштовних можливостей.

Переваги:

- наявність коментарів та чатів;
- наявність мобільного застосунку з функцією GPS.

Підбиваючи підсумки можна сказати, що функція аналітики досвіду користувача – буде дуже корисною, тому що скорочує час на пошук маршрутів.

1.2.1 Strava

Strava - це мобільний застосунок і сайт для відстеження спортивної активності через GPS. Девіз сервісу: «соціальна мережа для спортсменів» [4].

Застосунок має такі можливості, як пошук маршрутів і спортсменів у базі даних. Спортсмени можуть рухатися один за одним, дані будуть

автоматично групуватися, якщо їх рух відбувається одночасно та в тому ж місці (наприклад, участь в організованих марафонах або групових забігах/заїздах). Спортсмени можуть виявляти «пошану» (аналогічно до вподобань у соціальній мережі Facebook) і коментувати активність один одному, а також додавати фотографії до своєї активності. Головну сторінку застосунку представлено на рис.1.1.

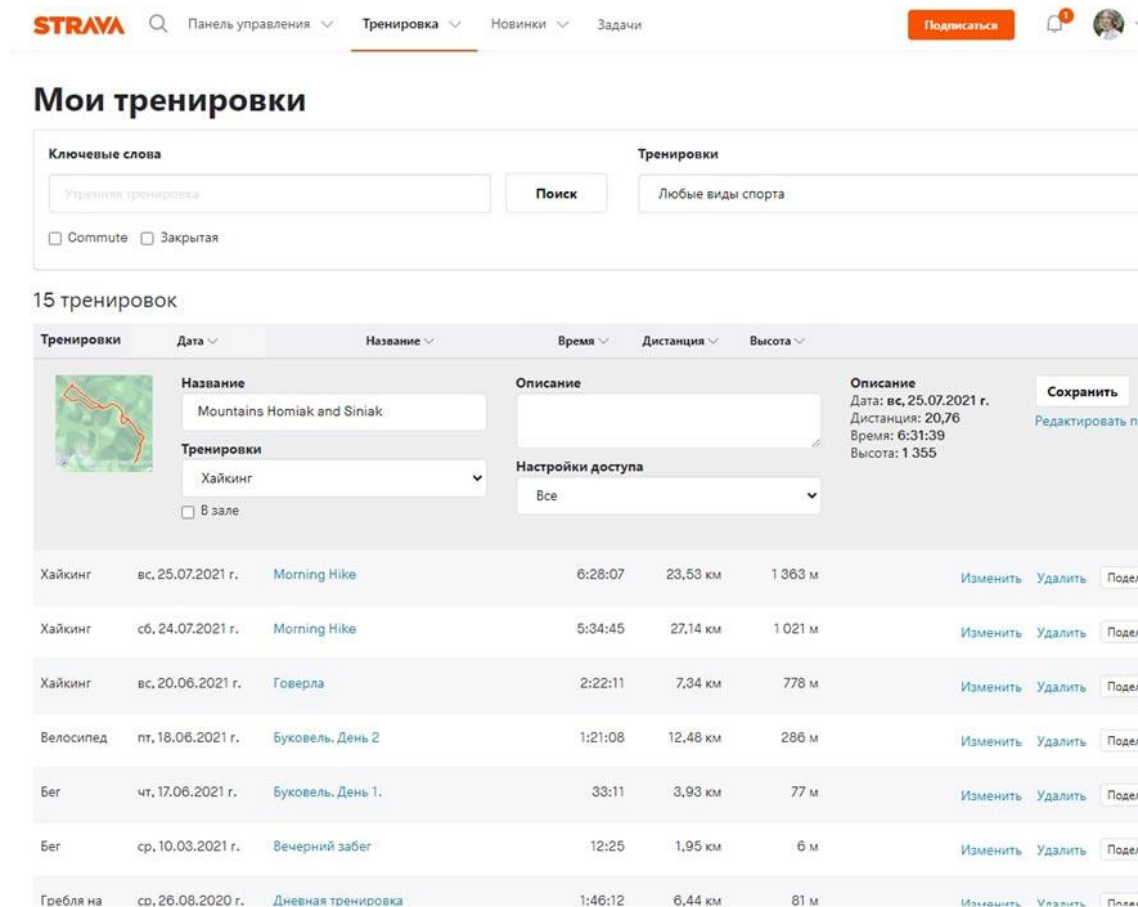


Рисунок 1.1 – Сервіс Strava

1.2.2 В Похід Карпатами

Цей вебзастосунок створено для любителів пішого туризму в Карпатах. Мета сайту - зібрати всю інформацію, яка може знадобитися туристам на одному ресурсі та надати зручний доступ до неї. Головну сторінку застосунку «В Похід Карпатами» зі списком пішохідних маршрутів Карпат представлено на рис.1.2.

Вебзастосунок має такі функції:

- інтерактивна карта Карпат. З інформацією про райони, маршрути походів, місця, населені пункти;
- база даних інформації. Маршрути, місця (джерела, таборування, колиби, цікаві місця, вершини та інше), села та міста. Пошук по цій базі;
- прогноз погоди для місць, маршрутів та населених пунктів;
- спілкування та обговорення у форумі та чаті [5].

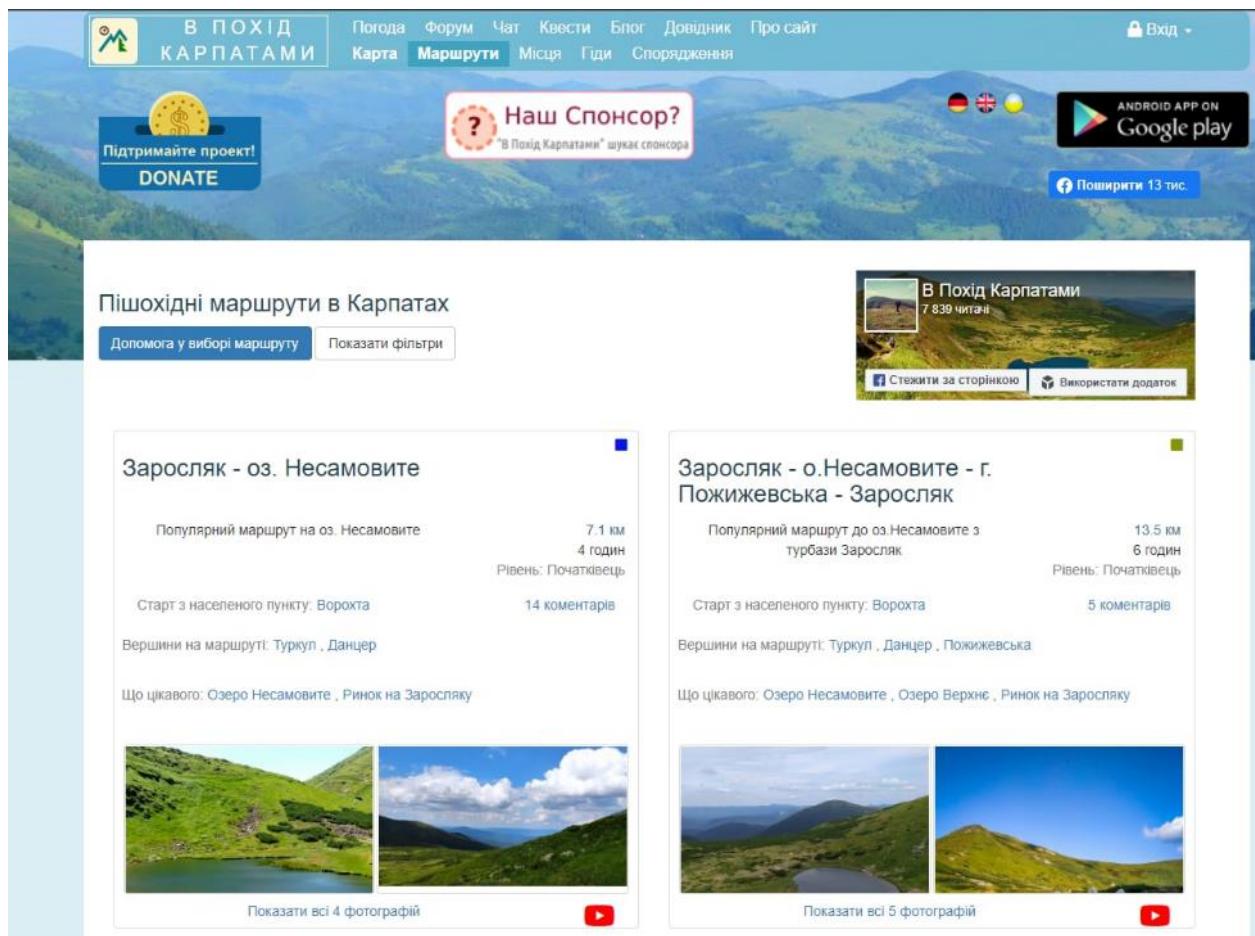


Рисунок 1.2 – Сайт «В Похід Карпатами»

1.2.3 Wikiloc

Wikiloc - це гібридний вебзастосунок, де можна зберігати та обмінюватися геоприв'язаними зовнішніми маршрутами (особливо з GPS) та точками інтересу (шляховими точками) з усього світу. Маршрути відносяться

до різних категорій залежно від типу подорожі, включаючи піші, велосипедні та багато інших видів діяльності. Маршрути відображаються на картах Apple Maps, OpenStreetMap та інших картографічних картах. У сервісу є мобільні додатки для iPhone та Android [6]. Головну сторінку застосунку «Wikiloc» зі списком велосипедних маршрутів України представлено на рис.1.3.

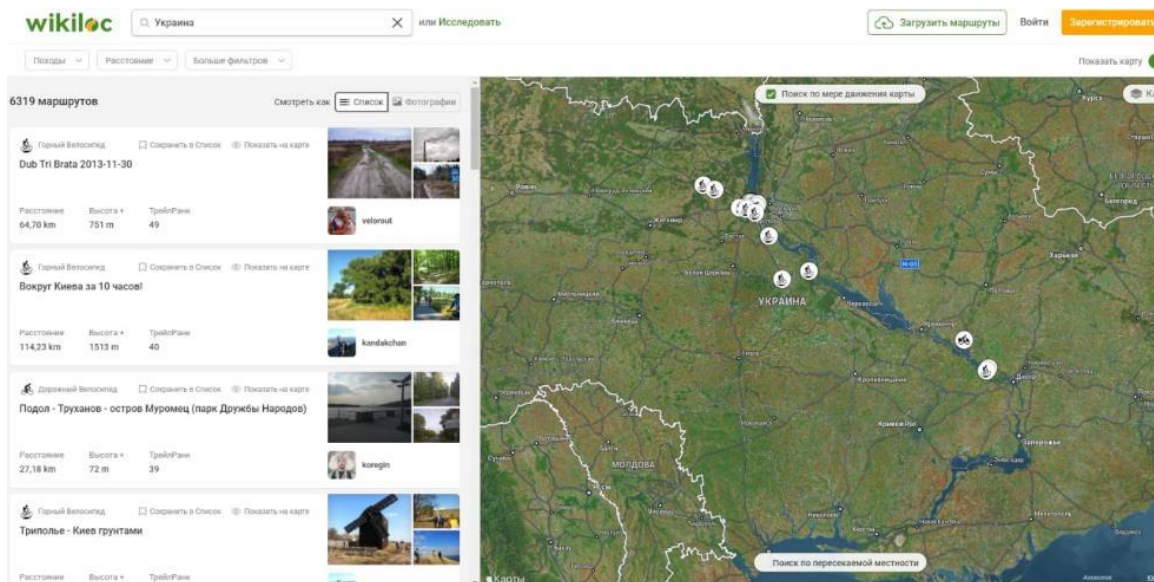


Рисунок 1.3 – Вебзастосунок «Wikiloc»

1.2.4 Порівняння існуючих аналогів

Проведемо порівняння за певними критеріями, хід якого представимо у вигляді таблиці (табл. 1.1). У таблиці будуть порівнюватися попередні аналоги із застосунком, що розробляється – «Travel Helper». Серед критеріїв виділимо наступні:

- фільтрація маршрутів – функція, яка дозволяє продивитися маршрути за конкретним критерієм, наприклад, за країною або містом, за оцінкою користувачів, за складністю;
- аналітика та рейтингування маршрутів – функція, яка дозволяє продивитися маршрути які найбільш підходять для конкретного користувача. Рейтинг базується на розташуванні користувача, його рівні підготовки та минулому досвіді;

- калькуляція продовольства – функціонал, який допомагає розрахувати кількість необхідного запасу їжі для конкретного походу;
- безкоштовний доступ – доступ до функціоналу, без додаткової оплати послуг.

Таблиця 1.1 – Порівняння існуючих аналогів

Критерій порівняння	Strava	В Похід Карпатами	Wikiloc	Мій застосунок
Фільтрація маршрутів	-	+	+	+
Аналітика, рейтингування маршрутів	-	-	-	+
Калькуляція продовольства	-	-	-	+
Безкоштовний доступ	-	+	-	+

З результатів порівняння можна зробити наступні висновки. По-перше, майже всі додатки потребують додаткової оплати своїх послуг. Більш того, не всі сайти мають фільтрацію та рейтингування маршрутів. Також жоден аналог не має функції розрахунку продовольства.

Саме тому актуальною залишається задача розробки нового застосунку «Travel Helper» із привабливим та сучасним дизайном, високою швидкістю роботи та достатнім рівнем усіх показників. А головне - цей застосунок надає унікальну можливість аналітики досвіду користувача.

1.3 Постановка завдання

За результатами аналізу проблемної області та дослідження існуючих аналогів для вирішення проблемних питань, сформулюємо постановку задачі.

Метою є розробка нового застосунку «Travel Helper». Система повинна реалізувати функціональні вимоги:

- реєстрація та авторизація в системі;

- пошук маршрутів;
- перегляд повного списку маршрутів;
- перегляд рекомендованих маршрутів;
- додавання (редагування/видалення) особистих маршрутів;
- рейтингування маршрутів;
- фільтрація маршрутів;
- калькуляція розрахунків запасів їжі.

Система також повинна бути стійкою до збоїв, працювати як на десктопах, так і на мобільних пристроях, бути адаптованою під різні розширення екранів і легкою для подальшої підтримки і масштабування.

1.3.1 Функціональність системи

Межі системи: система повинна працювати постійно без збою.

Вимоги до функціональних характеристик.

Робота із застосунком – не зареєстрований користувач:

- реєстрація;
- перегляд повного списку маршрутів;
- фільтрація маршрутів по доступним критеріям.

Робота із застосунком – зареєстрований користувач:

- реєстрація та авторизація;
- перегляд повного списку маршрутів;
- перегляд рекомендованих маршрутів;
- фільтрація маршрутів по доступним критеріям;
- можливість додати/редагувати/видалити власні маршрути;
- можливість ставити оцінки маршрутам;
- можливість додавати маршрути в обране;
- калькуляція провізії на певні маршрути.

1.3.2 Вимоги до інтерфейсу

Для забезпечення ергономічності, зручності та зрозумілості інтерфейс розроблюваного застосунку повинен бути інтуїтивно зрозумілим, простим та сучасним. Також, застосунок має працювати без перебоїв на десктопах та мобільних пристроях.

1.3.3 Не функціональні вимоги до продукту

Мінімальні системні вимоги та програмне забезпечення, що необхідні для функціонування серверної частини мобільного застосунку:

- операційна система: Linux;
- процесор з двома ядрами і з тактовою частотою не менше 2 ГГц;
- оперативна пам'ять: 8 Гбайт;
- встановлене програмне забезпечення Docker;
- дисковий простір: SSD 25Гбайт.

Для експлуатації клієнтської частини необхідні такі програмно-технічні засоби:

- сучасний браузер (Google Chrome, Mozilla Firefox, Safari);
- стабільне підключення до мережі Інтернет.

1.3.4 Усвідомлення середовища користувача

Застосунок повинен надавати інформацію про туристичні маршрути, яка може знадобитися туристам на одному ресурсі та надати зручний доступ до неї.

Для зареєстрованих клієнтів – будуть доступні такі функції:

- пошук та перегляд вже існуючих маршрутів;
- додавання, редагування та видалення особистих маршрутів;
- аналітика та рейтингування маршрутів;

- фільтрація маршрутів;
- калькуляція розрахунків запасів їжі.

Основною перевагою є аналітичний модуль - яка дозволяє продивитися маршрути, які найбільш підходять для конкретного користувача. Рейтинг базується на розташуванні користувача, його рівні підготовки та минулому досвіді.

Кожен бажаючий може переглядати повний список маршрутів та статті. Але не може коментувати та додавати в особисті.

1.3.5 Оцінка надійності, продуктивності і підтримки

Надійне функціонування розробки повинно бути забезпечене виконанням сукупності організаційно-технічних заходів, перелік яких наведений нижче:

- захист персональних даних користувачів;
- використання ліцензійного програмного забезпечення;
- обробка помилок та виведення їх у зрозумілому для користувача вигляді;
- перевірка введених даних як на стороні клієнта, так і на стороні сервера
- стійкість до збоїв.

1.4 Висновки за розділом 1

В першому розділі було розглянуто основні поняття, що стосуються сервісів для допомоги мандрівникам. У тому числі було проведено порівняння вже існуючих аналогів із розроблюваним програмним продуктом.

У ході аналізу було виявлено, що подібні сервіси розповсюджені, але в основному не мають безкоштовного доступу.

Було сформовано функціональні та не функціональні вимоги до розроблюваного додатку, переглянуто вимоги до користувацького інтерфейсу, а також проведена оцінка надійності та продуктивності майбутнього програмного продукту.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

В попередньому розділі відбулося дослідження предметної області. Відбулася фіксація існуючих рішень та їх переваг і недоліків серед яких: відсутність можливості фільтрації маршрутів та аналітики досвіду користувача, нестача функції розрахунку необхідного продовольства, наявність коментарів та чатів, та можливість синхронізування з мобільними застосунками з функцією GPS. За результатами представленого аналізу були виявлені: межі системи, вимоги до інтерфейсу, не функціональні вимоги до продукту, усвідомлення середовища користувача та проведена оцінка надійності, продуктивності і підтримки.

Надалі необхідно визначитися із технологією, технікою та принципами реалізації вебзастосунку для моніторингу та рейтингування маршрутів.

2.1 Вибір інструментарію розробки

2.1.1 Вибір мови програмування

Для написання серверної частини застосунку було обрано мову програмування PHP. Основними перевагами цієї мови програмування являється ефективність, гнучкість, швидкість виконання коду та підтримка великої кількості баз даних.

PHP – скриптова мова загального призначення, інтенсивно застосовується для розробки вебзастосунків. В даний час підтримується переважною більшістю хостинг-провайдерів і є одним з лідерів серед мов, що застосовуються для створення динамічних вебсайтів [7].

Для створення розмітки сторінок застосунку була використана штучна комп'ютерна мова HTML – це мова опису структури сторінок документів, яка дозволяє звичайний текст формувати в абзаци, заголовки, списки та інші структури, створювати посилання на інші сторінки. Це текстова мова, в якій інструкції з форматування, що називаються тегами, вбудовані в розділи

документа, які містять конкретну інформацію. Теги повідомляють браузерам, як формувати і представляти інформацію на екрані.

Для написання клієнтської частини застосунку було обрано мову програмування JavaScript – це крос-платформна, об'єктно-орієнтована скриптова мова, яка є невеликою і легковажною. Всередині виконання JavaScript може бути пов'язаний з об'єктами даного середовища і надавати програмний контроль над ними.

Як мінімум, три сильні сторони JavaScript:

- повна інтеграція з HTML/CSS;
- прості речі робляться просто;
- підтримується всіма основними браузерами та включений за замовчуванням [8].

JavaScript - це єдина браузерна технологія, що поєднує всі ці три речі.

Проведемо порівняння мов програмування для реалізації вебзастосунків, результати порівняння представимо в таблиці 2.1. У якості критеріїв будемо використовувати наступні:

- продуктивність – швидкість роботи;
- вимогливість до ресурсів системи – над-використання об'єму оперативної пам'яті;
- зрозумілість синтаксису – легкість читання написаного коду.

Таблиця 2.1 – Порівняння мов програмування

Критерій	Java	Python	PHP
Продуктивність	+	–	++
Вимогливість до ресурсів системи	+	–	+
Зрозумілість синтаксису	+	–	+

Таким чином, для реалізації серверної частини застосунку було обрано мову PHP, а для створення клієнтської – JavaScript.

2.1.2 Обґрунтування вибору програмної надбудови

Laminas Framework – це об'єктно-орієнтований PHP фреймворк для розробки вебзастосунків, розроблений та підтримуваний компанією Zend.

Платформа базується на принципах MVC з усіма перевагами та недоліками. Усі компоненти Laminas Framework написані на об'єктно-орієнтованому коді PHP 5, тут реалізовано дуже гнучку архітектуру зв'язування. Це означає, що окремі частини проєкту матимуть мінімальні залежності одна від одної. Важливою перевагою даного фреймворку є універсальність – він сумісний з усіма актуальними системами керування базами даних, включаючи MySQL, MariaDB, Oracle Database, IBM DB2, Microsoft SQL Server, PostgreSQL, SQLite та Informix. Laminas Framework вважається одним з кращих рішень для тих, хто віддає перевагу об'єктно-орієнтованому програмуванню при розробці сайтів та ПЗ [9].

Порівняльна характеристика найпоширеніших php-фреймворків наведена у таблиці 2.2.

Серед критеріїв порівняння будемо розглядати наступні:

- підтримка – документація, кількість статей з прикладами коду та спільнота;
- частота оновлення – оновлення фреймворку корисними функціями або виправленнями помилок;
- об'єктно-орієнтований підхід (ООП) розробки – структура системи описується в термінах об'єктів і зв'язків між ними, а поведінка системи описується в термінах обміну повідомленнями між об'єктами.

Таблиця 2.2 – Порівняння php-фреймворків

	Yii2	Laravel	Symphony	Laminas
Підтримка	–	–	+	+
Частота оновлення	–	+	+	++
ООП	+	–	–	+

З порівняння наведеного в таблиці можна зробити висновок, що Laminas Framework є найкращим варіантом для розробки серверної частини застосунку.

Через те що JavaScript – динамічно типізована мова програмування, було вирішено використовувати надбудову TypeScript для стеження за типізацією.

TypeScript – це різновид популярної мови програмування JavaScript, яка додає деякі ключові функції, важливі для розвитку застосунків. Він має ряд переваг:

- через статичну типізацію код на TypeScript більш передбачуваний і тому його легко дебажити;
- завдяки гарній підтримці ООП, модулів та просторів імен, проєкт, навіть при великому об'ємі, буде впорядкованим;
- компілятор на стадії збирання знаходить багато помилок, ще до того як вони потраплять в рантайм і зламують щось.

Однією з найпримітніших особливостей TypeScript є статична типізація. Це означає, що тип змінних вказується явно, а компілятор слідкує щоб розробник дотримувався написаного. Якщо ж при оголошенні змінної розробник не вкаже її тип, компілятор зробить це сам, в залежності від того, яке значення ви привласните змінній вперше [10].

2.1.3 Система керування базами даних

Система керування базами даних — це комплекс програм, що забезпечує введення, зберігання, пошук, опрацювання даних у базі даних.

Для управління базою даних у ході роботи було використано MySQL. Серед переваг можна виділити такі як: швидкість, безпека, надійність та зручна взаємодія з серверною мовою PHP.

MySQL – компактний багатопотоковий сервер баз даних. Характеризується високою швидкістю, стійкістю і простотою використання.

MySQL вважається гарним рішенням для малих і середніх застосунків.

Можливості сервера MySQL:

- простота у встановленні та використанні;
- підтримується необмежена кількість користувачів, що одночасно працюють із БД;
- кількість рядків у таблицях може досягати 50 млн;
- висока швидкість виконання команд;
- наявність простої і ефективної системи безпеки [11].

2.1.4 Розподіл та реалізація бізнес-логіки

Загальна схема застосунку представлена рис. 2.1.

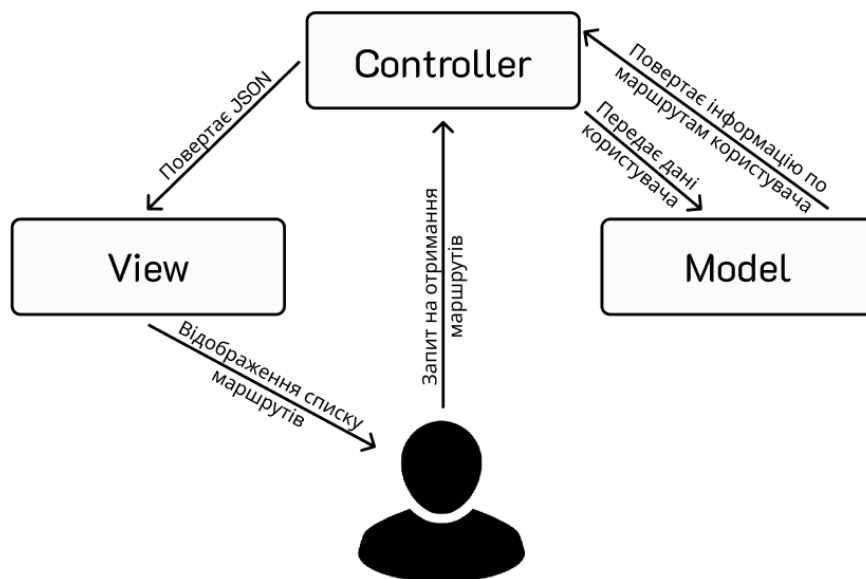


Рисунок 2.1 – Схема застосунку «Travel Helper»

Для представлення бізнес-логіки використаємо шаблон Model-View-Controller.

Model-View-Controller (MVC) – схема поділу даних програми, призначеного для користувача інтерфейсу і керуючої логіки на три окремих компоненти: модель, уявлення і контролер - таким чином, що модифікація кожного компонента може здійснюватися незалежно.

React – одна з бібліотек JavaScript з відкритим кодом. Він використовується для побудови інтерактивних інтерфейсів користувача. Це ефективна, декларативна та гнучка бібліотека. Він стосується компонента V (View), тобто перегляду модельного перегляду-контролера (MVC).

Model – Модель, це частина програми, яка визначає його основну функціональність як безліч абстракцій. У моделі можна визначити процедури доступу до даних та бізнес-логіку. Тобто, модель отримує з бази даних інформацію, потрібну для виконання конкретних запитів. До моделі відноситься Laminas Framework з ORM Doctrine для обробки запитів до БД.

Controller - Контролери пов'язують весь патерн у єдине ціле. Вони керують моделями, на основі введених даних користувачем та інших факторів, вирішують який з уявлень відобразити, передають уявленню необхідні дані, або повністю передають управління іншому контролеру. Роль контролерів виконує Laminas Framework, запити виконуються через програмний інтерфейс застосунку (API).

Основна перевага компонентів полягає в тому, що зміна будь-якого одного компонента не впливає на всю програму. На рисунку 2.2 наведено архітектуру застосунку.

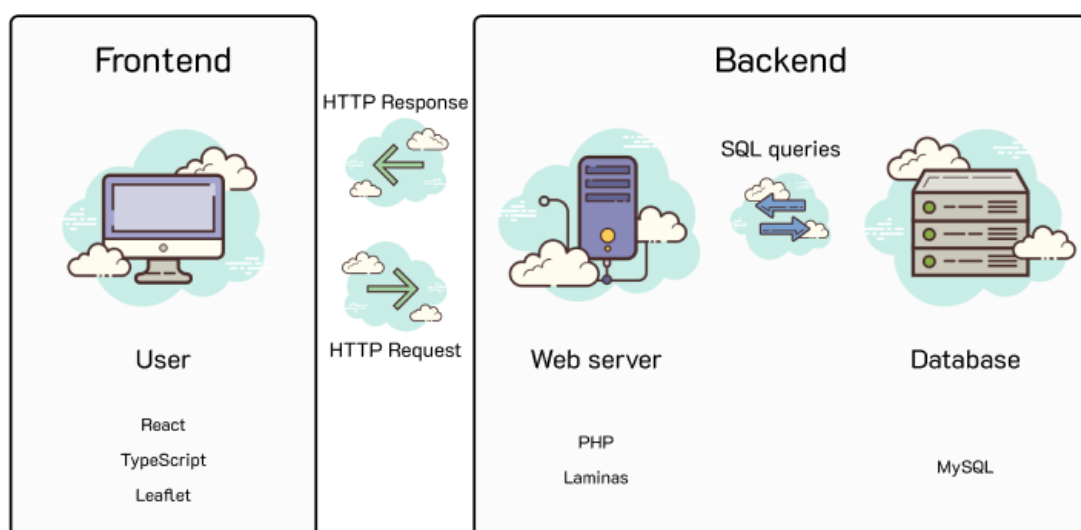


Рисунок 2.2 – Архітектура застосунку

2.2 Структурна схема розробки

Структурно застосунок поділений на серверну частину, в тому числі API для роботи з базою даних, клієнтську частину та утиліти для роботи з мапою.

На рисунку 2.3 наведено структурну схему ПЗ, що розробляється.

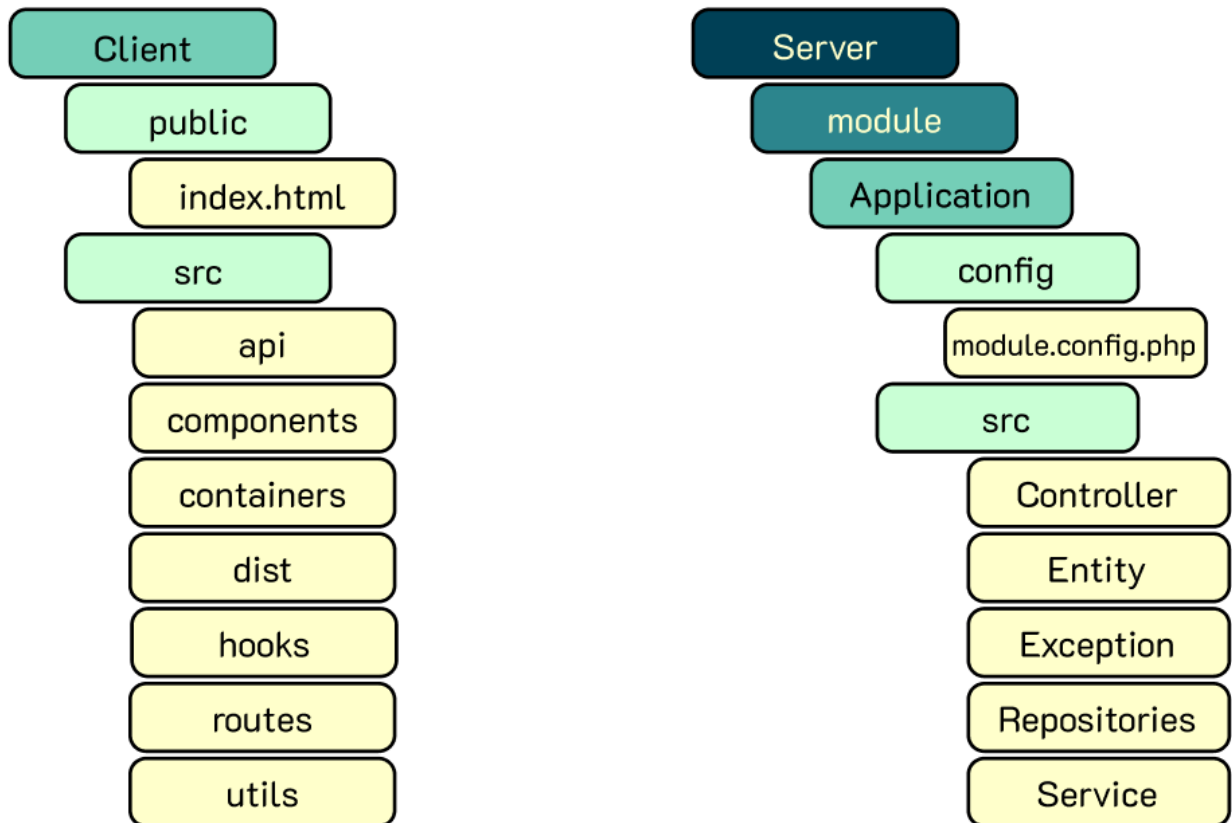


Рисунок 2.3 – Загальна структурна схема системи

У таблиці 2.3 наведено опис модулів реалізованого додатку з визначенням їх основних функцій.

Загалом було виділено 5 модулів для застосунку, що буде розроблятися, поділ на такі модулі можна пояснити закінченою функціональністю кожного. Тобто в майбутньому при масштабованості функціоналу застосунку не буде потреби оновлювати його повністю, а лише оновити конкретний модуль

Таблиця 2.3 – Опис модулів реалізованого застосунку з визначенням основних функцій

Модуль	Опис
Авторизація	Використовується для авторизації користувачів у системі
Реєстрація	Використовується для реєстрації нових користувачів у системі
База даних	Використовується для роботи з базою даних
Фронтенд	Використовується для роботи користувача з системою
Leaflet	Використовується для роботи користувача з мапою

2.3 Висновки за розділом 2

У другому розділі було зазначено особливості обраної мови програмування, їх надбудови та системи керування базами даних (СКБД). Також було наведено структурну схему додатку і розглянуто основні модулі розроблюваної системи.

Для реалізації серверної частини було обрано мову програмування PHP, через її ефективність, гнучкість, швидкість виконання коду та підтримка великої кількості баз даних. У якості програмних надбудов було обрано Laminas Framework, через його універсальність та об'єктно-орієнтований підхід програмування.

Для реалізації клієнтської частини було обрано мову програмування JavaScript. З такими надбудовами: React, TypeScript та Material-ui, через їх ефективність та гнучкість.

У якості патерну було обрано MVC.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Фізичне проєктування

Структуру вебзастосунку для моніторингу та рейтингування маршрутів подамо у вигляді структурних діаграм в нотації UML, зокрема діаграми класів, пакетів та компонентів (рис. 3.1 – 3.3).

Загальна діаграма класів програмного забезпечення для моніторингу та рейтингування маршрутів зображена на рис. 3.1.

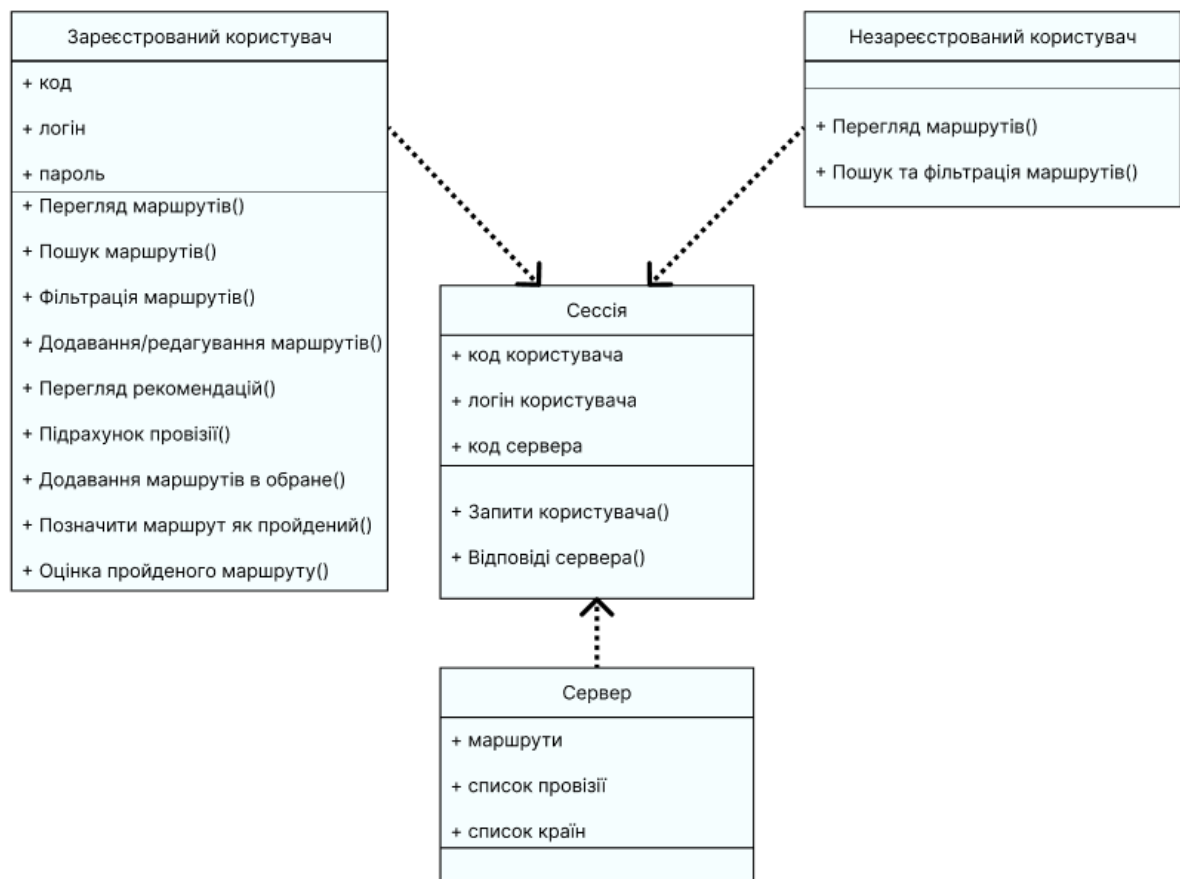


Рисунок 3.1 – Діаграма класів програмного забезпечення для моніторингу та рейтингування маршрутів у нотації UML

Загальна діаграма пакетів програмного забезпечення для моніторингу та рейтингування маршрутів зображена на рис. 3.2.



Рисунок 3.2 – Діаграма пакетів ПЗ

Загальна діаграма компонентів програмного забезпечення для моніторингу та рейтингування маршрутів зображена на рис. 3.3.

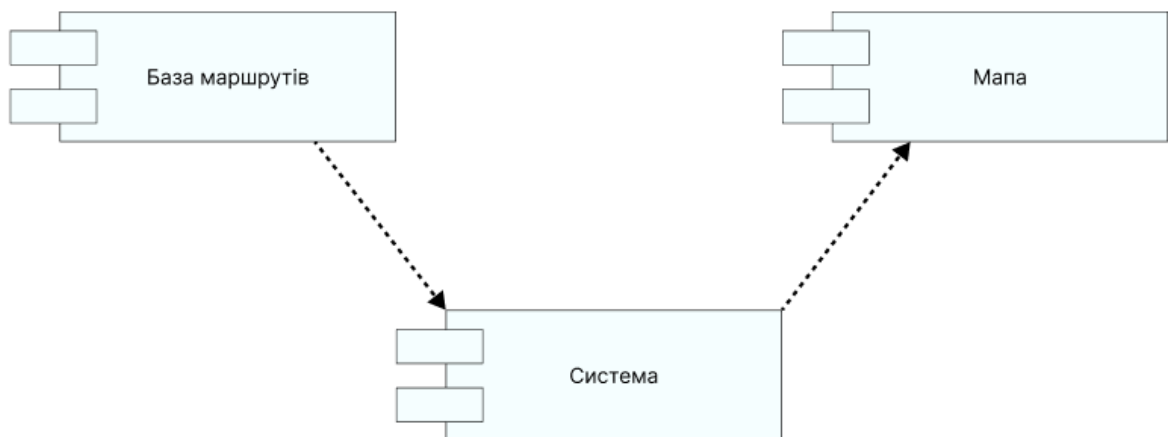


Рисунок 3.3 – Діаграма компонентів ПЗ для моніторингу та рейтингування маршрутів у нотації UML

3.1.1 Проєктування логіки взаємодії

Функціонування застосунку для моніторингу та рейтингування маршрутів подамо за допомогою абстрактних моделей у нотації UML основі діаграм поведінки.

В процесі розробки програмного застосунку моніторингу та рейтингування маршрутів на основі діаграм поведінки були створені діаграми прецедентів, станів, діяльності системи.

Загальна діаграма прецедентів ПЗ для моніторингу та рейтингування маршрутів зображена на рис. 3.4.

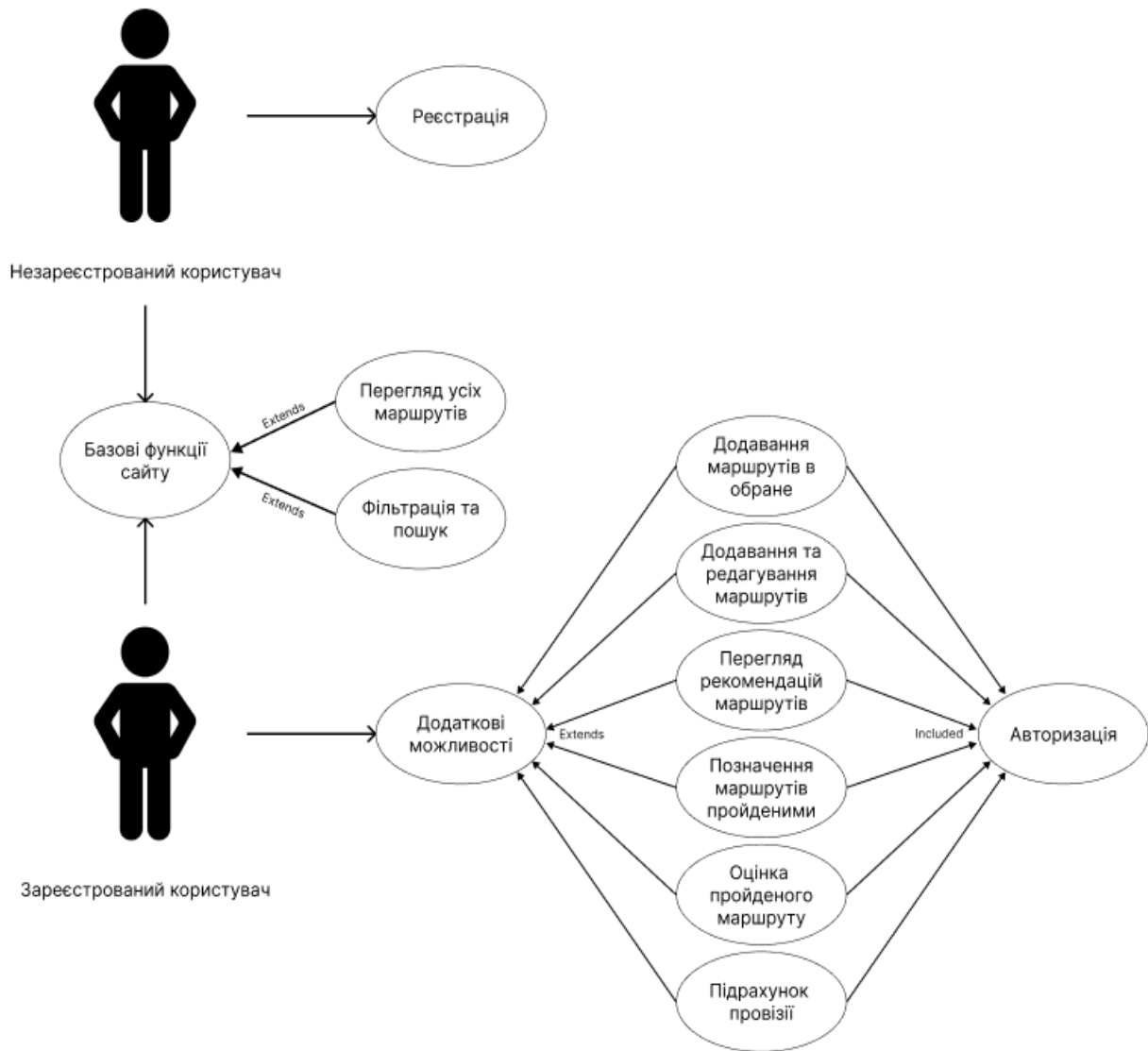


Рисунок 3.4 – Діаграма прецедентів ПЗ для моніторингу та рейтингування маршрутів у нотації UML

Загальна діаграма діяльності для моніторингу та рейтингування маршрутів зображена на рис. 3.5.

Загальна діаграма станів для моніторингу та рейтингування маршрутів зображена на рис. 3.6.

Також в процесі розробки програмного забезпечення для моніторингу та рейтингування маршрутів на основі діаграм взаємодії були створені діаграми кооперації та послідовності у нотації UML.

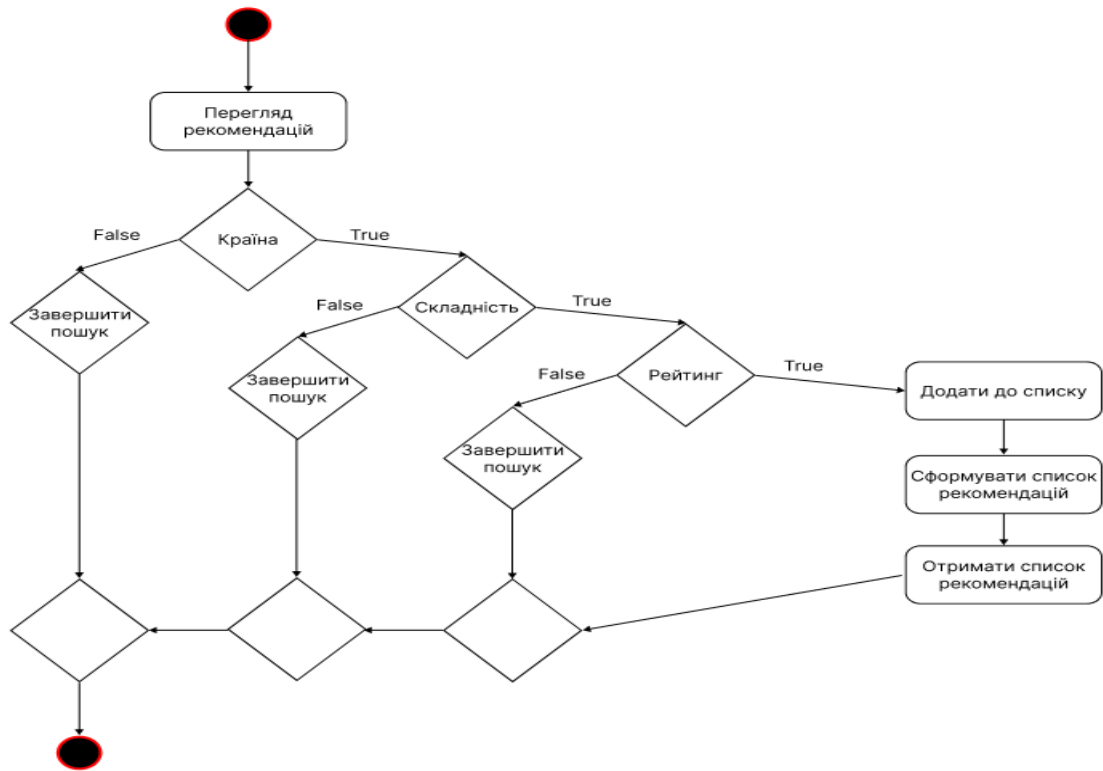


Рисунок 3.5 – Діаграма діяльності

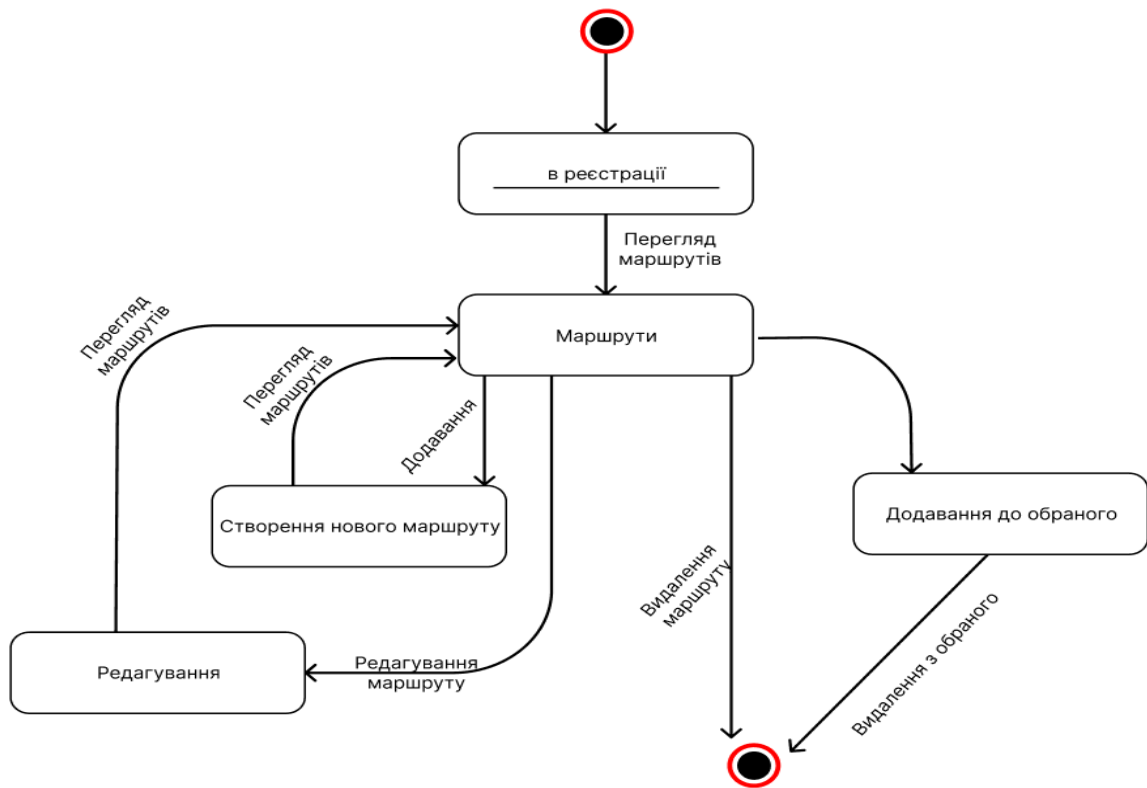


Рисунок 3.6 – Діаграма станів

Діаграма послідовності показує учасника взаємодії і послідовність дій, які він може виконувати. Загальна діаграма послідовності програмного забезпечення для моніторингу та рейтингування маршрутів у нотації UML зображена на рис. 3.7.

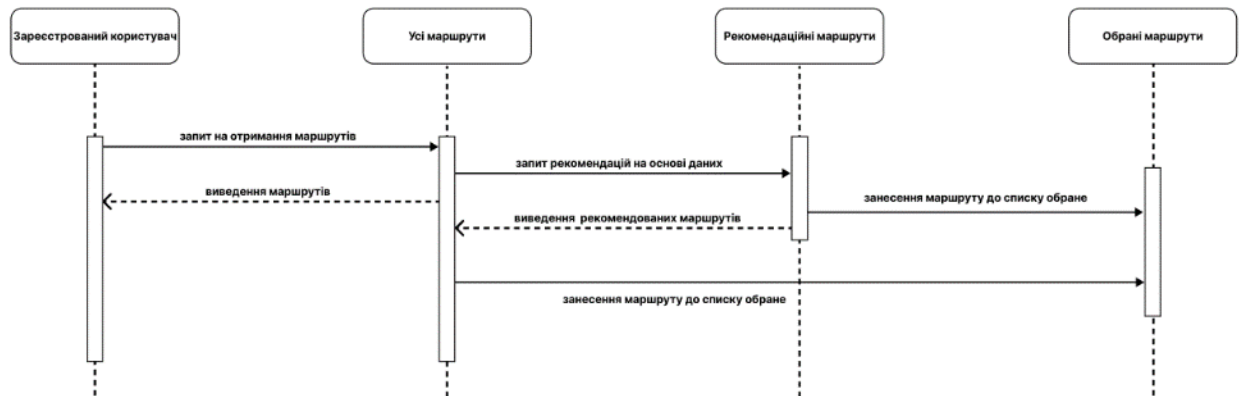


Рисунок 3.7 – Діаграма послідовностей

Загальна діаграма кооперацій ПЗ для моніторингу та рейтингування маршрутів у нотації UML зображена на рис. 3.8.

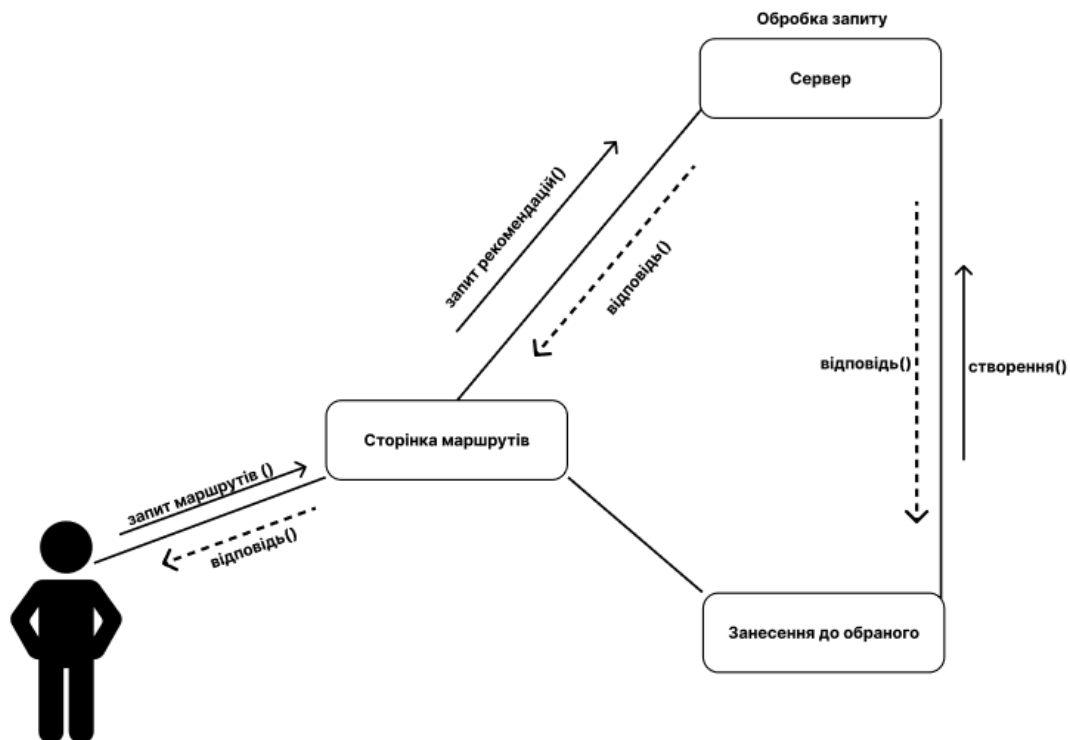


Рисунок 3.8 – Діаграма кооперацій

3.2 Проектування бази даних

Було спроектовано базу даних для збереження повної інформації про користувачів, дані для побудови маршрутів та інформацію про дії користувача над маршрутом такі як додавання та видалення з обраного, оцінка пройденого маршруту та оцінка його складності. Концептуальна модель бази даних з таблицями та полями сутностей зображена на рис. 3.9.

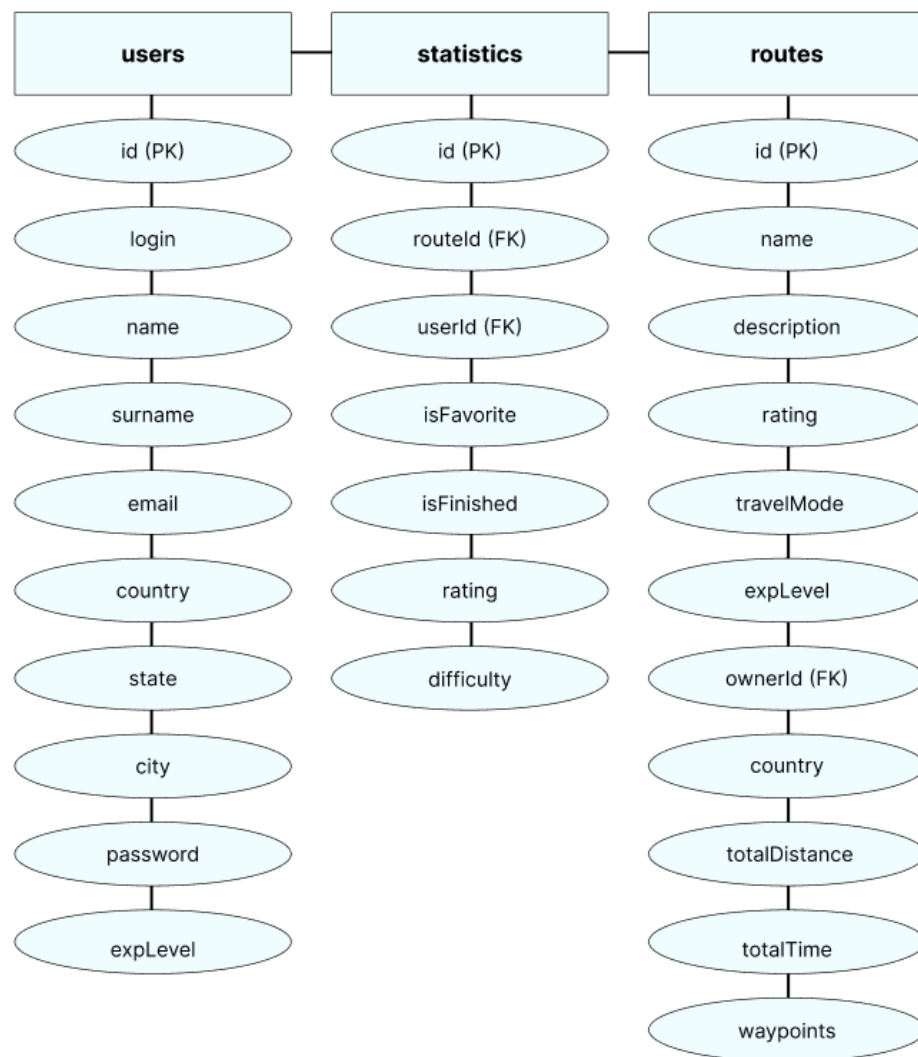


Рисунок 3.9 – Концептуальна модель бази даних

Сутність «users» (id, login, name, surname, email, country, state, city, password, expLevel) – зберігає повну інформацію про користувачів. Опис полів сутності та їх типу наведено у таблиці 3.1.

Таблиця 3.1 – Опис полів сутності «users»

№	Назва поля	Тип даних	Розмір
1	id	int	-
2	login	varchar	45
3	name	varchar	45
4	surname	varchar	45
5	email	varchar	45
6	country	varchar	50
7	state	varchar	50
8	city	varchar	50
9	password	varchar	255
10	expLevel	int	-

Сутність «routes» (id, name, description, rating, travelMode, expLevel, ownerId, country, totalDistance, totalTime, waypoints) – зберігає інформацію про маршрути. Опис полів сутності та їх типу наведено у таблиці 3.2.

Таблиця 3.2 – Опис полів сутності «routes»

№	Назва поля	Тип даних	Розмір
1	id	int	-
2	name	varchar	50
3	description	longtext	-
4	rating	double	-
5	travelMode	varchar	20
6	expLevel	tinyint	1
7	ownerId	int	-
8	country	varchar	50
9	totalDistance	int	-
10	totalTime	int	-
11	waypoints	longtext	-

Сутність «statistics» (id, routeId, userId, isFavorite, isFinished, rating, difficulty) – зберігає інформацію про дії користувача з маршрутом (додавання та видалення з обраного, оцінка маршруту та оцінка його складності, статус проходження маршруту). Опис полів сутності «routes» та їх типу наведено у таблиці 3.3.

Таблиця 3.3 – Опис полів сутності «statistics»

№	Назва поля	Тип даних	Розмір
1	id	int	-
2	routeId	int	-
3	userId	int	-
4	isFavorite	tinyint	1
5	isFinished	tinyint	1
6	rating	double	-
7	difficulty	int	-

3.3 Проектування дизайну

було проаналізовано основні методи проектування користувацьких інтерфейсів та було спроектовано дизайн на основі головних принципів user experience.

На рис. 3.10 зображено форму авторизації користувачів. Також з цієї форми можна перейти на форму реєстрації.

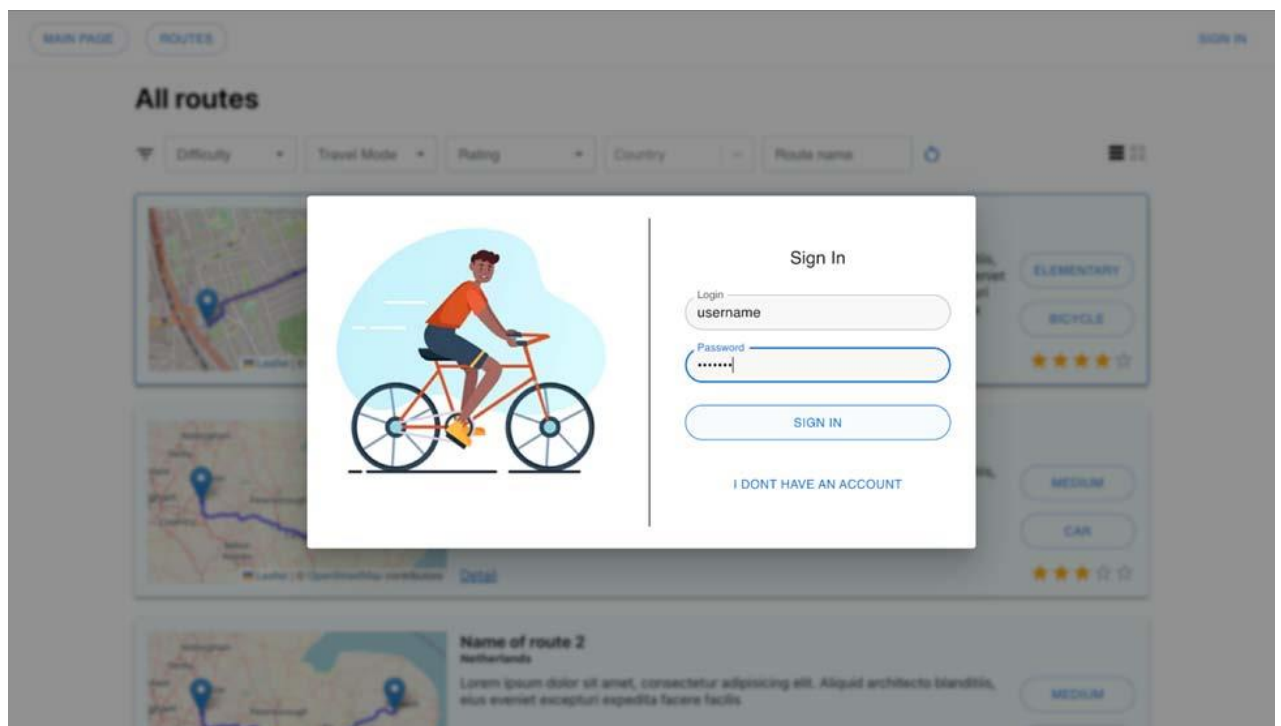


Рисунок 3.10 – Форма авторизації

На рис. 3.11 зображено форму реєстрації користувачів.

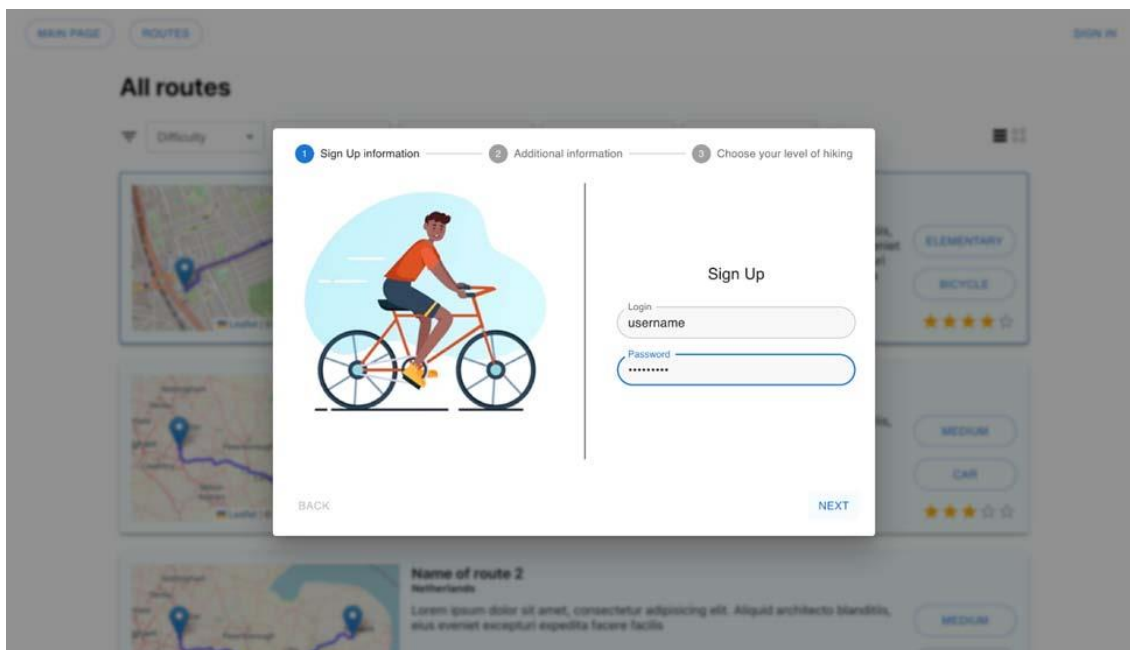
The image shows a web application interface with a modal registration form. The background is a blurred view of a 'All routes' page with map thumbnails and route details. The modal form is titled 'Sign Up' and has a progress indicator at the top with three steps: '1 Sign Up information' (active), '2 Additional information', and '3 Choose your level of hiking'. On the left side of the modal, there is an illustration of a person riding a bicycle. On the right side, there are two input fields: 'Login' with the placeholder 'username' and 'Password' with masked characters. At the bottom left of the modal is a 'BACK' button, and at the bottom right is a 'NEXT' button.

Рисунок 3.11 – Форма реєстрації

На рис. 3.12 зображено форму реєстрації користувачів для заповнення основної інформації, необхідної для логіну.

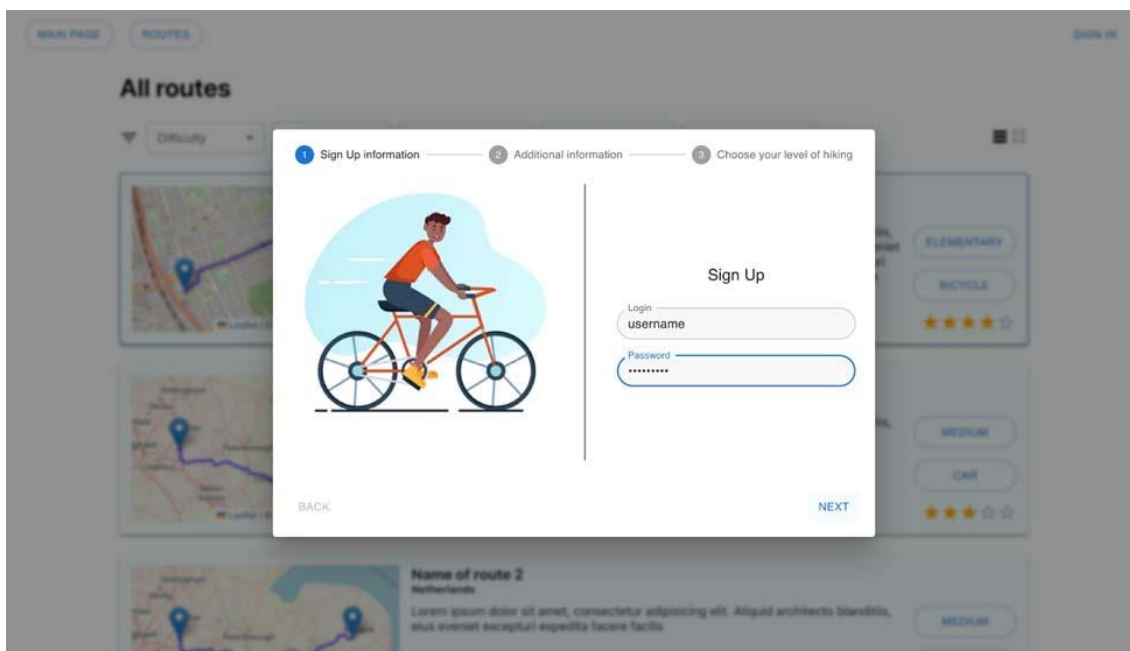
This image is identical to the one above, showing the same registration modal form. It features the same background, progress indicator, bicycle illustration, input fields for 'Login' (username) and 'Password', and 'BACK'/'NEXT' buttons.

Рисунок 3.12 – Форма реєстрації - заповнення основної інформації користувача

На рис. 3.13 зображено форму реєстрації користувачів для заповнення інформації про досвід користувача.

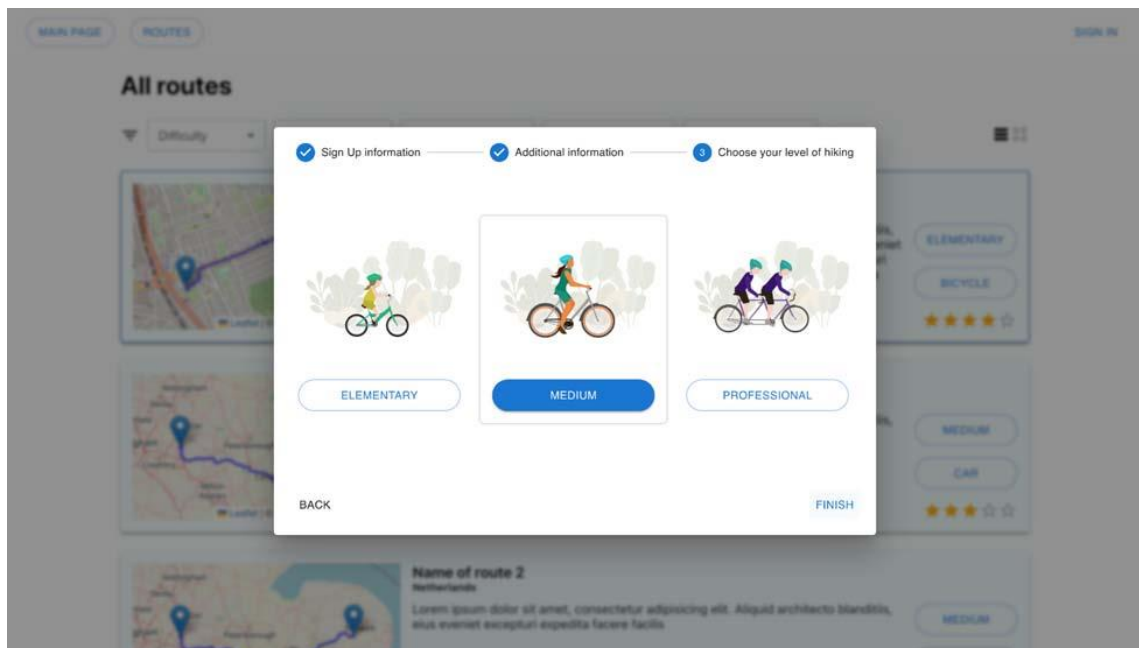


Рисунок 3.13 – Форма реєстрації – вибір досвіду користувача

На рис. 3.14 та рис. 3.15 представлено сторінку з усіма маршрутами в двох варіантах відображення.

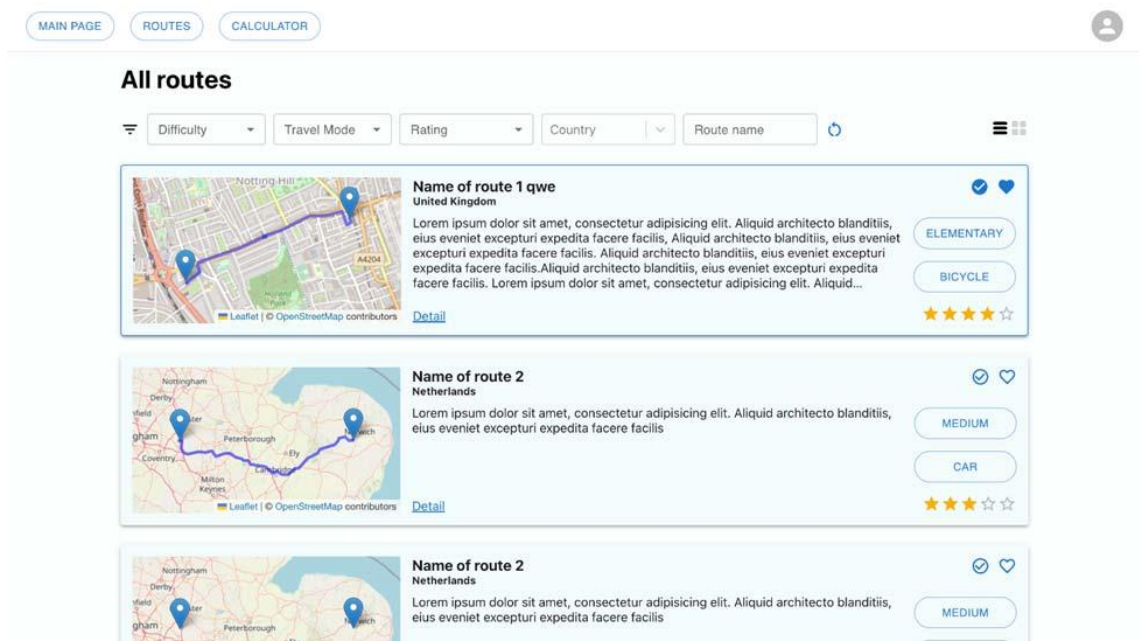


Рисунок 3.14 – Сторінка усіх маршрутів

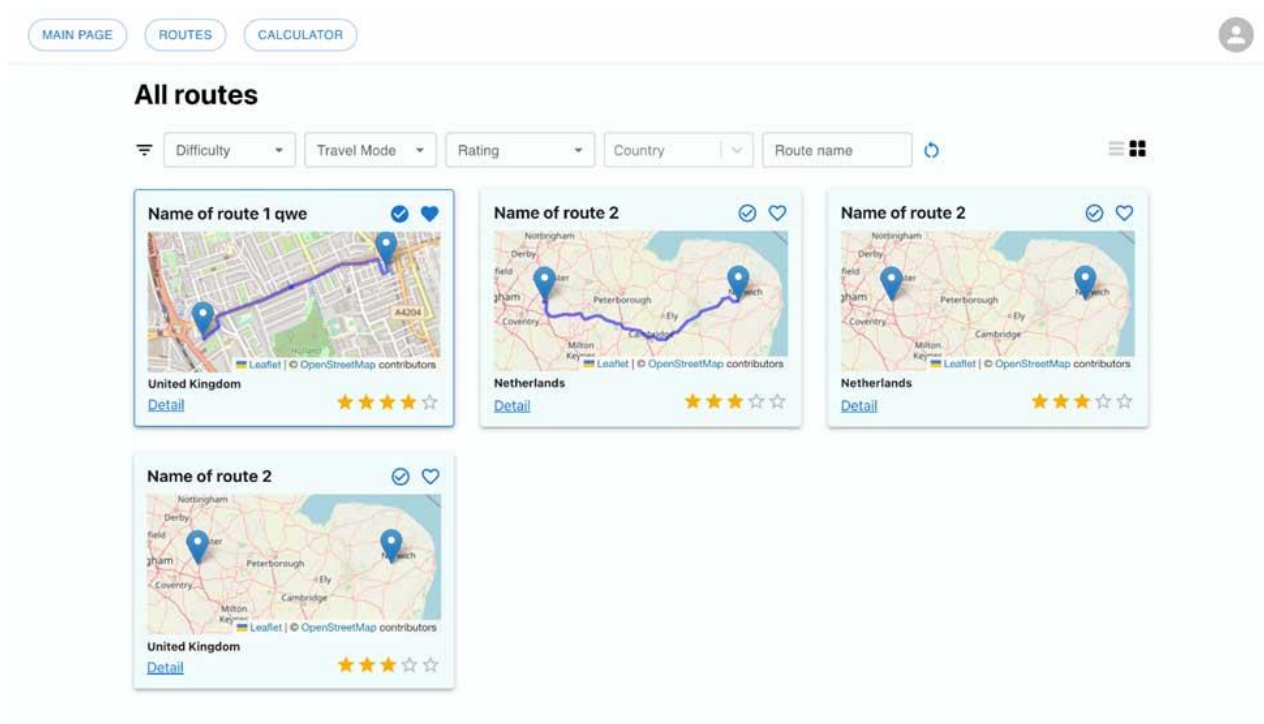


Рисунок 3.15 – Сторінка усіх маршрутів – з табличним виглядом

Сторінку редагування інформації користувача зображено на рис. 3.16. На цій сторінці можна змінити основну інформацію та рівень досвіду користувача.

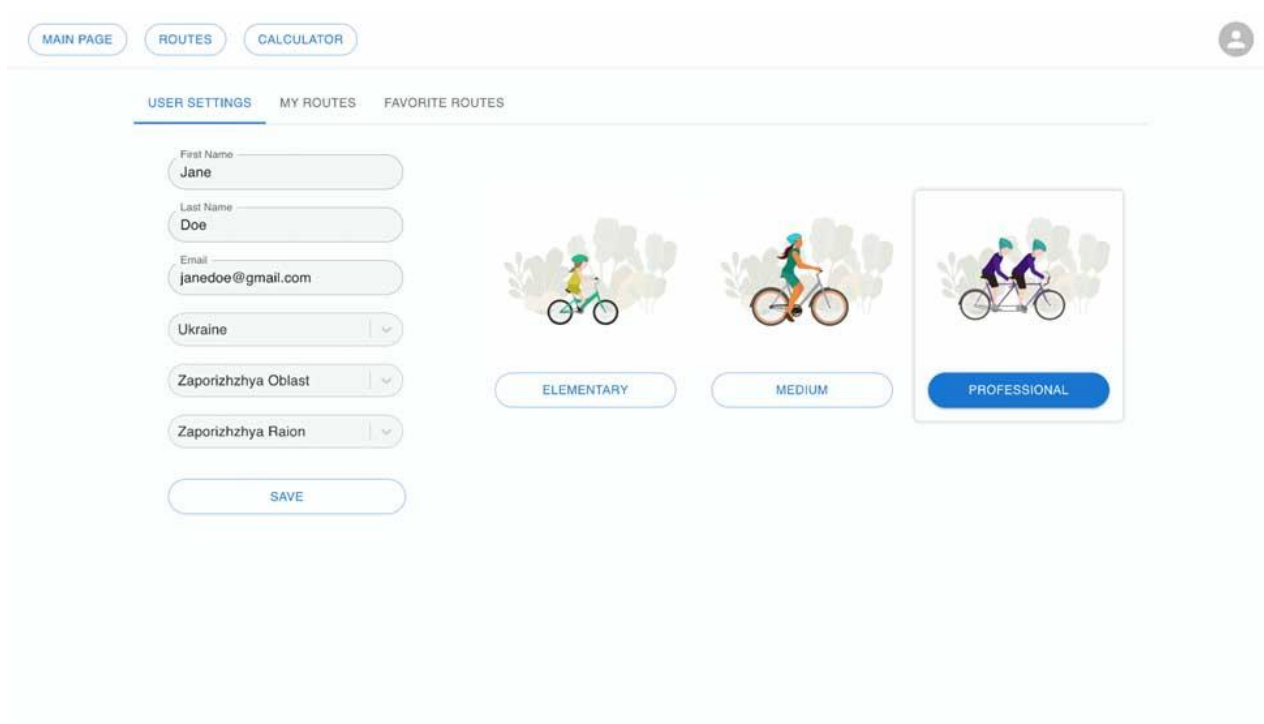


Рисунок 3.16 – Сторінка редагування інформації користувача

На рис. 3.17 зображено форму створення маршруту.

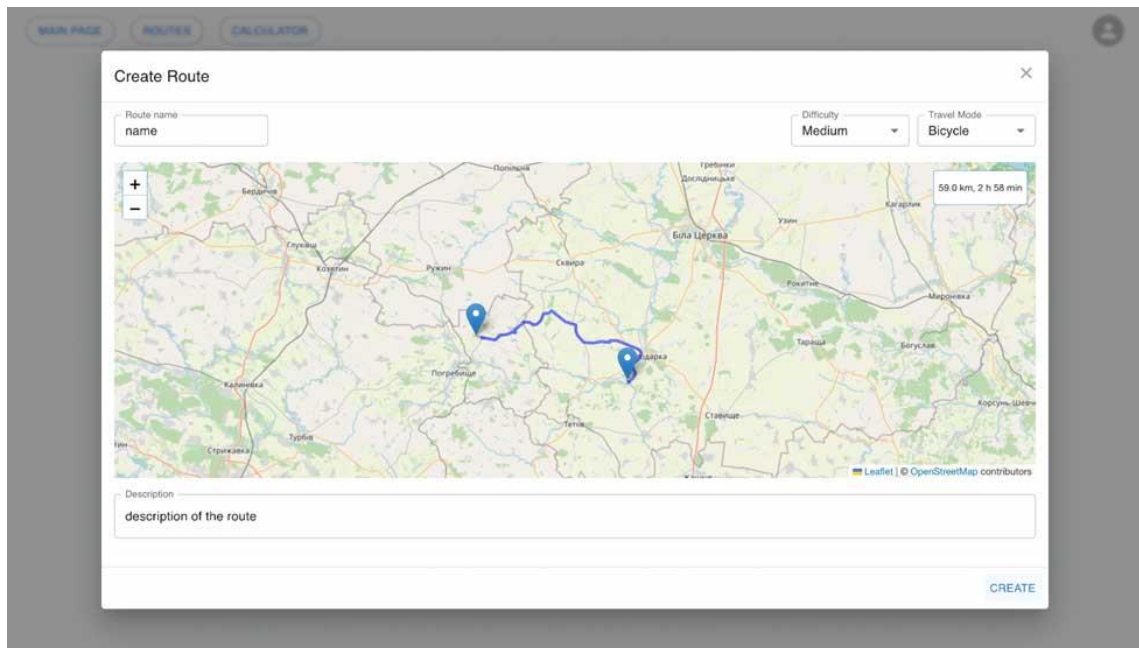


Рисунок 3.17 – Форма створення маршруту

На рис. 3.18 та рис. 3.19 представлено сторінку перегляду створених маршрутів у двох варіантах відображення.

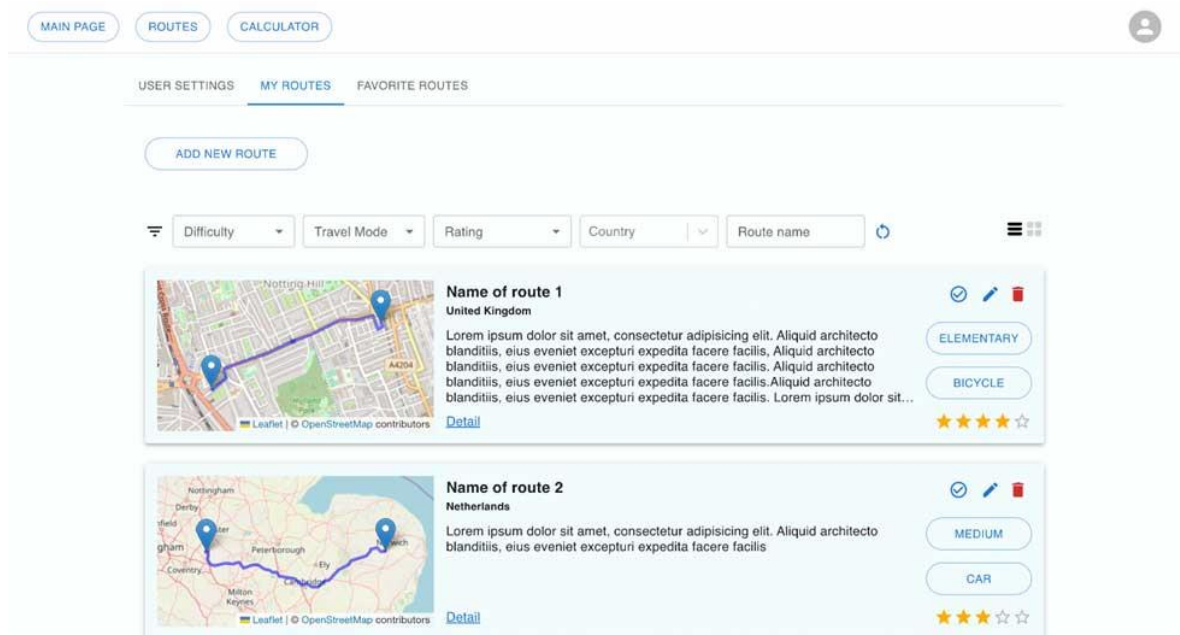


Рисунок 3.18 – Сторінка перегляду створених маршрутів

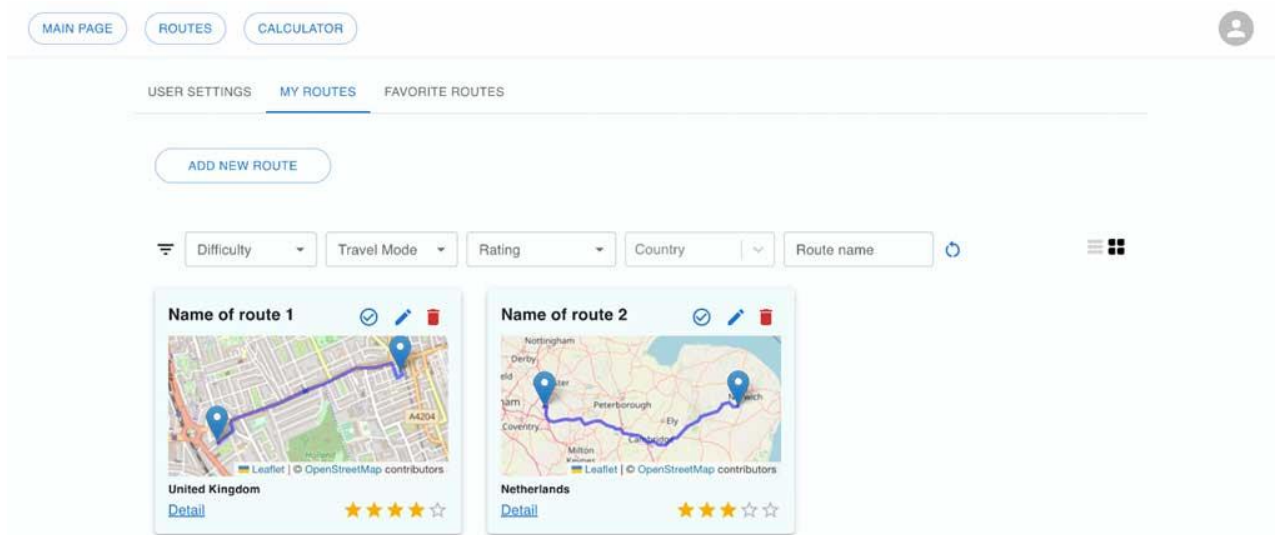


Рисунок 3.19 – Сторінка перегляду створених маршрутів в табличному вигляді

3.4 Висновки за розділом 3

Розроблене програмне забезпечення у вигляді вебсайту для перегляду інформації, моніторингу та рейтингування маршрутів.

Запропоновано структуру програмного забезпечення для перегляду інформації, моніторингу та рейтингування маршрутів, яку подано у вигляді структурних діаграм в нотації UML, зокрема діаграми класів, пакетів та компонентів.

Функціонування програмного забезпечення для перегляду інформації, моніторингу та рейтингування маршрутів зображено за допомогою абстрактних моделей у нотації UML основі діаграм поведінки, зокрема, діаграм прецедентів, станів, діяльності системи. В процесі розробки програмного забезпечення для перегляду інформації, моніторингу та рейтингування маршрутів на основі діаграм взаємодії було створено діаграми кооперації та послідовності у нотації UML.

Також було розглянуто проектування бази даних, наведено таблиці з описом полів сутностей та їх типів.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Опис застосунку

Даний проєкт – призначений для туристів та дозволяє спростити пошук та вибір маршрутів у наступну подорож. Створений ресурс, дозволить легко моніторити особисті та популярні маршрути. За допомогою цього проєкту легко знайти або створити нові напрямки для походів та розрахувати необхідну кількість провізії.

Створення даного продукту – призначене для допомоги мандрівникам, побачити свою країну. Цей застосунок допоможе знайти найцікавіші місця та маршрути, а також – розрахувати необхідну кількість провізії на цей шлях.

Кожен користувач зможе передивлятися списки маршрутів. Крім того зареєстровані користувачі зможуть користуватися аналітичною складовою, яка розрахована на рейтингування та представлення популярних маршрутів місяця або міста користувача, а також надання рекомендованих маршрутів, в залежності від досвіду користувача.

Програмне забезпечення у вигляді вебсайту для моніторингу та рейтингування маршрутів забезпечує виконання таких функцій:

- реєстрація та авторизація в системі;
- пошук маршрутів;
- перегляд повного списку маршрутів;
- перегляд рекомендованих маршрутів;
- додавання (редагування/видалення) особистих маршрутів;
- рейтингування маршрутів;
- фільтрація маршрутів;
- калькуляцію розрахунків запасів їжі.

Система також повинна бути стійкою до збоїв, працювати як на десктопах, так і на мобільних пристроях, бути адаптованою під різні розширення екранів і легкою для подальшої підтримки і масштабування.

4.2 Умови виконання програми

Мінімальні системні вимоги та програмне забезпечення , що необхідні для функціонування серверної частини мобільного застосунку:

- операційна система: Linux;
- процесор з двома ядрами і з тактовою частотою не менше 2 ГГц;
- оперативна пам'ять: 8 Гбайт;
- встановлене програмне забезпечення Docker;
- дисковий простір: SSD 25Гбайт.
- Для експлуатації клієнтської частини необхідні такі програмно-технічні засоби:
- сучасний браузер (Google Chrome, Mozilla Firefox, Safari);
- стабільне підключення до мережі Інтернет.

4.3 Інструкція по експлуатації програми

4.3.1 Характеристики програми

Найбільш повно логічна взаємодія відображена на рис. 4.1, завдяки розподілу функцій між частинами застосунку.

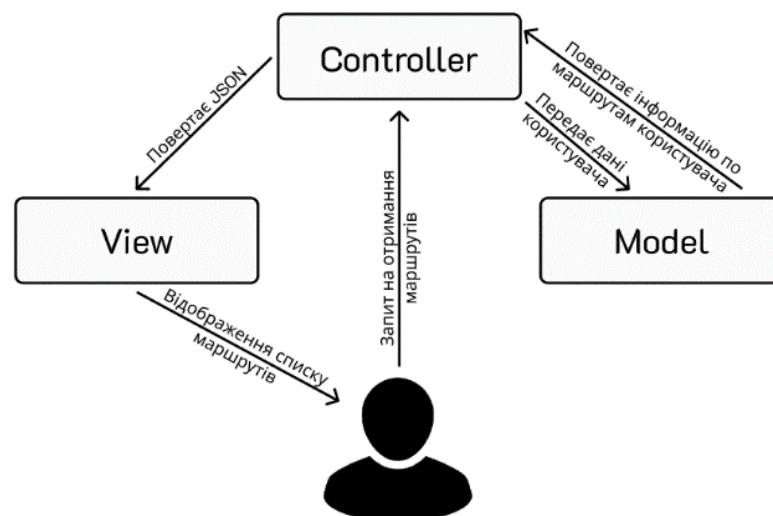


Рисунок 4.1 – Компоненти та зв'язки

Для представлення бізнес-логіки було використано шаблон Model-View-Controller. MVC – це схема поділу даних програми, призначеного для користувача інтерфейсу і керуючої логіки на три окремих компоненти: модель, уявлення і контролер - таким чином, що модифікація кожного компонента може здійснюватися незалежно.

Роль контролерів виконує Laminas Framework, запити виконуються через програмний інтерфейс застосунку (API).

В якості моделі було використано Laminas Framework з ORM Doctrine для обробки запитів до БД.

В якості уявлення було використано React - він використовується для побудови інтерактивних інтерфейсів користувача.

4.3.2 Використані технічні засоби

При розробці ПЗ для моніторингу та рейтингування маршрутів, для побудови маршрутів було використано такі технічні засоби як:

- Leaflet – використовується для відображення карт;
- React Leaflet – обгортка для Leaflet карт для зручнішого використання їх у React;
- TomTom –сервіс онлайн маршрутизації;
- Leaflet Routing Machine – використовується для поєднання React Leaflet та TomTom і розширює можливості Leaflet мап, такими функціями як встановлення, видалення та зміна міток на мапі [12]-[15].

4.3.3 Звернення до програми

Для розгортання застосунку необхідно мати на ПК встановлені PHP 8.1, Apache2, платформа Node.js версії 14.17.0 або вище, завантажити код з GitHub репозиторію.

Для запуску проєкту локально, необхідно встановити залежності за допомогою пакетного менеджера npm. Для цього потрібно перейти в директорію клієнтської частини програми в терміналі та виконати команду `npm install`, після чого виконати команду `npm start` – виконання цього скрипту запустить програму, яка буде доступна за адресою – `http://localhost:3000/`.

Для запуску серверної частини застосунку необхідно перейти в директорію серверної частини програми в терміналі та виконати команду: `php -S 127.0.0.1:3000 index.php`.

Також потрібно додати змінні оточення. Для цього створюємо файл `.env`. Приклад файлу знаходиться в директорії клієнтської частини - `.env.example`.

- `REACT_APP_API_HOST_PROTOCOL` - хост протокол бекенд частини програми (`http/https`);
- `REACT_APP_API_URL` - url бекенд частини програми;
- `REACT_APP_API_PORT` – порт бекенда;
- `REACT_APP_TOM_TOM_API_KEY` - апі ключ вебсервісу маршрутизації `tomtom`.
- `REACT_APP_GEONAMES_USERNAME` - ім'я користувача для запитів до АПІ для отримання списку країн.

4.3.4 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення для трекінгу туристичних маршрутів є база даних, яка зберігає інформацію про користувачів та усі маршрути. Вхідними даними також є дані, введені користувачем з клавіатури та отримані дані з серверу, кнопки, завдяки яким є можливість обрати відповідний пункт меню та отримати відповідну інформацію.

На основі вхідних та вихідних даних було спроектовано базу даних та було побудовано концептуальну модель бази даних (рис. 4.2).

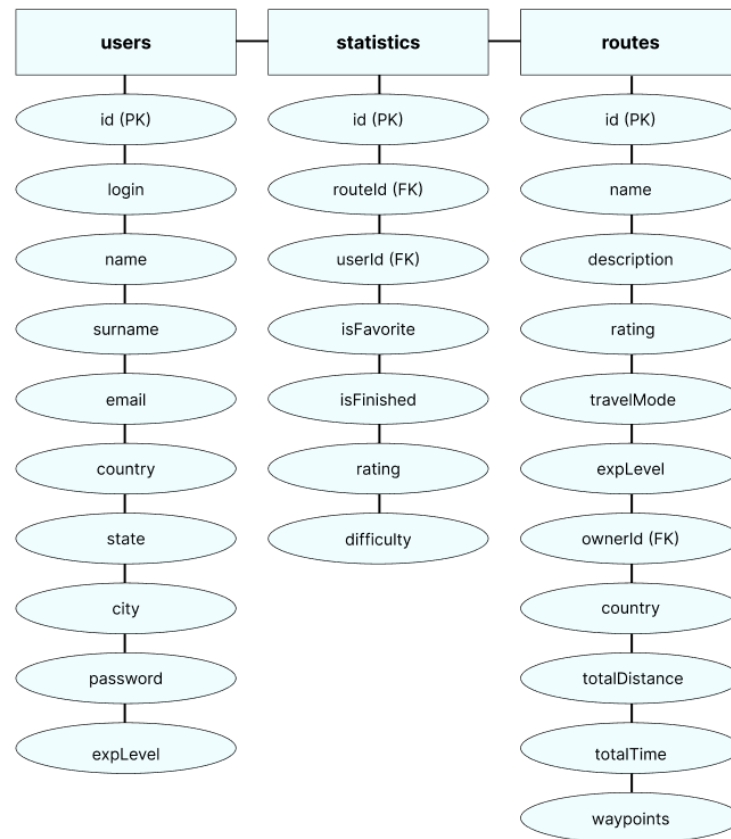


Рисунок 4.2 – Концептуальна модель бази даних

Вихідними даними є графічне представлення маршрутів на картах.

4.4 Методика та результати тестування

Для того аби перевірити роботу спроможність застосунку будуть проведені наступні види тестування:

- функціональне тестування. Перевірка функцій та характеристик ПЗ, що розробляється, на основі проєктної документації;
- юзабіліті-тестування. Мета даного виду тестування полягає у визначенні ступеня зручності та практичності користувальницького інтерфейсу;
- тестування сумісності. Перевірка можливості програми взаємодіяти з різними програмними продуктами, операційними системами та оточенням.

4.4.1 Тестування функціоналу

Незареєстрований користувач може переглядати список усіх маршрутів та користуватися фільтрами. Для цього користувачу треба на головній сторінці обрати необхідні параметри – фільтр повинен автоматично спрацьовувати. На рис. 4.3 продемонстрована коректна робота фільтрів.

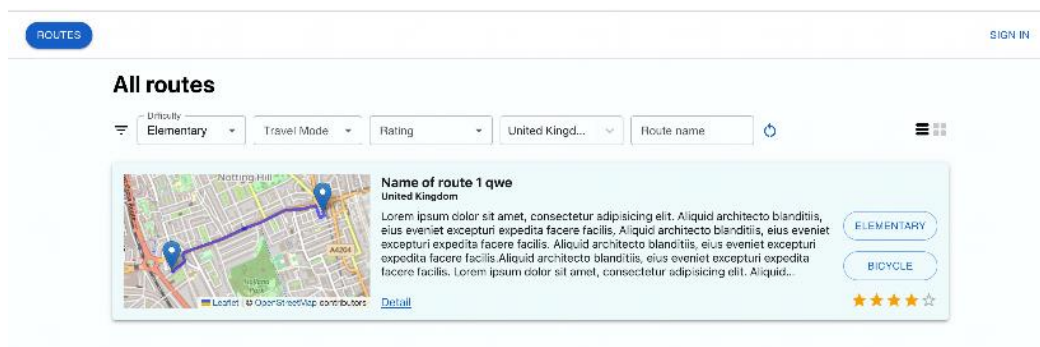


Рисунок 4.3 – Відображення списку маршрутів з використаними фільтрами

Для того щоб зареєструватися, незареєстрованому користувачу треба натиснути кнопку «Sign In» та обрати пункт «I don't have an account», заповнити усю необхідну інформацію та натиснути кнопку «Create». Форма реєстрації наведена на рис. 4.4-4.6.

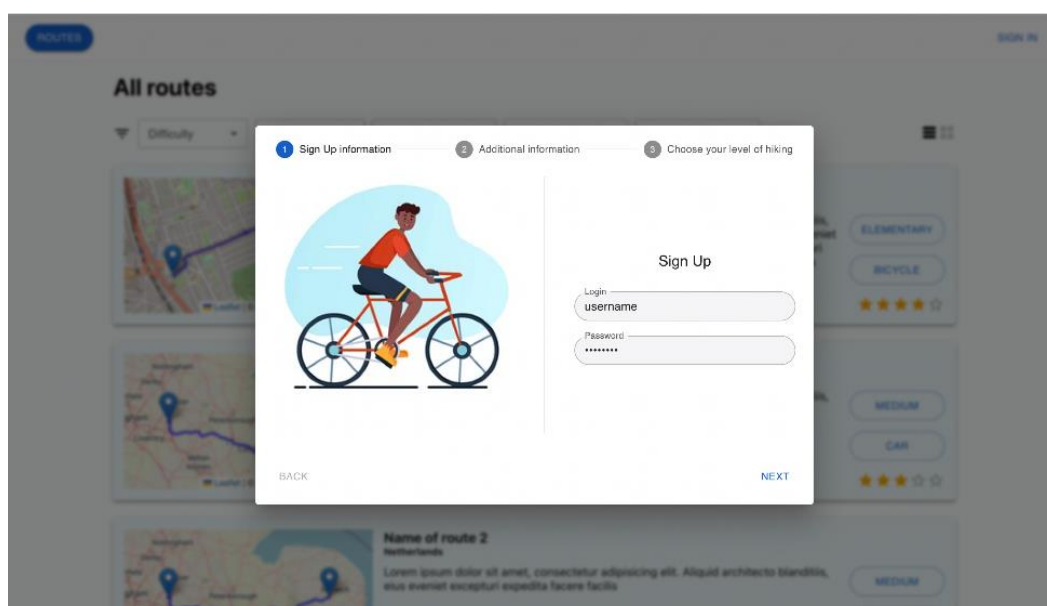


Рисунок 4.4 – Форма реєстрації

ROUTES SIGN IN

All routes

Difficulty

1 Sign Up information 2 Additional information 3 Choose your level of hiking

First Name

Last Name

Email

Country

BACK NEXT

Name of route 2
Netherlands
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam architecto blandit, eius enim excepturi expedita facere facilis

ELEMENTARY BICYCLE

MEDIUM CAR

MEDIUM

Рисунок 4.5 – Форма реєстрації – заповнення основної інформації користувача

ROUTES SIGN IN

All routes

Difficulty

1 Sign Up information 2 Additional information 3 Choose your level of hiking

ELEMENTARY MEDIUM PROFESSIONAL

BACK CREATE

Name of route 2
Netherlands
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam architecto blandit, eius enim excepturi expedita facere facilis

ELEMENTARY BICYCLE

MEDIUM CAR

MEDIUM

Рисунок 4.6 – Форма реєстрації – вибір досвіду користувача

Зареєстрований користувач також може переглядати список усіх маршрутів і використовувати фільтри. На рис. 4.7 відображено список маршрутів з фільтрами у скороченому вигляді.

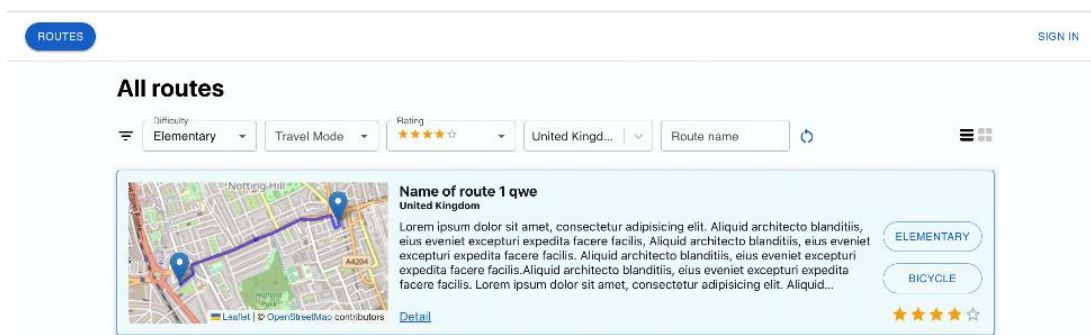


Рисунок 4.7 – Відображення списку маршрутів з використаними фільтрами

Зареєстрований користувач може створювати нові маршрути. Для цього необхідно перейти на вкладинку «User» -> «My Routes» та натиснути кнопку «Add new route». На рис. 4.8 зображено форму створення нового маршруту, на якій треба намалювати на карті маршрут, ввести назву, обрати тип маршруту та його складність.

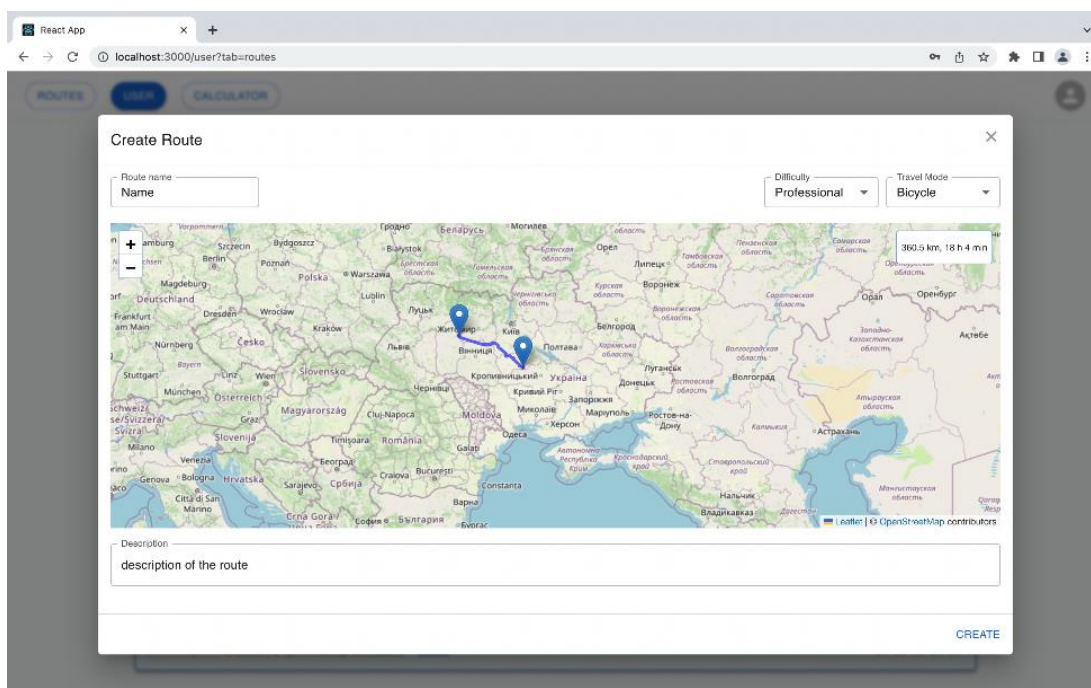


Рисунок 4.8 – Форма створення нового маршруту

Для розрахунку провізії, користувачу необхідно перейти на вкладинку «Calculator», ввести кількість осіб та кількість днів. Підрахована провізія забражена на рис. 4.9.

ROUTESUSERCALCULATOR

Calculator

Number of participants

2

Duration, days

2

Day	Calories	Weight (g)	Protein (g)	Fat (g)	Carbs (g)	Weight for group (kg)	actions
Day 1	3836	970	142.53	171.64	406.92	1.94	
Day 2	3836	970	142.53	171.64	406.92	1.94	

Total	Calories	Weight (kg)	Protein (g)	Fat (g)	Carbs (g)
Person for all days	7672	1.94	285.06	343.28	813.84
Group for all days	15344	3.88	570.12	686.56	1627.68

Рисунок 4.9 – Відображення таблиці провізії

4.4.2 Тестування верстки

Тестування верстки програмного забезпечення для моніторингу та рейтингування маршрутів буде проводитись у різних браузерах для перевірки, наскільки розроблений застосунок сумісний із найбільш популярними браузерами.

На рис. 4.10, продемонстровано коректне відображення застосунку через браузер Google Chrome.

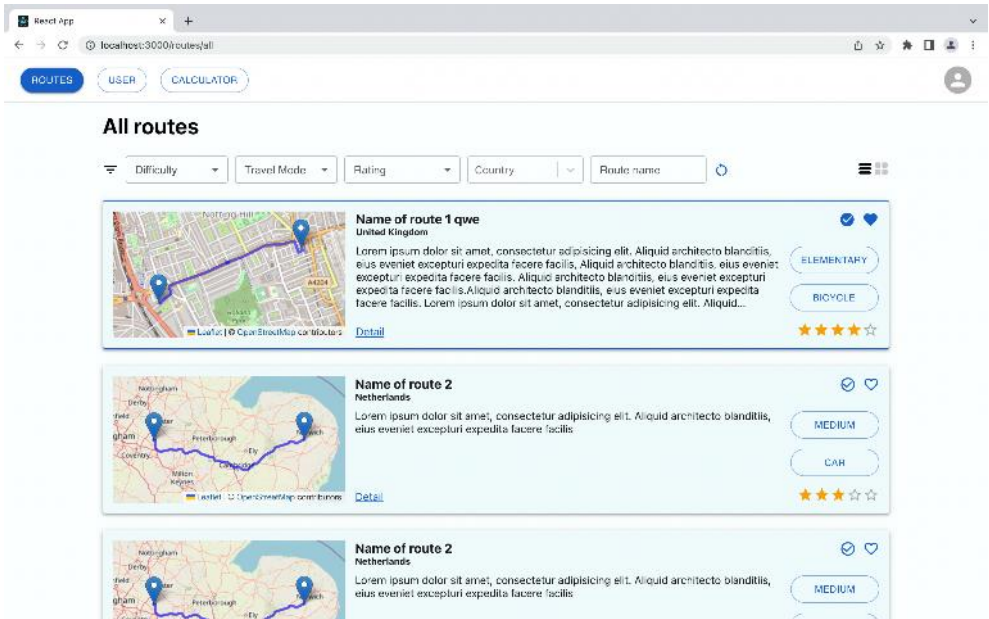


Рисунок 4.10 – Відображення головної сторінки через браузер Google Chrome

Аналогічно в браузері Safari сторінки програмного забезпечення для моніторингу та рейтингування маршрутів відображаються коректно. Це відображено на рис. 4.11.

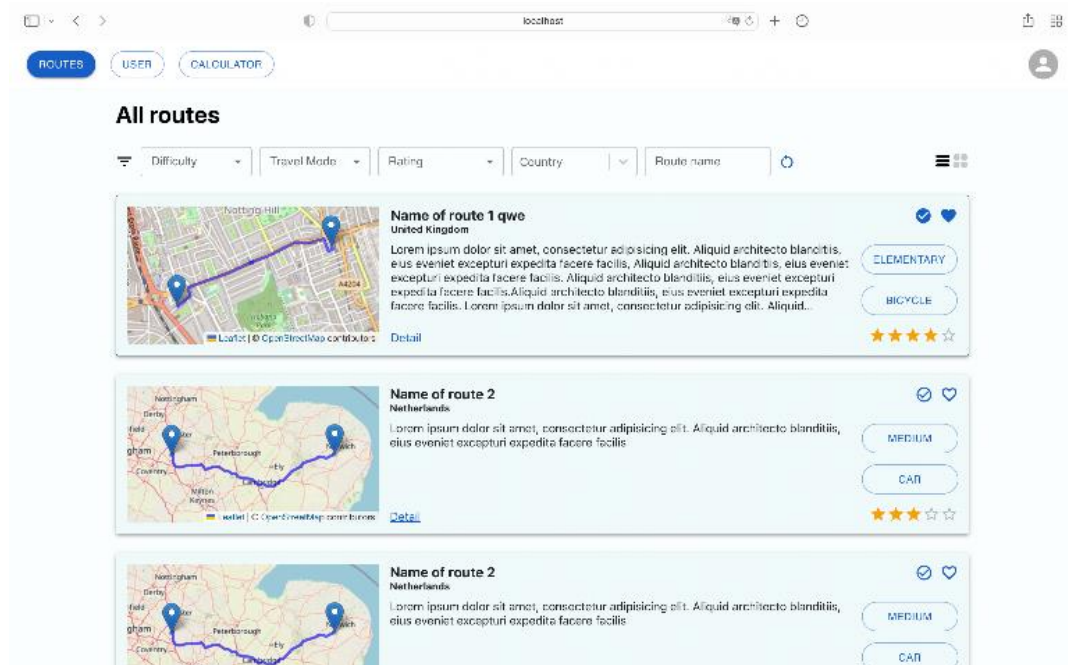


Рисунок 4.11 – Відображення головної сторінки застосунку через браузер Safari

Інші сторінки також коректно відображаються у браузерах Google Chrome та Safari.

4.4.3 Чек-листи тестування

Функціональне тестування проводилося разом із тестуванням верстки. Для цього використовуючи різні апаратні та програмні платформи, на яких було запущено клієнтську частину за стосунку. Фіксація результатів тестування представлена у вигляді складених чек-листів, де серед критеріїв зазначено етапи функціонального тестування, а результати фіксувались у закріпі відповідному до платформи тестування.

Результати тестування основного функціоналу наведені у таблиці 4.1.

Таблиця 4.1 – Основні перевірки функціоналу

Перевірка	Google Chrome	Safari
Реєстрація	+	+
Авторизація	+	+
Авторизація неіснуючого користувача	+	+
Авторизація користувача з неправильними даними	+	+
Перегляд списку маршрутів (неавторизований користувач)	+	+
Перегляд списку маршрутів (авторизований користувач)	+	+
Фільтрація маршрутів	+	+
Пошук маршрутів	+	+
Додавання маршрутів в обране (авторизований користувач)	+	+
Видалення маршруту з обраного (авторизований користувач)	+	+
Позначення маршруту пройденим (авторизований користувач)	+	+
Оцінка пройденого маршруту	+	+
Підрахунок провізії	+	+

4.5 Висновки за розділом 4

Описано програмне забезпечення у вигляді вебсайту для моніторингу та рейтингування маршрутів.

Визначено мінімальні системні вимоги необхідні для коректної роботи програмного забезпечення для моніторингу та рейтингування маршрутів.

Наведено опис та результати процесу тестування програмного забезпечення для моніторингу та рейтингування маршрутів.

Результати тестування програмного забезпечення для моніторингу та рейтингування маршрутів показали, що розроблений вебзастосунок працює коректно та може використовуватися за призначенням.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для трекінгу туристичних маршрутів.

Проаналізовано проблему розроблення програмного забезпечення для моніторингу та рейтингування туристичних маршрутів. Визначено, що наше сьогодення стрімко змінює ставлення громадян до туризму. Сучасна молодь активно пропагує здоровий спосіб життя, саморозвиток, заняття спортом, зокрема, туризмом. Охочих подорожувати дедалі стає все більше. При цьому кількість подорожей заради відпочинку (пасивний туризм) зменшується. Натомість отримують поширення подорожі-експедиції, які ставлять метою пізнання культури, історії, природи країни, в якій подорожуєш. Саме для допомоги мандрівникам, побачити свою країну – розроблюється вебзастосунок «Travel Helper». Він допоможе знайти найцікавіші місця та маршрути, а також – розрахувати необхідну кількість провізії на цей шлях..

Проаналізовано програмні засоби для трекінгу туристичних маршрутів. Визначено, що майже всі додатки потребують додаткової оплати своїх послуг. Більш того, не всі сайти мають фільтрацію та рейтингування маршрутів. Також жоден аналог не має функції розрахунку продовольства. Саме тому актуальною залишається задача розробки нового застосунку «Travel Helper» із привабливим та сучасним дизайном, високою швидкістю роботи та достатнім рівнем усіх показників. А головне - цей застосунок надає унікальну можливість аналітики досвіду користувача.

Для написання серверної частини застосунку було обрано мову програмування PHP. Основними перевагами цієї мови програмування являється ефективність, гнучкість, швидкість виконання коду та підтримка великої кількості баз даних.

Для написання клієнтської частини застосунку було обрано мову програмування JavaScript – це крос-платформна, об'єктно-орієнтована

скриптова мова, яка є невеликою і легковажною. Всередині виконання JavaScript може бути пов'язаний з об'єктами даного середовища і надавати програмний контроль над ними.

Розроблене програмне забезпечення у вигляді вебсайту для для трекінгу туристичних маршрутів.

Запропоновано структуру програмного забезпечення для моніторингу та рейтингування туристичних маршрутів, яку подано у вигляді структурних діаграм в нотації UML, зокрема діаграми класів, пакетів та компонентів.

Функціонування програмного трекінгу туристичних маршрутів зображено за допомогою абстрактних моделей у нотації UML основі діаграм поведінки, зокрема, діаграм прецедентів, станів, діяльності системи. В процесі розробки програмного забезпечення для моніторингу та рейтингування туристичних маршрутів на основі діаграм взаємодії було створено діаграми кооперації та послідовності у нотації UML.

Описано файлову структура програмного забезпечення для трекінгу туристичних маршрутів.

Здійснено проєктування інтерфейсу взаємодії користувача з програмою для моніторингу та рейтингування туристичних маршрутів.

Описано програмне забезпечення у вигляді вебсайту для трекінгу туристичних маршрутів «Travel Helper».

Визначено мінімальні системні вимоги необхідні для коректної роботи програмного забезпечення у вигляді вебзастосунку для трекінгу туристичних маршрутів.

Наведено опис та результати процесу тестування програмного забезпечення для трекінгу туристичних маршрутів. Також було описано процес встановленню серверу та клієнтської частини для роботи з програмою для моніторингу та рейтингування туристичних маршрутів.

Результати тестування програмного забезпечення для трекінгу туристичних маршрутів показали, що розроблений вебзастосунок працює коректно та може використовуватися за призначенням.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. TurUA. Велотуризм в Україні: маршрути, спорядження, поради [Електрон. ресурс]. – Режим доступу: <https://turua.com.ua/articles/veloturizm-v-ukrayini-marshruti-sporyadzhennya-poradi/>.
2. IGoToWorld. Велотуризм в Україні: поради та спорядження [Електрон. ресурс]. – Режим доступу: https://ua.igotoworld.com/ru/article/644_veloturizm-v-ukraine.htm.
3. Велосипедний туризм і велопоходи [Електрон. ресурс]. – Режим доступу: <https://boomerang-boardshop.ua/velosypednyj-turyzm-i-velopohody/>.
4. Strava [Electronic resource]. – Access mode: <https://www.strava.com/>.
5. В Похід Карпатами [Електрон. ресурс]. – Режим доступу: <https://vpohid.com.ua/>.
6. Wikiloc - Стежки Світу [Електрон. ресурс]. – Режим доступу: <https://uk.wikiloc.com/>.
7. PHP [Electronic resource]. – Access mode: <https://www.php.net/>.
8. JS [Electronic resource]. – Access mode: <https://www.javascript.com/>.
9. Laminas [Electronic resource]. – Access mode: <https://api-tools.getlaminas.org/>.
10. TypeScript in Visual Studio Code [Electronic resource]. – Access mode: <https://code.visualstudio.com/docs/languages/typescript>.
11. MySQL [Electronic resource]. – Access mode: <https://www.mysql.com/>.
12. Leaflet [Electronic resource]. – Access mode: <https://leafletjs.com/>.
13. React-leaflet [Electronic resource]. – Access mode: <https://react-leaflet.js.org/docs/start-introduction/>.
14. TomTom [Electronic resource]. – Access mode: <https://developer.tomtom.com/routing-api/documentation/product-information/>.
15. Leaflet Routing Machine [Electronic resource]. – Access mode: <https://www.liedman.net/leaflet-routing-machine/#about>.

ДОДАТОК А
Технічне завдання

A.1 Вступ

Назва програми – Програмна реалізація застосунку для трекінгу туристичних маршрутів «Travel Helper». Система призначена для допомоги мандрівникам, які шукають нові маршрути для себе або компанії. За допомогою цього застосунку легко моніторити особисті або популярні маршрути.

A.2 Підстава для розробки

Підставою для розробки слугує завдання на випускню роботу на тему «Програмна реалізація застосунку для трекінгу туристичних маршрутів», затверджене наказом Національного університету «Запорізька політехніка» № 102 від 27 квітня 2022 р.

A.3 Призначення розробки

Даний проєкт – призначений для туристів та дозволяє спростити пошук та вибір маршрутів у наступну подорож. Створений ресурс, дозволить легко моніторити особисті та популярні маршрути. За допомогою цього проєкту легко знайти або створити нові напрямки для походів та розрахувати необхідну кількість провізії.

Створення даного продукту – призначене для допомоги мандрівникам, побачити свою країну. Цей застосунок допоможе знайти найцікавіші місця та маршрути, а також – розрахувати необхідну кількість провізії на цей шлях.

Кожен користувач зможе передивлятись списки маршрутів. Крім того зареєстровані користувачі зможуть користуватися аналітичною складовою, яка розрахована на рейтингування та представлення популярних маршрутів місяця або міста користувача, а також надання рекомендованих маршрутів, в залежності від досвіду користувача.

A.4 Основні вимоги до програми, що розробляється

A.4.1 Вимоги до функціональних характеристик

Вимоги до функціональних характеристик.

Робота із застосунком – не зареєстрований користувач:

- реєстрація;
- перегляд повного списку маршрутів;
- фільтрація маршрутів по доступним критеріям.

Робота із застосунком – зареєстрований користувач:

- реєстрація та авторизація;
- перегляд повного списку маршрутів;
- перегляд рекомендованих маршрутів;
- фільтрація маршрутів по доступним критеріям;
- можливість додати/редагувати/видалити власні маршрути;
- можливість ставити оцінки маршрутам;
- калькуляція провізії на певні маршрути.

A.4.2 Вимоги до надійності

Вся інформація про користувачів системи буде зберігатись у базі даних MySQL, застосунок буде розгорнутий у контейнері Docker.

Захист БД забезпечується за допомогою приватної мережі, тобто підключення до бази даних можливе лише користувачами та сервісами, які мають доступ до цієї мережі.

A.4.3 Умови експлуатації

Код програми буде розміщений у приватному репозиторії GitHub і може бути наданий певному колу людей.

Експлуатація програми здійснюється згідно експлуатаційної документації на розроблену програму, яка відповідає стандартам і містить інформацію, необхідну для її освоєння та експлуатації.

A.4.4 Вимоги до складу та параметрів технічний засобів

Для експлуатації клієнтської частини необхідні такі програмно-технічні засоби: будь-який сучасний браузер та стабільне підключення до мережі Інтернет.

Серед вимог до серверу виділяють:

- операційна система: Linux;
- процесор з двома ядрами і з тактовою частотою не менше 2 ГГц;
- оперативна пам'ять: 8 Гбайт;
- встановлене програмне забезпечення Docker;
- дисковий простір: SSD 25 Гбайт.

A.4.5 Вимоги до маркування і пакування

Готовий програмний продукт поставляється на сервер у вигляді образу Docker. Серверна частина його розгортає та продукт готовий до роботи з ним.

На впакуванні повинна бути назва програми – «Travel Helper».

A.4.6 Вимоги до транспортування й зберігання

Програмний код застосунку зберігається у GitHub. Для транспортування коду на інший персональний комп'ютер його необхідно завантажити з сервісу GitHub.

A.4.7 Вимоги до програмної документації

Програмний продукт повинен поставлятися у наступному складі:

- технічне завдання;
- опис програми;
- текст програми.

A.5 Стадії та етапи розробки

Виділяють наступні етапи розробки програмного забезпечення:

Етап 1: Початок проєкту.

Фаза ініціації проєкту – це перший етап перетворення абстрактної ідеї на значущу мету. На цьому етапі вам необхідно розробити бізнес-кейс і визначити проєкт на широкому рівні. Для цього необхідно визначити потребу в проєкті та створити статут проєкту.

Етап 2: Планування проєкту.

Етап планування проєкту вимагає повної ретельності, оскільки викладає дорожню карту проєкту. На цьому етапі основними завданнями є визначення технічних вимог, розробка детального розкладу проєкту, створення комунікаційного плану та встановлення цілей/результатів.

Етап 3: Виконання проєкту.

На етапі виконання проєкту ваша команда виконує фактичну роботу. Як керівник проєкту, ваша робота полягає в тому, щоб налагодити ефективні робочі процеси та уважно стежити за прогресом вашої команди.

Етап 4: Моніторинг та контроль проєкту.

Фаза моніторингу та контролю проєкту виконується одночасно з виконанням проєкту, таким чином забезпечуючи досягнення цілей та результатів проєкту.

На етапі моніторингу управління проєктом менеджер також відповідає за кількісне відстеження зусиль і витрат під час процесу. Це відстеження не

тільки гарантує, що проєкт залишається в межах бюджету, але також важливо для майбутніх проєктів.

Етап 5: Закриття проєкту.

Це завершальний етап процесу управління проєктом. Етап закриття проєкту вказує на завершення проєкту після остаточної поставки.

Кінцевим завданням цього етапу є розгляд всього проєкту, завершення детального звіту, який охоплює кожен аспект. Усі необхідні дані зберігаються в захищеному місці, до якого можуть отримати доступ керівники проєктів цієї організації.

А.6 Порядок контролю та приймання системи

Приймання системи здійснюється відповідно до наступного порядку.

Звіт про проведену роботу направляється керівникові дипломної роботи д.т.н., професору Олійнику А.О. Нормоконтроль за оформленням пояснювальної записки виконує асистент Липовець М.В. Представлення та захист результатів роботи виконується в присутності екзаменаційної комісії.

Вимоги до оформлення проміжних звітів і підсумкового звіту узгоджуються із науковим керівником дипломної роботи, нормо контролером та завідуючим кафедрою програмних засобів.

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

Тема дипломної кваліфікаційної роботи бакалавра - Програмна реалізація застосунку для трекінгу туристичних маршрутів.

Метою роботи є розробка системи для допомоги мандрівникам. Вона забезпечує пошук маршрутів, додавання (редагування/видалення) особистих маршрутів, рейтингування маршрутів, калькуляцію розрахунків запасів їжі.

Б.2 Функціональне призначення

Даний проєкт – призначений для туристів та дозволяє спростити пошук та вибір маршрутів у наступну подорож. Створений ресурс, дозволить легко моніторити особисті та популярні маршрути. За допомогою цього проєкту легко знайти або створити нові напрямки для походів та розрахувати необхідну кількість провізії.

Створення даного продукту – призначене для допомоги мандрівникам, побачити свою країну. Цей застосунок допоможе знайти найцікавіші місця та маршрути, а також – розрахувати необхідну кількість провізії на цей шлях.

Кожен користувач зможе передивлятись списки маршрутів. Крім того зареєстровані користувачі зможуть користуватися аналітичною складовою, яка розрахована на рейтингування та представлення популярних маршрутів місяця або міста користувача, а також надання рекомендованих маршрутів, в залежності від досвіду користувача.

Б.3 Опис логічної структури програми

Структурно застосунок поділений на серверну частину, в тому числі API для роботи з базою даних, клієнтську частину та утиліти для роботи з мапою.

На рисунку Б.1 наведено структурну схему ПЗ, що розробляється.

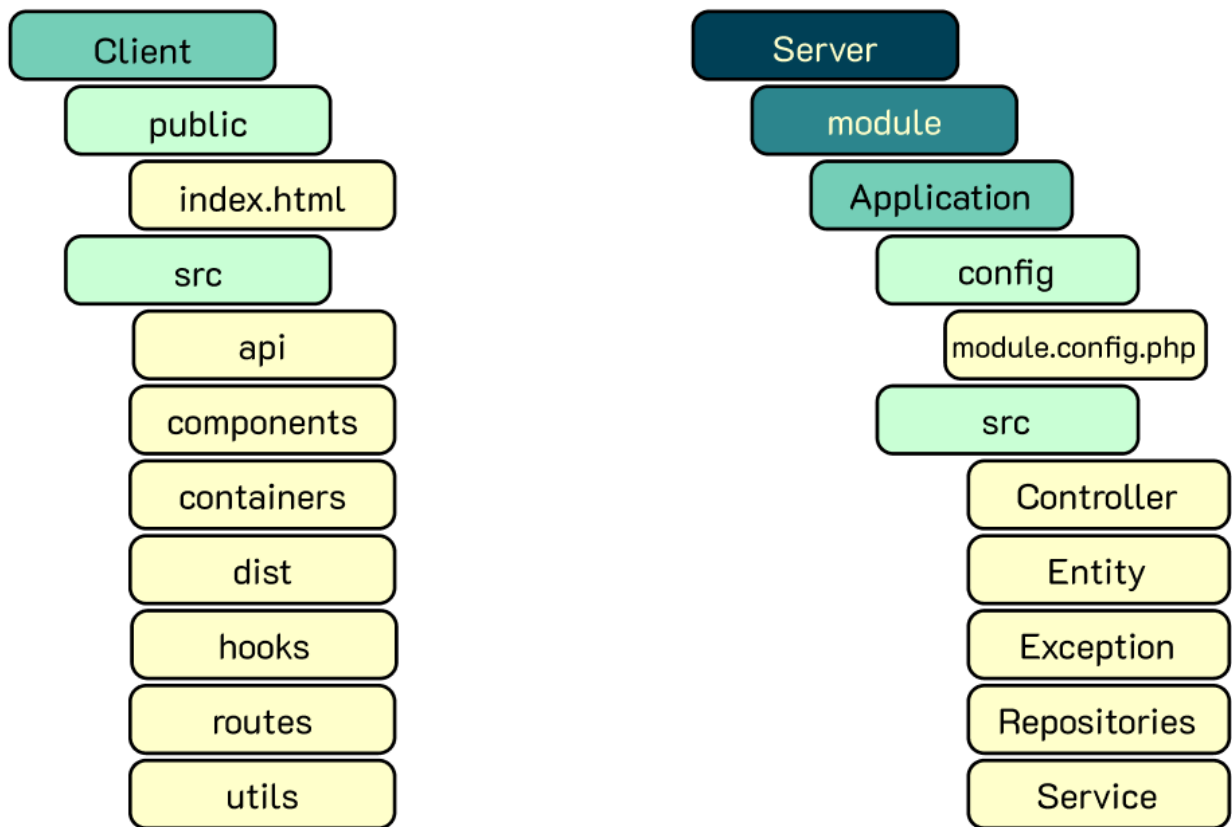


Рисунок Б.1 – Загальна структурна схема системи

Б.4 Використані технічні засоби

Для реалізації серверної частини було обрано мову програмування PHP, через її ефективність, гнучкість, швидкість виконання коду та підтримка великої кількості баз даних. У якості програмних надбудов було обрано Laminas Framework, через його універсальність та об'єктно-орієнтований підхід програмування.

Для реалізації клієнтської частини було обрано мову програмування JavaScript. З такими надбудовами: React, TypeScript та Material-ui, через їх ефективність та гнучкість.

При розробці ПЗ для моніторингу та рейтингування маршрутів, для побудови маршрутів було використано такі технічні засоби як:

- Leaflet – використовується для відображення карт;

- React Leaflet – обгортка для Leaflet карт для зручнішого використання їх у React;
- TomTom –сервіс онлайн маршрутизації;
- Leaflet Routing Machine – використовується для поєднання React Leaflet та TomTom і розширює можливості Leaflet мап, такими функціями як встановлення, видалення та зміна міток на мапі.

У якості патерну було обрано MVC.

Б.5 Виклик і завантаження

Для розгортання застосунку необхідно мати на ПК встановлені PHP 8.1, Apache2, платформа Node.js версії 14.17.0 або вище, завантажити код з GitHub репозиторію.

Для запуску проєкту локально, необхідно встановити залежності за допомогою пакетного менеджера npm. Для цього потрібно перейти в директорію клієнтської частини програми в терміналі та виконати команду `npm install`, після чого виконати команду `npm start` – виконання цього скрипту запустить програму, яка буде доступна за адресою – `http://localhost:3000/`.

Для запуску серверної частини застосунку необхідно перейти в директорію серверної частини програми в терміналі та виконати команду: `php -S 127.0.0.1:3000 index.php`.

Також потрібно додати змінні оточення. Для цього створюємо файл `.env`. Приклад файлу знаходиться в директорії клієнтської частини - `.env.example`.

- `REACT_APP_API_HOST_PROTOCOL` - хост протокол бекенд частини програми (`http/https`);
- `REACT_APP_API_URL` - url бекенд частини програми;
- `REACT_APP_API_PORT` – порт бекенда;
- `REACT_APP_TOM_TOM_API_KEY` - апі ключ вебсервісу маршрутизації `tomtom`.

– REACT_APP_GEONAMES_USERNAME - ім'я користувача для запитів до АПІ для отримання списку країн.

Б.6 Початкові дані

Вхідними даними до програмного забезпечення для трекінгу туристичних маршрутів є база даних, яка зберігає інформацію про користувачів та усі маршрути. Вхідними даними також є дані, введені користувачем з клавіатури та отримані дані з серверу, кнопки, завдяки яким є можливість обрати відповідний пункт меню та отримати відповідну інформацію.

Вихідними даними є графічне представлення маршрутів на картах.

ДОДАТОК В
Текст програми

```

        <?php

/*
 * (c) Copyright 2022. Prokopieva
 */

namespace Application\Controller;

use Application\Exception\JwtAuthConfigException;
use Application\Library\ParsUrl;
use Application\Service\EngineService;
use Application\Service\UserService;
use Application\Service\UserSettingsService;
use Doctrine\ORM\EntityManager;
use Firebase\JWT\JWT;
use Laminas\Http\PhpEnvironment\Request;
use Laminas\Http\PhpEnvironment\Response;
use Laminas\Mvc\Controller\AbstractRestfulController;
use Psr\Container\ContainerExceptionInterface;
use Psr\Container\ContainerInterface;
use Psr\Container\NotFoundExceptionInterface;

/**
 * @OA\Info(
 *   title = "Application module",
 *   description = "Application controllers",
 *   version = "1.0.0",
 * ),
 *
 * @OA\Server(
 *   url = "http://localhost:3000/",
 *   description = "local server"
 * ),
 *
 * @OA\SecurityScheme(
 *   securityScheme="bearerAuth",
 *   type="http",
 *   scheme="bearer",
 *   bearerFormat="JWT",
 * ),
 *
 *
 * Class AbstractApiController
 * @method Response getResponse()
 * @method Request getRequest()
 *
 * @property Response $response
 * @property Request $request
 */
abstract class AbstractApiController extends AbstractRestfulController
{
    use ParsUrl;

    /**
     * @var ContainerInterface ServiceManager
     */
    protected ContainerInterface $serviceManager;

    /** @var EntityManager|null */
    protected $entityManager;

    /**
     * AbstractApiController constructor.
     */

```

```

        *
        * @param EntityManager|null $entityManager
        */
    public function __construct($entityManager)
    {
        $this->entityManager = $entityManager;
    }

    /**
     * @return ContainerInterface
     */
    public function getServiceManager()
    {
        return $this->serviceManager;
    }

    /**
     * @param ContainerInterface $serviceManager
     *
     * @return $this
     */
    public function setServiceManager(ContainerInterface $serviceManager)
    {
        $this->serviceManager = $serviceManager;
        return $this;
    }

    /**
     * @param array $payload
     *
     * @return string
     * @throws JwtAuthConfigException
     * @throws ContainerExceptionInterface
     * @throws NotFoundExceptionInterface
     */
    protected function generateJwtToken(array $payload): string
    {
        $config = $this->serviceManager->get('config');

        if (isset($config['jwtAuth'])) {
            $cypherKey = $config['jwtAuth']['cypherKey'];
            $tokenAlgorithm = $config['jwtAuth']['tokenAlgorithm'];
            $token = JWT::encode($payload, $cypherKey, $tokenAlgorithm);
        } else {
            throw new JwtAuthConfigException();
        }

        return $token;
    }
}

<?php
/*
 * (c) Copyright 2022. Prokopieva
 */

namespace Application\Controller;

use Application\Controller\Plugin\UsersPlugin;
use Application\Entity\Users;
use Application\Exception\DataAuthWasNotProvidedException;
use Application\Exception\JwtAuthConfigException;
use Application\Exception>LoginAuthWasNotProvidedException;
use Application\Exception>PasswordAuthWasNotProvidedException;

```

```

use Application\Exception\UserDoesNotExistException;
use Application\Service\AuthManager;
use Application\Service\TokenPayloadService;
use Doctrine\ORM\EntityManager;
use Exception;
use Laminas\View\Model\JsonModel;
use Psr\Container\ContainerExceptionInterface;
use Psr\Container\NotFoundExceptionInterface;

/**
 *
 * @OA\Tag(name = "AuthController")
 *
 * Class AuthController
 *
 * @method UsersPlugin user()
 */
class AuthController extends AbstractApiController
{
    /**
     * @var AuthManager
     */
    private $authManager;

    /**
     * @var TokenPayloadService
     */
    private $tokenPayloadService;

    /**
     * Constructor.
     *
     * @param EntityManager $entityManager
     * @param AuthManager $authManager
     * @param TokenPayloadService $tokenPayloadService
     */
    public function __construct(
        EntityManager $entityManager,
        AuthManager $authManager,
        TokenPayloadService $tokenPayloadService
    ) {
        parent::__construct($entityManager);

        $this->authManager = $authManager;
        $this->tokenPayloadService = $tokenPayloadService;
    }

    /**
     *
     * @OA\Post(
     *     path = "/",
     *     description = "create token",
     *     operationId = "create",
     *     tags = {"AuthController"},
     *     security = {
     *         { "bearerAuth" = {} }
     *     }
     * )
     *
     * @param mixed $data
     *
     * @return JsonModel
     * @throws DataAuthWasNotProvidedException
     * @throws LoginAuthWasNotProvidedException

```

```

* @throws PasswordAuthWasNotProvidedException
* @throws UserDoesNotExistException
* @throws ContainerExceptionInterface
* @throws NotFoundExceptionInterface
* @throws JwtAuthConfigException
*/
public function create($data): JsonModel
{
    if (empty($data)) {
        throw new DataAuthWasNotProvidedException();
    }

    if (empty($data['login'])) {
        throw new LoginAuthWasNotProvidedException();
    }

    if (empty($data['password'])) {
        throw new PasswordAuthWasNotProvidedException();
    }

    $login = $data['login'];
    $password = $data['password'];

    /** @var Users|null $user */
    $user = $this->authManager->login($login, $password);

    if (empty($user)) {
        throw new UserDoesNotExistException();
    }

    $payload = $this->tokenPayloadService->getPayload($user);

    $result = [
        'token' => $this->generateJwtToken($payload),
    ];

    return new JsonModel($result);
}

/**
 * @OA\GET(
 *     path = "/logout",
 *     description = "logout",
 *     operationId = "logout",
 *     tags = {"AuthController"},
 *     security = {
 *         { "bearerAuth" = {} }
 *     },
 * )
 * @return JsonModel
 * @throws Exception
 */
public function logoutAction(): JsonModel
{
    $this->authManager->logout();

    return new JsonModel();
}
}

<?php
/*
 * (c) Copyright 2022. Prokopieva
 */

```

```

namespace Application\Controller;

use Application\Controller\Plugin\UsersPlugin;
use Application\Entity\Users;
use Application\Repositories\RUsers;
use Doctrine\ORM\Exception\ORMException;
use Doctrine\ORM\OptimisticLockException;
use Laminas\Http\PhpEnvironment\Request;
use Laminas\View\Model\JsonModel;

/**
 *
 * @OA\Tag(name = "UserController")
 *
 * Class UserController
 *
 * @method Request getRequest()
 * @method UsersPlugin user()
 */
class UserController extends AbstractApiController
{
    /**
     * @OA\Get(
     *     path = "/user/{id}",
     *     description = "get",
     *     operationId = "getList",
     *     tags = {"UserController"},
     *     security = {
     *         { "bearerAuth" = {} }
     *     }
     * )
     * @param $id
     * @return JsonModel
     */
    public function get($id): JsonModel
    {
        /** @var RUsers $userRepository */
        $userRepository = $this->entityManager->getRepository(Users::class);
        $user = $userRepository->findOneBy(['id' => $id]);

        $result['userName'] = $user->getName();
        $result['level'] = $user->getExpLevelId();

        return new JsonModel($result);
    }

    /**
     * @OA\Put (
     *     path = "/user/{id}",
     *     description = "put",
     *     operationId = "update",
     *     tags = {"UserController"},
     *     security = {
     *         { "bearerAuth" = {} }
     *     }
     * )
     * @param $id
     * @param $data
     * @return JsonModel
     * @throws ORMException
     * @throws OptimisticLockException
     */
    public function update($id, $data)
    {

```

```

    /** @var RUsers $userRepository */
    $userRepository = $this->entityManager->getRepository(Users::class);
    $user = $userRepository->findOneBy(['id' => $id]);
    $user
        ->setName($data['firstName'])
        ->setSurname($data['lastName'])
        ->setCountry($data['country'])
        ->setState($data['state'])
        ->setCity($data['city']);
    $this->entityManager->persist($user);
    $this->entityManager->flush();

    $result['status'] = "Success";
    $result['id'] = $id;

    return new JsonModel($result);
}
}

```

```
<?php
```

```

/*
 * (c) Copyright 2022. Prokopieva
 */

```

```
namespace Application\Controller;
```

```

use Application\Controller\Plugin\UsersPlugin;
use Application\Entity\Users;
use Application\Exception\DataWasNotProvidedException;
use Application\Exception>LoginIsNotUniqueException;
use Application\Repositories\RUsers;
use Application\Validator\UserValidator;
use Doctrine\ORM\Exception\ORMException;
use Doctrine\ORM\NonUniqueResultException;
use Doctrine\ORM>NoResultException;
use Doctrine\ORM\OptimisticLockException;
use Exception;
use Laminas\Crypt\Password\Bcrypt;
use Laminas\Http\PhpEnvironment\Request;
use Laminas\View\Model\JsonModel;
use Psr\Container\ContainerExceptionInterface;
use Psr\Container\NotFoundExceptionInterface;

```

```
/**
```

```

 *
 * @OA\Tag(name = "UserSignUpController")
 *
 * Class UserSignUpController
 *
 * @method Request getRequest()
 * @method UsersPlugin user()
 */

```

```
class UserSignUpController extends AbstractApiController
```

```

{
    /**
     * @OA\Post(
     *     path = "/auth/sign-up",
     *     description = "Create a new user",
     *     operationId = "create",
     *     tags = {"UserSignUpController"}
     * )
     *
     * @param $data
     * @return JsonModel
     */
}

```

```

* @throws DataWasNotProvidedException
* @throws LoginIsNotUniqueException
* @throws NoResultException
* @throws NonUniqueResultException
* @throws ORMException
* @throws OptimisticLockException
* @throws ContainerExceptionInterface
* @throws NotFoundExceptionInterface
*/
public function create($data): JsonModel
{
    if (empty($data)) {
        throw new DataWasNotProvidedException();
    }
    $config = $this->getServiceManager()->get('configuration');
    if (empty($config['mail'])) {
        throw new Exception($this->translate('Missing mail config'), 400);
    }

    $validator = new UserValidator($this->entityManager);
    $validatedData = $validator->validateUserData($data,
$config['mail']['validator']['regex']);

    $login = $validatedData['login'];
    $email = $validatedData['email'];
    $fullName = $validatedData['name'];
    $roleId = $validatedData['roleId'] ?? null;
    $password = $validatedData['password'];
    $level = $validatedData['level'];
    $country = $validatedData['country'];

    /** @var RUsers $userRepository */
    $userRepository = $this->entityManager->getRepository(Users::class);
    $loginUnique = $userRepository->isUniqueLogin($login);
    if ($loginUnique) {
        throw new LoginIsNotUniqueException();
    }
    $bcrypt = new Bcrypt();
    $user = new Users();
    $user->setLogin($login);
    $user->setName($fullName);
    $user->setEmail($email);
    $user->setPassword($bcrypt->create($password));
    $user->setCountry($country);
    $user->setRoleId($roleId);
    $user->setExpLevelId($level);

    $this->entityManager->persist($user);
    $this->entityManager->flush();
    $userId = $user->getId();

    $result = $userRepository->getUser($userId);

    return new JsonModel($result);
}
}

<?php
/*
 * (c) Copyright 2022. Prokopieva
 */

namespace Application\Controller;

```

```

use Application\Controller\Plugin\UsersPlugin;
use Application\Entity\Routes;
use Application\Exception\DataWasNotProvidedException;
use Application\Repositories\RRoutes;
use Doctrine\ORM\Exception\ORMException;
use Doctrine\ORM\OptimisticLockException;
use Exception;
use Laminas\Http\PhpEnvironment\Request;
use Laminas\View\Model\JsonModel;

/**
 * @OA\Tag(name = "RouteController")
 *
 * Class RouteController
 *
 * @method Request getRequest()
 * @method UsersPlugin user()
 */
class RouteController extends AbstractApiController
{
    public function get($id): JsonModel
    {
        /** @var RRoutes $routesRepository */
        $routesRepository = $this->entityManager->getRepository(Routes::class);
        $isExist = $routesRepository->findOneBy(["id" => $id]);
        if ($isExist !== null)
            $result = $routesRepository->getRoute($id);
        else
            throw new Exception('Route does not exist', 404);

        return new JsonModel($result);
    }

    /**
     * @throws OptimisticLockException
     * @throws ORMException
     * @throws DataWasNotProvidedException
     */
    public function create($data): JsonModel
    {
        $currentUserId = $this->user()->userId();

        if (empty($data)) {
            throw new DataWasNotProvidedException();
        }

        $route = new Routes();
        $route
            ->setName($data['name'])
            ->setDescription($data['description'])
            ->setTravelMode($data['travelMode'])
            ->setCountry($data['country'])
            ->setExpLevel($data['expLevel'])
            ->setOwnerId($currentUserId)
            ->setTotalDistance($data['summary']['totalDistance'])
            ->setTotalTime($data['summary']['totalTime'])
            ->setWaypoints(json_encode($data['waypoints']));

        $this->entityManager->persist($route);
        $this->entityManager->flush();
        $routeId = $route->getId();

        /** @var RRoutes $routesRepository */

```

```

        $routesRepository = $this->entityManager->getRepository(Routes::class);
        $result = $routesRepository->getRoute($routeId);

        return new JsonModel($result);
    }

    public function update($id, $data): JsonModel
    {
        $currentUserId = $this->user()->userId();

        if (empty($data)) {
            throw new DataWasNotProvidedException();
        }

        /** @var RRoutes $routesRepository */
        $routesRepository = $this->entityManager->getRepository(Routes::class);
        $route = $routesRepository->findOneBy(['id' => $id]);

        $route
            ->setName($data['name'])
            ->setDescription($data['description'])
            ->setTravelMode($data['travelMode'])
            ->setCountry($data['country'])
            ->setExpLevel($data['expLevel'])
            ->setOwnerId($currentUserId)
            ->setTotalDistance($data['summary']['totalDistance'])
            ->setTotalTime($data['summary']['totalTime'])
            ->setWaypoints(json_encode($data['waypoints']));

        $this->entityManager->persist($route);
        $this->entityManager->flush();

        $result = $routesRepository->getRoute($id);

        return new JsonModel($result);
    }

    public function delete($id): JsonModel
    {
        /** @var RRoutes $routesRepository */
        $routesRepository = $this->entityManager->getRepository(Routes::class);
        $routesRepository->deleteRoute($id);

        $result['status'] = "Success";

        return new JsonModel($result);
    }
}

<?php
/*
 * (c) Copyright 2022. Prokopieva
 */

namespace Application\Service;

use Doctrine\ORM\EntityManager;
use Interop\Container\ContainerInterface;
use Laminas\Authentication\AuthenticationService;
use Laminas\Json\Json;
use Laminas\Stdlib\AbstractOptions;
use Laminas\Validator\ValidatorInterface;
use RuntimeException;

```

```

abstract class AbstractService
{
    /**
     * @var ContainerInterface ServiceManager
     */
    protected $serviceManager;

    /** @var EntityManager */
    protected $entityManager;

    /** @var IdentityService */
    protected $identityService;

    /**
     * Authentication service.
     * @var AuthenticationService $authService
     */
    private $authService;

    /** @var AbstractOptions */
    protected $options;

    /** @var null|ValidatorInterface */
    protected $validator;

    protected $jsonDecodeType = true;

    /**
     * AbstractService constructor.
     *
     * @param EntityManager $entityManager
     */
    public function __construct($entityManager)
    {
        $this->entityManager = $entityManager;
    }

    /**
     * @param iterable $options
     *
     * @return void
     */
    public function setOptions(iterable $options): void
    {
        $this->options->setFromArray($options);
    }

    /**
     * Validate option if validator is defined
     */
    public function validateOptions(): void
    {
        if (null !== $this->validator) {
            if ($this->validator instanceof ValidatorInterface) {
                $this->validator->isValid($this->options);
            }
        }
    }

    /**
     * @return ContainerInterface ServiceManager
     */

```

```

public function getServiceManager(): ContainerInterface
{
    return $this->serviceManager;
}

/**
 * @param ContainerInterface $serviceManager
 * @return $this
 */
public function setServiceManager(ContainerInterface $serviceManager): self
{
    $this->serviceManager = $serviceManager;
    return $this;
}

/**
 * @return AuthenticationService
 */
public function getAuthService()
{
    if (null === $this->authService) {
        $this->setAuthService($this->getServiceManager()-
>get(AuthenticationService::class));
    }

    return $this->authService;
}

/**
 * @param AuthenticationService $authService
 * @return $this
 */
public function setAuthService(AuthenticationService $authService)
{
    $this->authService = $authService;
    return $this;
}

/**
 * @return IdentityService
 */
public function getIdentityService()
{
    if (null === $this->identityService) {
        $this->setIdentityService($this->getServiceManager()-
>get(IdentityService::class));
    }

    return $this->identityService;
}

/**
 * @param IdentityService $identityService
 * @return $this
 */
public function setIdentityService(IdentityService $identityService)
{
    $this->identityService = $identityService;
    return $this;
}

public function jsonDecode($string)
{
    if (class_exists(Json::class)) {

```

```

        return Json::decode($string, (int)$this->jsonDecodeType);
    }
    // @codeCoverageIgnoreStart
    throw new RuntimeException(sprintf(
        'Unable to parse JSON request, due to missing ext/json and/or %s',
        Json::class
    ));
    // @codeCoverageIgnoreEnd
}

<?php
/*
 * (c) Copyright 2022. Prokopieva
 */

namespace Application\Service;

use Application\Entity\Users;
use Application\Repositories\RUsers;
use Doctrine\ORM\OptimisticLockException;
use Doctrine\ORM\ORMException;
use Laminas\Authentication\Adapter\AdapterInterface;
use Laminas\Authentication\Result;
use Laminas\Crypt\Password\Bcrypt;

class AuthAdapter extends AbstractService implements AdapterInterface
{
    /**
     * User login.
     * @var string
     */
    private $login;

    /**
     * Password
     * @var string
     */
    private $password;

    /**
     * Sets user login.
     * @param string $login
     */
    public function setLogin($login): void
    {
        $this->login = $login;
    }

    /**
     * Sets password.
     * @param string $password
     */
    public function setPassword($password): void
    {
        $this->password = (string)$password;
    }

    /**
     * Performs an authentication attempt.
     * @return Result
     * @throws ORMException
     * @throws OptimisticLockException
     */
    public function authenticate(): Result

```

```

{
    // Check the database if there is a user with such login.
    /** @var RUsers $userRepository */
    $userRepository = $this->entityManager->getRepository(Users::class);
    $user = $userRepository->getUserByLogin($this->login);

    // If there is no such user, return 'Identity Not Found' status.
    $code = Result::FAILURE_IDENTITY_NOT_FOUND;
    $message = 'Invalid credentials.';
    if ($user !== null) {
        if ($this->changePassword($user)) {
            $code = Result::SUCCESS;
            $message = 'Authenticated successfully.';
        } else {
            // If password check didn't pass return 'Invalid Credential'
failure status.
            $code = Result::FAILURE_CREDENTIAL_INVALID;
            $user = null;
        }
    }
    $result = new Result(
        $code,
        $user,
        [$message]
    );
    return $result;
}

/**
 * @param Users $user
 * @return bool
 * @throws ORMException
 * @throws OptimisticLockException
 */
public function changePassword($user)
{
    $result = false;
    $bcrypt = new Bcrypt();

    if ($bcrypt->verify($this->password, $user->getPassword())) {
        $result = true;
    }
    return $result;
}
}
<?php
/*
 * (c) Copyright 2022. Prokopieva
 */

namespace Application\Service;

use Application\Entity\Users;

class AuthManager extends AbstractService
{
    /**
     * @param string $login
     * @param string $password
     * @return Users|null
     */
    public function login(string $login, string $password): ?Users
    {
        $authAdapter = $this->getAuthService()->getAdapter();

```

```

        /** @var AuthAdapter $authAdapter */
        $authAdapter->setLogin($login);
        $authAdapter->setPassword($password);

        return $this->getAuthService()->authenticate()->getIdentity();
    }

    /**
     * @return void
     * @throws \Exception
     */
    public function logout()
    {
        // Remove identity from session
        $this->getAuthService()->clearIdentity();
    }

    /**
     * @return mixed|null
     */
    public function getIdentity()
    {
        return $this->getAuthService()->getIdentity();
    }
}
<?php
/*
 * (c) Copyright 2022. Prokopieva
 */

namespace Application\Service;

/**
 * Class IdentityService
 */
class IdentityService
{
    /**
     * @var string
     */
    private $userLogin;

    /**
     * @var int
     */
    private $userId;

    /**
     * @param int $userId
     * @param string $userLogin
     */
    public function setIdentity(
        int $userId,
        string $userLogin,
    ): void {
        $this->userId = $userId;
        $this->userLogin = $userLogin;
    }

    /**
     * @return string
     */
    public function getUserLogin()

```

```

    {
        return $this->userLogin;
    }

    /**
     * @return int
     */
    public function getUserId()
    {
        return $this->userId;
    }
}

// work with map

import React, { useRef } from 'react';
import L, { LatLng, LeafletMouseEvent } from 'leaflet';
import { useMap } from 'react-leaflet';
import { createControlComponent } from '@react-leaflet/core';
import 'leaflet-routing-machine';
import 'dist/lrm-tomtom.min.js';
import { some, size } from 'lodash';

import geoJSONdata from 'dist/world.json';

function createButton(label: string, container: HTMLElement) {
    const button = L.DomUtil.create('button', '', container);

    button.setAttribute('type', 'button');
    button.innerHTML = label;

    return button;
}

const useCreateRoutineMachineLayer = ({
    onChangeWaypoints,
    onChangeRoute,
    travelMode,
    isPreview,
    isDisabled,
    waypoints,
    isInit,
    country
}: any): L.Control => {
    const map = useMap();

    const isInitRef = useRef<boolean>(isInit);

    // set view of the map to fit all waypoints
    if (size(waypoints)) {
        if (isPreview || isDisabled || isInitRef.current) {
            map.fitBounds(L.latLngBounds(waypoints));
            isInitRef.current = false;
        }
    } else {
        const geoJSONgroup = L.geoJSON(geoJSONdata, {
            style: () => ({
                fillOpacity: 0,
                opacity: 0
            })
        }).addTo(map);

        geoJSONgroup.eachLayer(function (layer) {

```

```

        if (layer.feature.properties.name === country) {
            map.fitBounds(layer.getBounds());
        }
    });
}

const controlRef = React.useRef(L.Routing.control({
    router: new L.Routing.TomTom(process.env.REACT_APP_TOM_TOM_API_KEY, {
        travelMode: travelMode // car, truck, taxi, bus, van, motorcycle,
bicycle, pedestrian
    }),
    lineOptions: {
        styles: [
            {
                color: "blue",
                opacity: 0.6,
                weight: 4
            }
        ]
    },
    routeWhileDragging: !(isPreview || isDisabled),
    addWaypoints: !(isPreview || isDisabled),
    draggableWaypoints: !(isPreview || isDisabled),
    fitSelectedRoutes: false,
    showAlternatives: false,
    createMarker: (waypointIndex: number, waypoint: L.Routing.Waypoint) => {
        const marker = L.marker(waypoint.latLng, {
            draggable: !(isPreview || isDisabled)
        });

        if (isPreview || isDisabled) {
            return marker;
        }

        marker.on('click', (event: LeafletMouseEvent) => {
            const container = L.DomUtil.create('div');
            const removeButton = createButton('Remove marker', container);

            removeButton.classList.add('leaflet-popup-content-remove-btn');

            L.DomEvent.on(removeButton, 'click', () => {
                controlRef.current.spliceWaypoints(waypointIndex, 1);
                map.closePopup();
            });

            L.popup()
                .setContent(container)
                .setLatLng(event.latLng)
                .openOn(map);
        });

        return marker;
    }
}));

React.useEffect(() => {
    controlRef.current.getRouter().options.travelMode = travelMode;
    controlRef.current.route();
}, [travelMode]);

React.useEffect(() => {
    controlRef.current.setWaypoints(waypoints);
}, [waypoints]);

```

```

map.on('click', function(event: LeafletMouseEvent) {
  if (isPreview || isDisabled) {
    return null;
  }

  const container = L.DomUtil.create('div');
  const startButton = createButton('Start location', container);
  const finishButton = createButton('Finish location', container);

  L.DomEvent.on(startButton, 'click', function() {
    controlRef.current.spliceWaypoints(0, 1, event.latlng);
    map.closePopup();
  });

  L.DomEvent.on(finishButton, 'click', function() {
    controlRef.current.spliceWaypoints(controlRef.current.getWaypoints().length -
    1, 1, event.latlng);
    map.closePopup();
  });

  L.popup()
    .setContent(container)
    .setLatLng(event.latlng)
    .openOn(map);
});

if (isPreview) {
  map._handlers.forEach((handler) => {
    handler.disable();
  });

  map._controlCorners.topright.style.display = 'none';
}

controlRef.current.on('waypointschanged', (updatedWaypoints) => {
  const isSomeValidWaypoints = some(updatedWaypoints.waypoints, (waypoint) =>
  waypoint.latlng);

  if (onChangeWaypoints && updatedWaypoints && isSomeValidWaypoints) {
    const latLngArray: Array<LatLng> = [];

    updatedWaypoints.waypoints.forEach((waypoint: Record<string, any>) => {
      if (waypoint.latlng) {
        latLngArray.push(waypoint.latlng)
      }
    });

    onChangeWaypoints(latLngArray);
  }
});

controlRef.current.on('routesfound', (event) => {
  if (onChangeRoute && event?.routes[0]?.summary) {
    onChangeRoute(event.routes[0].summary);
  }
});

return controlRef.current;
};

const RoutingMachine = createControlComponent(useCreateRoutineMachineLayer);

export default RoutingMachine;

```

ДОДАТОК Д
Слайди презентації

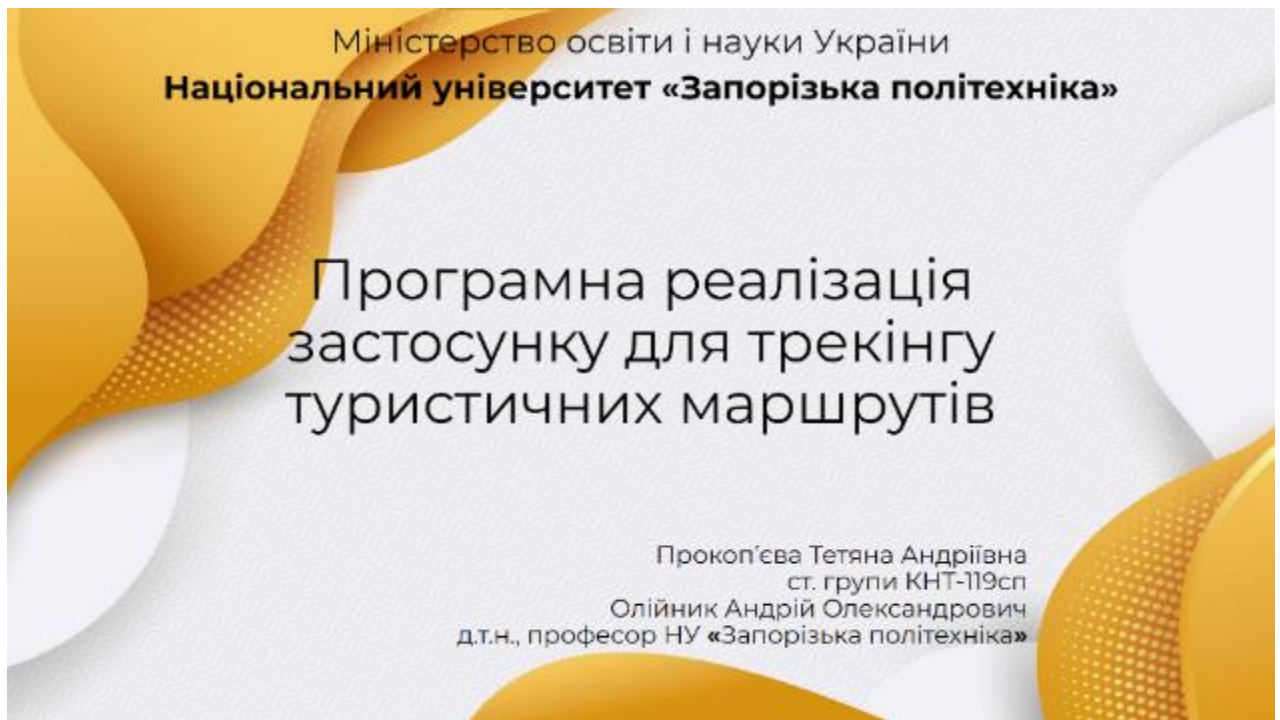


Рисунок Д.1 – Слайд 1

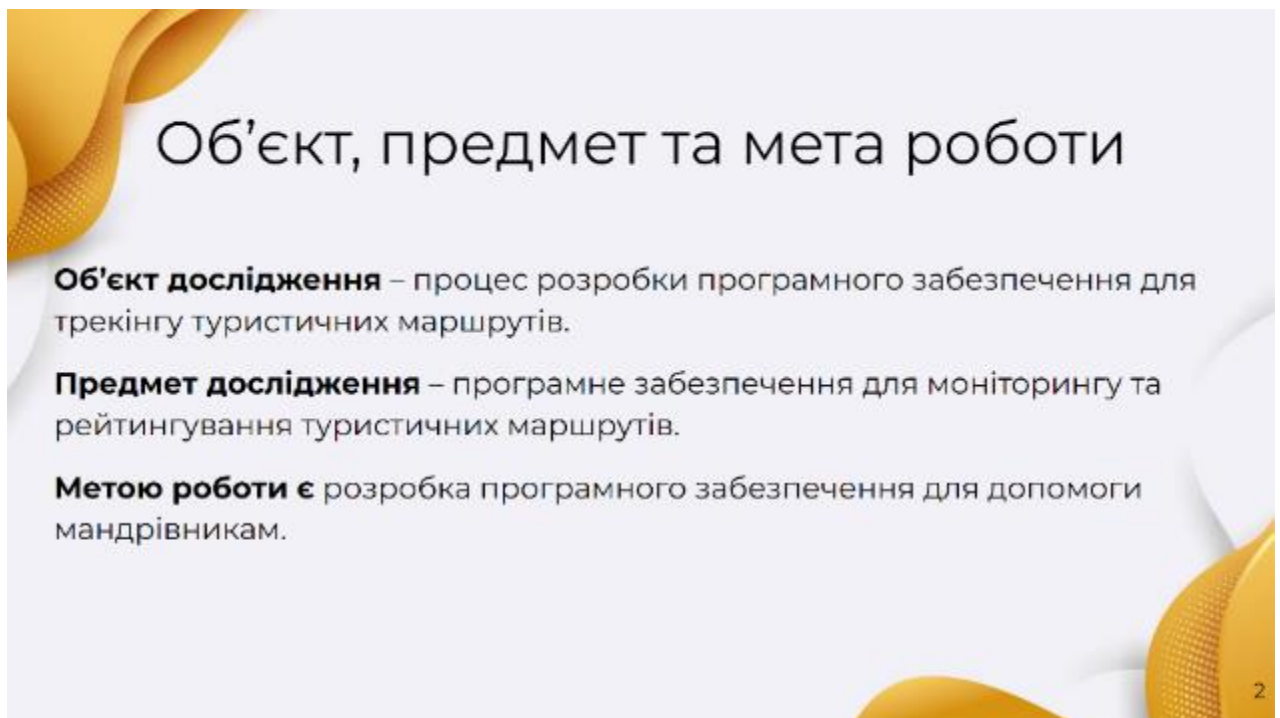


Рисунок Д.2 – Слайд 2

Порівняння існуючих аналогів

Критерій порівняння	Strava	В Похід Карпатами	Wikiloc	Мій застосунок
Фільтрація маршрутів	–	+	+	+
Аналітика, рейтингування маршрутів	–	–	–	+
Калькуляція провізії	–	–	–	+
Безкоштовний доступ	–	+	–	+

Рисунок Д.3 – Слайд 3

Постановка завдання

За результатами аналізу проблемної області та дослідження існуючих аналогів для вирішення проблемних питань – потрібно розробити новий застосунок «Travel Helper». Система повинна реалізувати такі функціональні вимоги:

- реєстрація та авторизація в системі;
- пошук маршрутів;
- перегляд повного списку маршрутів;
- перегляд рекомендованих маршрутів;
- додавання (редагування/видалення) особистих маршрутів;
- рейтингування маршрутів;
- фільтрація маршрутів;
- калькуляція розрахунків запасів їжі.

Рисунок Д.4 – Слайд 4

Порівняння мов програмування

Критерій	Java	Python	PHP
Продуктивність	+	–	++
Вимогливість до ресурсів системи	+	–	+
Зрозумілість синтаксису	+	–	+

Рисунок Д.5 – Слайд 5

Порівняння фреймворків

Критерій	Yii2	Laravel	Symphony	Laminas
Підтримка	–	–	+	+
Частота оновлення	–	+	+	++
ООП	+	–	–	+

Рисунок Д.6 – Слайд 6

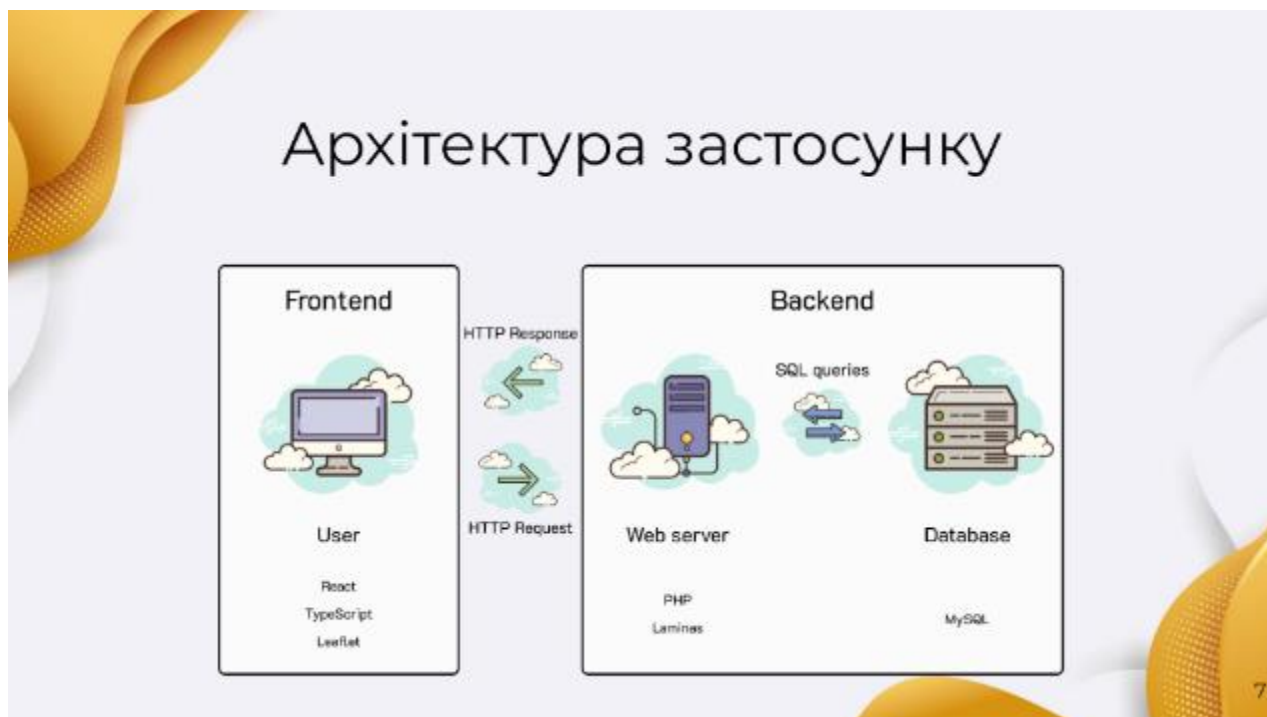


Рисунок Д.7 – Слайд 7

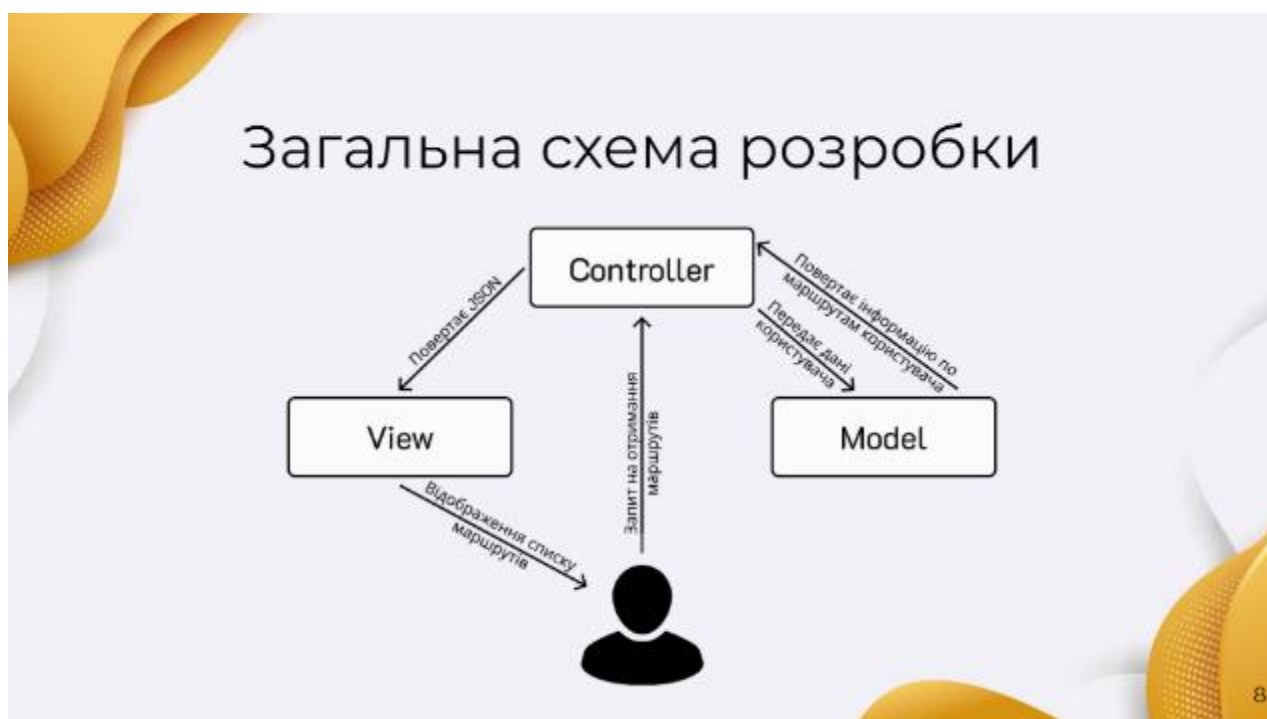


Рисунок Д.8 – Слайд 8



Рисунок Д.9 – Слайд 9



Рисунок Д.10 – Слайд 10

Діаграма кооперацій



Рисунок Д.11 – Слайд 11

Робота з сайтом. Сторінка маршрутів

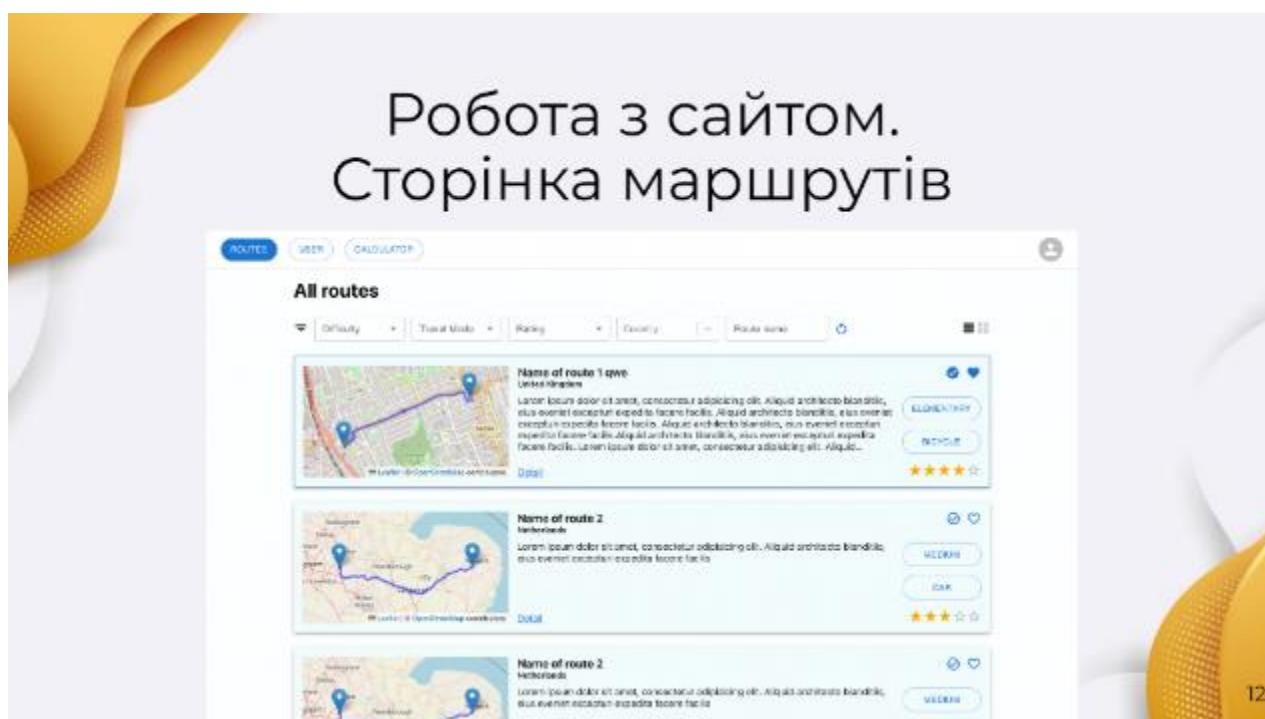


Рисунок Д.12 – Слайд 12

Робота з сайтом. Авторизація та реєстрація

The image displays two side-by-side mockups of a website's authentication interface. The left mockup is for 'Sign In' and includes a 'Login' field, a 'Password' field, a 'SIGN IN' button, and a link 'I DON'T HAVE AN ACCOUNT'. The right mockup is for 'Sign Up' and includes a 'Login' field, a 'Password' field, a 'Repeat Password' field, and a 'NEXT' button. Both forms feature an illustration of a person on a bicycle.

Рисунок Д.13 – Слайд 13

Робота з сайтом. Створення маршруту

The image shows a 'Create Route' interface. It includes a map with a blue route line and two blue location pins. Above the map are input fields for 'Route name' and 'Name', and dropdown menus for 'Difficulty' (set to 'Professional') and 'Travel Mode' (set to 'Bicycle'). Below the map is a text area for 'Description' with the placeholder 'description of the route'. A 'CREATE' button is at the bottom right.

Рисунок Д.14 – Слайд 14

Методика та результати тестування

Для того аби перевірити роботоспроможність застосунку були проведені наступні види тестування:

- **функціональне тестування.** Перевірка функцій та характеристик ПЗ, що розробляється, на основі проєктної документації;
- **юзабіліті-тестування.** Мета даного виду тестування полягає у визначенні ступеня зручності та практичності користувацького інтерфейсу;
- **тестування сумісності.** Перевірка можливості програми взаємодіяти з різними програмними продуктами, операційними системами та оточенням.

15

Рисунок Д.15 – Слайд 15

Основні перевірки функціоналу

Перевірка	Google Chrome	Safari
Реєстрація та авторизація	+	+
Авторизація неіснуючого користувача	+	+
Перегляд списку маршрутів	+	+
Фільтрація та пошук маршрутів	+	+
Додавання маршрутів в обране	+	+
Позначення маршруту пройденим	+	+
Оцінка пройденого маршруту	+	+
Підрахунок провізії	+	+

16

Рисунок Д.116 – Слайд 16

Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для трекінгу туристичних маршрутів.

Розроблено вебзастосунок для моніторингу та рейтингування туристичних маршрутів. Метою роботи є розробка програмного забезпечення для допомоги мандрівникам.

Застосунок забезпечує виконання таких функцій:

- реєстрація та авторизація в системі;
- додавання (редагування/видалення) особистих маршрутів;
- пошук маршрутів;
- рейтингування маршрутів;
- перегляд повного списку маршрутів;
- фільтрація маршрутів;
- перегляд рекомендованих маршрутів;
- калькуляція розрахунків запасів їжі.

Результати тестування програмного забезпечення для трекінгу туристичних маршрутів показали, що розроблений вебзастосунок працює коректно та може використовуватися за призначенням.

17

Рисунок Д.17 – Слайд 17

Дякую за увагу

18

Рисунок Д.18 – Слайд 18