МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії
програмного забезпечення

_____ Євген ДАВИДЕНКО

«___» _____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

Вебзастосунок планування подорожей

Спеціальність 121 Інженерія програмного забезпечення
Освітня програма «Інженерія програмного забезпечення»

Здобувач _____ **Олександр БАРАНОВ**

«_» _____ 2026 р.

Керівник роботи
PhD, доцентка (б.в.з.)

_____ **Катерина АНТІПОВА**

«_» _____ 2026 р.

Миколаїв – 2026

Завдання на виконання кваліфікаційної бакалаврської роботи

Чорноморський національний університет імені Петра Могили

Факультет	Комп'ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступінь	Бакалавр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
«___» _____ 2026 р.

ЗАВДАННЯ

на кваліфікаційну бакалаврську роботу здобувача

Баранова Олександра

1. Тема кваліфікаційної роботи «Вебзастосунок планування подорожей» затверджена наказом ректора ЧНУ ім. Петра Могили № 349 від « 26 » грудня 2025 р.
2. Строк представлення кваліфікаційної роботи «___» _____ 2026 р.
3. Очікуваний результат роботи: Вебзастосунок для планування подорожей.
4. Перелік питань, що підлягають розробці:

– провести аналіз існуючих програмних рішень для планування подорожей та визначити їхні переваги та недоліки;

– провести аналіз предметної сфери для визначення ключових вимог до системи;

– розробити архітектуру вебзастосунку, що дозволить ефективно обробляти дані та взаємодіяти з користувачем;

– виконати проєктування та моделювання основних компонентів вебзастосунку;

– реалізувати модуль генерації маршрутів, що враховує особисті вподобання користувача;

– інтегрувати систему бронювання житла та транспорту за допомогою арі зовнішніх сервісів;

– розробити інтуїтивний та адаптивний інтерфейс користувача для зручного використання вебзастосунку;

– виконати тестування розробленого вебзастосунку та аналіз його ефективності.

5. Перелік графічних матеріалів: презентація, діаграма класів.

6. Консультанти:

Консультант	**Кафедра (організація)**	**Частина роботи**

Дата видачі завдання « 12 » січня 2026 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної бакалаврської роботи

Тема: **«Вебзастосунок планування подорожей»**

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КБР	01.01.2026	12.01.2026	Виконано
2.	Огляд літератури за темою роботи	13.01.2026	19.01.2026	Виконано
3.	Складання календарного плану КБР	20.01.2026	21.01.2026	Виконано
4.	Аналіз предметної області	22.01.2026	30.01.2026	Виконано
5.	Розробка проєктних рішень	02.02.2026	09.02.2026	Виконано
6.	Моделювання та конструювання ПЗ	10.02.2026	16.02.2026	Виконано
7.	Розробка структури бази даних та міграцій до неї	17.02.2026	23.02.2026	Виконано
8.	Кодування ПЗ для API	24.02.2026	16.03.2026	Виконано
9.	Кодування ПЗ для клієнтської частини	17.03.2026	06.04.2026	Виконано
10.	Тестування ПЗ	07.04.2026	20.04.2026	Виконано
11.	Відгук керівника КБР	21.04.2026	27.04.2026	Виконано
12.	Оформлення КБР та презентації	04.05.2026	22.05.2026	Виконано
13.	Попередній захист	26.05.2026	26.05.2026	Виконано
14.	Рецензування			
15.	Завершення оформлення КБР та презентації			
16.	Захист кваліфікаційної роботи			

Здобувач _____ **Олександр БАРАНОВ**

«26» січня 2026 р.

Керівник роботи

PhD, доцентка

(б.в.з.) _____ **Катерина АНТІПОВА**

«26» січня 2026 р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи

«Вебзастосунок планування подорожей»

Здобувач 409 гр.: Олександр Баранов

Керівник: PhD, доцентка (б.в.з.) Катерина Антіпова

Зростання популярності самостійного планування подорожей спричинене розвитком цифрових сервісів та потребою користувачів у персоналізованих маршрутах. Водночас наявні рішення є фрагментованими й вимагають взаємодії з різними платформами для бронювання житла, транспорту та пошуку розваг, що ускладнює організацію поїздки. Крім того, більшість існуючих застосунків не враховують комплексно бюджет, часові обмеження та індивідуальні вподобання (дієтичні, інклюзивність, історію пошуку), а їхні інтерфейси часто не є адаптивними для мобільних пристроїв. Актуальність теми полягає у створенні єдиного вебзастосунку, який інтегрує всі етапи планування подорожі, автоматизує підбір маршрутів, бронювання послуг та враховує індивідуальні вподобання користувача, що дозволяє підвищити ефективність та зручність підготовки до поїздок. Крім того, сучасний ринок туристичних послуг демонструє стійкий попит на інтелектуальні системи, здатні аналізувати великі обсяги даних про транспорт, житло та визначні місця в реальному часі. Розробка такого застосунку також сприяє вирішенню проблеми «інформаційного перевантаження», коли користувач витрачає занадто багато часу на порівняння альтернатив і координацію різних сервісів.

Метою кваліфікаційної роботи є розробка вебзастосунку планування подорожей з інтеграцією зовнішніх сервісів, що дозволить підвищити зручність та ефективність самостійного формування маршрутів для користувачів.

Об'єктом кваліфікаційної роботи є процес планування подорожей користувачами в онлайн-середовищі.

Предметом кваліфікаційної роботи є інструменти для розробки вебзастосунку для автоматизованого планування подорожей. У роботі використано методи структурного та об'єктно-орієнтованого аналізу.

У вступі сформульовано актуальність, мету, об’єкт, предмет і завдання роботи.

У першому розділі проаналізовано сучасні застосунки, виявлено їхню фрагментарність та обмежену адаптацію до уподобань користувачів, сформульовано вимоги: інтуїтивний інтерфейс, варіативність маршрутів, інтеграція з API бронювання.

У другому розділі обґрунтовано вибір стеку технологій (Angular 19, Nest.js, PostgreSQL, Redis, Leaflet) та розроблено специфікацію вимог.

Третій розділ присвячено проєктуванню архітектури, створенню UML-діаграм і моделюванню компонентів.

У четвертому розділі описано реалізацію: модуль оптимізації маршрутів, інтеграцію OpenStreetMap, Booking.com і Travelpayouts API, адаптивний інтерфейс на Angular 19 та Tailwind CSS, кешування Redis. Проведено тестування, підтверджено працездатність системи, розроблено керівництво користувача.

Ключові слова: *вебзастосунок, планування подорожей, API, Angular, NestJS, PostgreSQL.*

ABSTRACT

for the bachelor's qualification work

"Web Application for Travel Planning"

Student of Group 409: Oleksandr Baranov

PhD, Associate Professor (w/o a.d.) Kateryna Antipova

The growing popularity of independent travel planning is driven by the development of digital services and users' need for personalized itineraries. At the same time, existing solutions are fragmented and require interaction with different platforms for booking accommodation, transport, and finding attractions, which complicates trip organization. Moreover, most existing applications do not comprehensively take into account budget, time constraints, and individual preferences (dietary needs, inclusivity, search history), and their interfaces are often not adaptive for mobile devices. The relevance of the topic lies in creating a single web application that integrates all stages of trip planning, automates route selection, service booking, and takes into account user preferences, thereby increasing the efficiency and convenience of trip preparation. Furthermore, the current travel services market demonstrates a steady demand for intelligent systems capable of analyzing large amounts of real-time data on transport, accommodation, and points of interest. The development of such an application also helps solve the problem of "information overload," where users spend too much time comparing alternatives and coordinating various services.

The aim of the bachelor's qualification work is to develop a travel planning web application with the integration of external services, which will increase the convenience and efficiency of independent route creation for users.

The object of the bachelor's qualification work is the process of travel planning by users in an online environment.

The subject is the tools for developing a web application for automated travel planning. The work employs methods of structural and object-oriented analysis.

The introduction formulates the relevance, aim, object, subject and tasks of the work.

The first chapter analyzes modern applications, identifies their fragmentation and limited adaptation to user preferences, and formulates the requirements: an intuitive interface, itinerary variability, and integration with booking APIs.

The second chapter substantiates the technology stack (Angular 19, Nest.js, PostgreSQL, Redis, Leaflet) and develops the requirements specification.

The third chapter is devoted to architecture design, creation of UML diagrams and component modeling.

The fourth chapter describes the implementation: a route optimization module, integration of OpenStreetMap, Booking.com and Travelpayouts APIs, a responsive interface using Angular 19 and Tailwind CSS, and Redis caching. Testing was carried out, confirming the system's operability, and a user manual was developed.

Keywords: *API, Angular, NestJS, PostgreSQL, travel planning, web application.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API	Application Programming Interface
CRUD	Create, Read, Update, Delete
HTTP	Hyper Text Transfer Protocol
HTTPS	HTTP Secure
JWT	JSON Web Token
MVC	Model-View-Controller
OAuth	Open Authorization
QA	Quality Assurance
REST	Representational State Transfer
SPA	Single-Page Application
SQL	Structured Query Language
UI	User Interface
UML	Unified Modeling Language
UX	User Experience

ВСТУП

У сучасному світі подорожі стали невід'ємною частиною життя людей, як у професійній діяльності, так і в особистих цілях. З розвитком цифрових технологій з’явилася потреба у створенні зручних вебзастосунків для планування поїздок, що дозволяють користувачам оптимізувати маршрути, знаходити найкращі варіанти транспорту та житла, а також отримувати інформацію про культурні та туристичні об'єкти. Наявні програмні рішення часто мають обмежені функціональні можливості або не враховують індивідуальні вподобання користувачів.

Розробка вебзастосунку для планування подорожей є актуальною задачею, оскільки вона дозволить спростити процес організації подорожей, підвищити ефективність використання ресурсів та забезпечити персоналізований підхід до планування маршрутів. Науково-практичне значення роботи полягає у створенні комплексного рішення, що інтегрує дані з різних джерел, використовує алгоритми рекомендацій та враховує особисті вподобання мандрівників.

Метою кваліфікаційної роботи є розробка вебзастосунку планування подорожей з інтеграцією зовнішніх сервісів, що дозволить підвищити зручність та ефективність самостійного формування маршрутів для користувачів.

Відповідно до мети визначено такі завдання:

– провести аналіз існуючих програмних рішень для планування подорожей та визначити їхні переваги та недоліки;

– провести аналіз предметної сфери для визначення ключових вимог до системи;

– розробити архітектуру вебзастосунку, що дозволить ефективно обробляти дані та взаємодіяти з користувачем;

– виконати проєктування та моделювання основних компонентів вебзастосунку;

– реалізувати модуль генерації маршрутів, що враховує особисті вподобання користувача;

– інтегрувати систему бронювання житла та транспорту за допомогою api зовнішніх сервісів;

– розробити інтуїтивний та адаптивний інтерфейс користувача для зручного використання вебзастосунку;

– виконати тестування розробленого вебзастосунку та аналіз його ефективності.

Об’єктом кваліфікаційної роботи є процес планування подорожей користувачами в онлайн-середовищі.

Предметом кваліфікаційної роботи є інструменти для розробки вебзастосунку для автоматизованого планування подорожей.

Розробка нового вебзастосунку дозволить усунути зазначені недоліки, впровадивши сучасні технології аналізу даних та інтеграції API.

Розроблений вебзастосунок стане ефективним інструментом для оптимізації подорожей, підвищення комфорту та економії ресурсів користувачів.

1 АНАЛІЗ СУЧАСНОГО СТАНУ СФЕРИ ПЛАНУВАННЯ ПОДОРОЖЕЙ

1.1 Опис предметної області

Сектор планування подорожей є одним із найбільш динамічних сегментів цифрової економіки. За даними Всесвітньої туристичної організації (UNWTO), у 2023 році міжнародний туризм відновився до 84–90% від рівня 2019 року, а за підсумками року кількість туристів сягнула майже 90% докризового показника [1]. Таке стрімке зростання попиту актуалізує потребу в сучасних цифрових інструментах для самостійного планування подорожей, оскільки дедалі більше мандрівників відмовляються від послуг традиційних турагентств на користь онлайн-сервісів.

Аналіз наявних вебзастосунків для планування подорожей дає змогу виокремити такі проблемні аспекти:

– відсутність комплексних екосистем, що інтегрували б усі складники поїздки (маршрутизацію, логістику, розміщення та дозвілля););

– обмеженість інструментів персоналізації рекомендацій та недостатня інтеграція із зовнішніми сервісами;

– недосконаліть алгоритмів оптимізації маршрутів відповідно до часових, бюджетних та індивідуальних критеріїв користувачів.

Самостійне планування подорожі є багатокроковим процесом, що потребує від користувача прийняття низки послідовних управлінських рішень в умовах опрацювання значних обсягів гетерогенної інформації. Цей процес як об'єкт автоматизації доцільно розділити на кілька ключових етапів, кожен з яких має специфічні функціональні вимоги та потреби в обробці даних.

На початковій стадії здійснюється формування параметрів подорожі, зокрема визначення бюджету, тривалості, складу учасників та їхніх пріоритетів. Наступним кроком є вибір ключових об'єктів маршруту: міст, пам'яток та природних локацій. Зараз цей процес базується на інтеграції даних із картографічних сервісів,

спеціалізованих туристичних ресурсів, соціальних мереж та рекомендацій, що зумовлює складність консолідації розрізненої інформації в межах єдиного плану та вимагає значних зусиль для ручної обробки.

Важливим етапом є логістичне планування, що передбачає оптимізацію послідовності відвідування обраних точок. Наразі користувачі змушені вирішувати задачу комівояжера самостійно, спираючись на суб'єктивні оцінки та враховуючи при цьому значну кількість змінних: географічну віддаленість, часові витрати, логістику транспортних сполучень та обмеження бюджету. Відсутність автоматизованих інструментів оптимізації спричиняє прокладання неефективних маршрутів, що зумовлює зростання фінансових та часових витрат.

Фінальним етапом планування є бронювання житла та транспортних засобів. Аналіз свідчить, що наразі цей процес реалізується через окремі сервіси (наприклад, Booking.com, Skyscanner або вебсайти авіакомпаній), які не інтегровані з уже сформованим маршрутом. Це вимагає багаторазового ручного введення ідентичних даних щодо дат, локацій та кількості осіб у кожну форму пошуку.

Отже, структурно-функціональний аналіз засвідчує фундаментальну проблему: сучасний процес планування подорожей є фрагментованим, трудомістким та недостатньо оптимізованим. Пріоритетними напрямками для підвищення ефективності є автоматизація логістичного етапу та інтеграція інструментів бронювання безпосередньо з робочим планом подорожі.

Ринкові тенденції (2023–2026). Сучасні споживачі туристичних послуг суттєво змінили свої пріоритети. Згідно з дослідженням Booking.com (2023), яке охопило понад 24 000 мандрівників у 32 країнах, 72% опитаних вважають, що подорожі завжди варті того, а 55% шукають «відпустки поза мережею» (off-grid), щоб втекти від реальності. Водночас 44% хочуть відчути «повернення до основ» (back-to-basics) – подорожі лише з найнеобхіднішим. Це свідчить про зростання попиту на якісне, продумане планування, яке враховує індивідуальні вподобання [2]. Наприклад, сучасні застосунки мають забезпечувати:

– дієтичні вподобання – автоматизований підбір закладів харчування з урахуванням вегетаріанського та веганського меню;

– інклюзивність – фільтрація локацій, адаптованих для осіб з обмеженими фізичними можливостями (наявність пандусів, ліфтів тощо);

– персоналізація рекомендації щодо відвідування музеїв, галерей чи вибору варіантів активного дозвілля на основі історії пошукових запитів користувача.

Водночас на ринку спостерігається функціональний дисбаланс: мобільні застосунки орієнтовані на оперативний доступ до інформації, але часто мають обмежений функціонал (зокрема, відсутність інструментів для детального бюджетування), тоді як вебплатформи пропонують розширені аналітичні можливості, проте не є достатньо адаптованими для мобільних інтерфейсів. Як наслідок, користувачі змушені постійно змінювати пристрої, що негативно впливає на ефективність процесу планування. Розроблюваний вебзастосунок вирішує цю проблему шляхом реалізації адаптивного дизайну, який забезпечує повноцінну роботу всіх інструментів незалежно від пристрою.

Згідно з даними Державної служби статистики України (2023), 72% українських користувачів надають перевагу застосункам, що підтримують інтерфейс державною мовою [3]. Водночас лише 11% міжнародних сервісів (таких як Wanderlog, TripIt) забезпечують локалізацію українською мовою. Це призводить до:

– складності використання. непрофесійні користувачі рідше залучаються до самостійного планування;

– помилок. неправильний переклад термінів (наприклад, "хостел" → "гуртожиток").

Крім мови, локалізація включає підтримку гривні, інтеграцію з українськими сервісами (наприклад, «Нова пошта» для доставки квитків) та врахування місцевих особливостей (наприклад, військові обмеження).

Цей аналіз показує, що розробка інтегрованого вебзастосунку з урахуванням українського контексту може суттєво спростити планування подорожей та підвищити їхню доступність.

1.2 Існуючі аналогічні рішення

Сучасний ринок замовлення онлайн подорожей пропонує різноманітні рішення, кожне з яких має свої особливості.

Система 1. TripIt

TripIt – це вебсервіс та мобільний застосунок для організації подорожей (рисунок 1.1). Його ключова функція – автоматичне створення єдиного маршруту на основі підтверджень бронювань, які користувач надсилає зі своєї електронної пошти (квитки на літак, бронювання готелів, оренду авто). Система аналізує дані та формує детальний план подорожі з часами, адресами та контактами. Застосунок синхронується з календарем (Google, Outlook) та надає сповіщення про зміни рейсів. Недолік – обмежена підтримка українських сервісів, наприклад, даних Укрзалізниці.

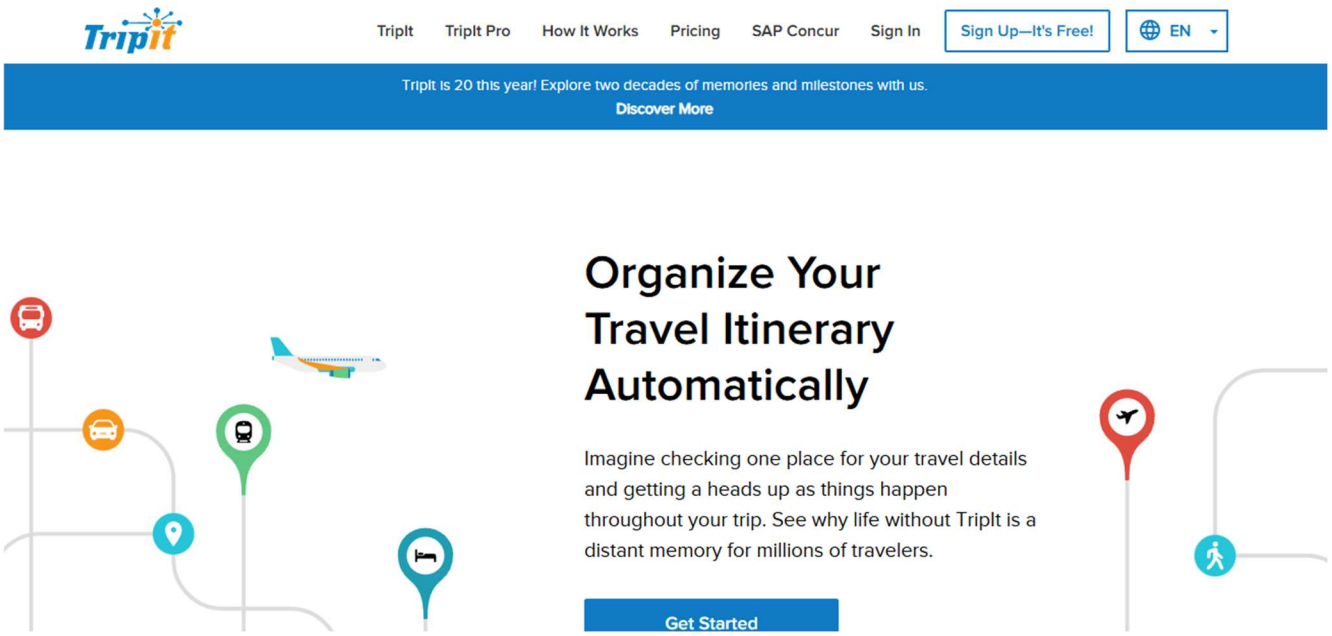

Рисунок 1.1 – Вебсервіс *TripIt*

Виробник: Concur Technologies.

Архітектура: Клієнт-серверна.

Мова реалізації: Java, JavaScript.

Основні функції:

– збереження маршрутів та подорожей;
– автоматичне синхронізування інформації з поштової скриньки;
– пошук готелів та рейсів;
– нагадування про події в реальному часі;
– відстеження витрат під час подорожі.

Переваги:

– простий інтерфейс;
– інтеграція з іншими сервісами (google calendar, outlook);
– офлайн-доступ.

Недоліки:

– більшість функцій доступна лише у платній версії;
– обмежена персоналізація.

Джерело інформації [4].

TripIt використовує складні алгоритми парсингу для вилучення структурованих даних (дати, час, локації, номери підтверджень) з неструктурованих листів-підтверджень від сотень різних постачальників послуг. Це є його ключовою технологічною перевагою, що дозволяє майже миттєво створювати детальний маршрут. Однак, подібний підхід робить систему надзвичайно залежною від форматів цих листів: будь-яка зміна в шаблоні авіакомпанії чи готелю може призвести до помилок парсингу, що вимагає постійної підтримки з боку розробників. Крім того, система реагує на вже здійснені бронювання, але не надає інструментів для проактивного планування з нуля.

Система 2. Skyscanner

Skyscanner – популярний агрегатор для пошуку авіаквитків, готелів та оренди авто (рисунок 1.2). Його головна перевага – порівняння цін від сотень авіакомпаній і турагенцій у режимі реального часу. Користувачі можуть шукати квитки з

гнучкими датами або навіть вибрати "Найдешевший місяць" для економії. Застосунок також показує альтернативні маршрути та надає прогнози зміни цін. Однак інтеграція з українськими лоукостерами (наприклад, SkyUp) обмежена.

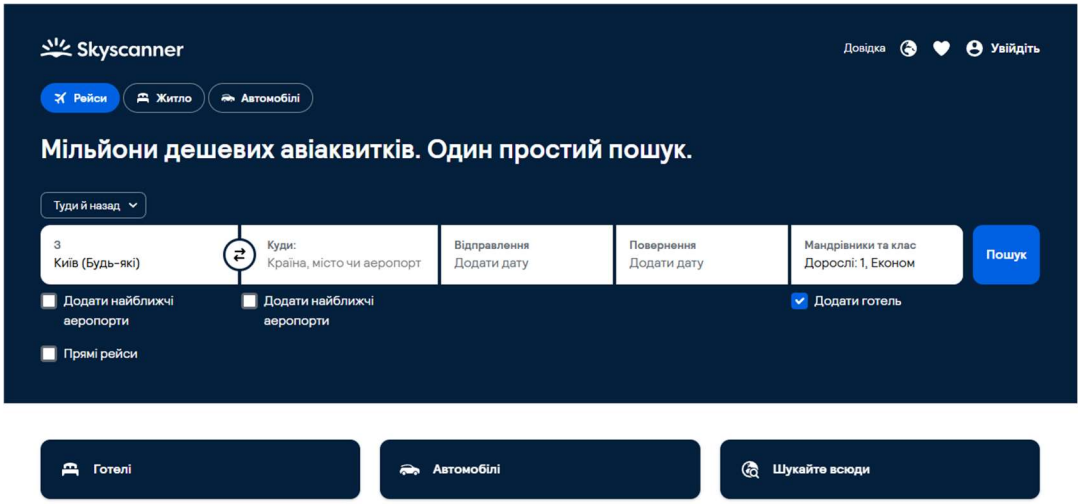

Рисунок 1.2 – Агрегатор *Skyscanner*

Виробник: Skyscanner Ltd.

Архітектура: Мікросервісна.

Мова реалізації: Python, JavaScript.

Основні функції:

– пошук авіаквитків, готелів та оренди автомобілів.
– порівняння цін;
– сповіщення про зміну вартості;
– інтерактивна мапа для вибору напрямків;
– відгуки користувачів.

Переваги:

– велика база даних партнерів;
– безкоштовне використання;
– можливість економії завдяки оповіщенням.

Недоліки:

– реклама партнерів;
– відсутність функції персонального планування.

Джерело інформації [5].

Основною відмінністю Skyscanner є його кешування та алгоритми прогнозування цін. Система не виконує запити в реальному часі до всіх авіакомпаній при кожному пошуку. Натомість, вона використовує попередньо закешовані дані та власні статистичні моделі, що дозволяє миттєво видавати результати. Функція «Найдешевший місяць» є прямим наслідком цієї архітектури. Проте, це також є джерелом обмежень: ціни не завжди є абсолютно точними в реальному часі, а деякі бюджетні авіакомпанії або специфічні маршрути можуть бути відсутніми в їхній базі даних. Skyscanner є чудовим інструментом для пошуку, але не для побудови повноцінного маршруту.

Система 3. Rome2Rio

Rome2Rio – сервіс для планування міжміських та міжнародних поїздок з комбінацією різних видів транспорту (літак, потяг, автобус, паром) (рисунок 1.3). Користувач вводить пункт відправлення та призначення, а система показує всі можливі варіанти з часом у дорозі, вартістю та посиланнями на бронювання. Наприклад, маршрут "Київ → Львів" може включати потяг, автобус або автомобіль з розрахунком витрат на пальне. Недолік – відсутність детальної інформації про локальний транспорт (наприклад, маршрутки в Україні).

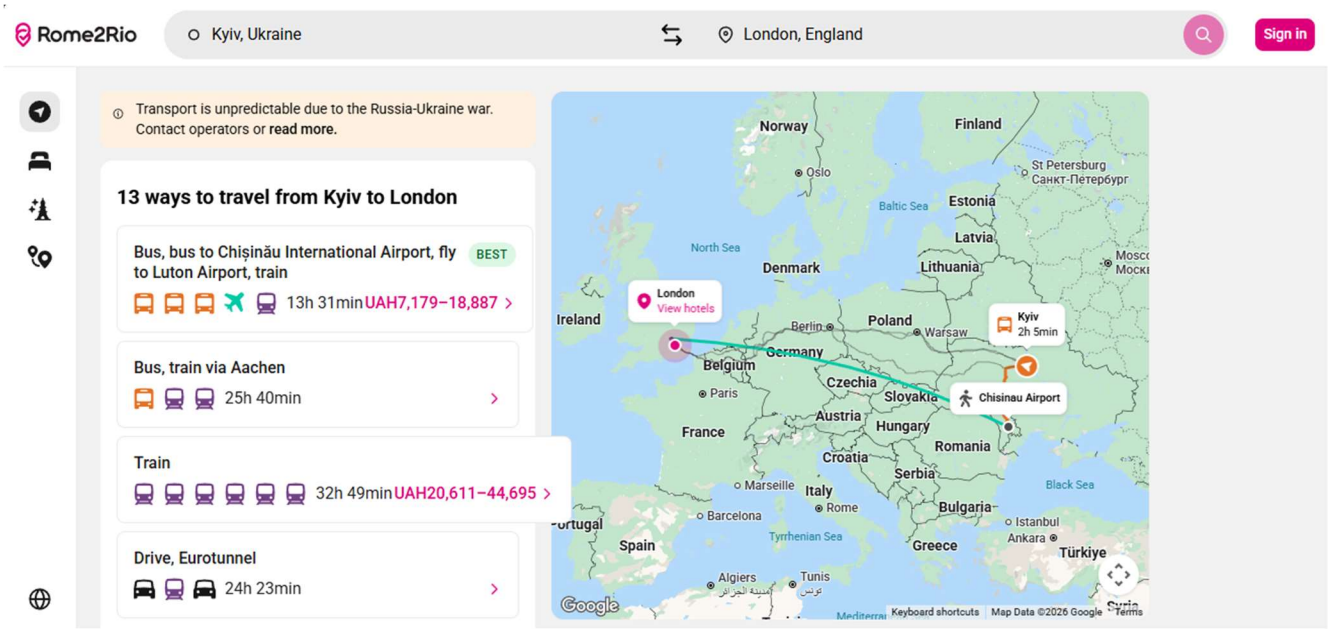

Рисунок 1.3 – Сервіс для планування міжміських та міжнародних поїздок

Rome2Rio

Виробник: Rome2Rio Pty Ltd.

Архітектура: Хмарна

Мова реалізації: JavaScript, Node.js

Основні функції:

– планування маршрутів з різними видами транспорту;
– інформація про витрати на подорож;
– деталізація часу подорожі;
– карта з маршрутами;
– інтеграція з платформами бронювання.

Переваги:

– зручність для складних маршрутів;
– глобальне покриття;
– простий інтерфейс.

Недоліки:

– неточність у деяких регіонах;
– лімітований офлайн-доступ.

Джерело інформації [6].

Сильна сторона Rome2Rio полягає в її графовій моделі даних. Вона представляє всю транспортну мережу як величезний граф, де вузли – це міста чи станції, а ребра – це можливі маршрути з різними видами транспорту та ваговими коефіцієнтами (час, ціна). Алгоритми пошуку найкоротшого шляху на такому графі дозволяють знаходити комплексні мультимодальні маршрути. Обмеженням є те, що система значною мірою покладається на розклади руху, які не завжди є точними або повними для локального транспорту, особливо в країнах, що розвиваються.

1.3 Особливості функціоналу вебзастосунку для планування подорожей

Проведений аналіз існуючих рішень (*TripIt, Skyscanner, Rome2Rio*) дозволив виявити ключові функціональні категорії, що визначають зручність та ефективність планування подорожей, а також встановити їхні обмеження, які має подолати новий вебзастосунок.

Побудова та візуалізація маршрутів. Усі розглянуті аналоги надають базову можливість формування послідовності точок. Наприклад, Rome2Rio дозволяє комбінувати різні види транспорту для міжміських переміщень, проте не надає інструментів для деталізації щоденного розкладу. TripIt автоматично створює маршрут із підтверджень бронювань, але не дозволяє гнучко додавати довільні точки інтересу. Жоден із сервісів не пропонує автоматичної оптимізації маршруту за критеріями часу або бюджету, що є суттєвим недоліком для користувачів, які прагнуть ефективного планування.

Управління бюджетом. Функціонал відстеження витрат у проаналізованих застосунках реалізовано фрагментарно. TripIt пропонує ручне введення витрат, однак не виконує автоматичного аналізу їх структури та не надає прогнозів щодо перевищення бюджету. Skyscanner, будучи агрегатором, взагалі не має засобів для обліку витрат. Таким чином, для користувачів, для яких бюджетування є важливим, виникає потреба у використанні сторонніх фінансових інструментів, що ускладнює загальний процес планування.

Бронювання житла та транспорту. Більшість рішень зосереджені на одному аспекті бронювання. Skyscanner – на авіаквитках, Booking – на житлі. Інтеграція обох функцій в єдиному інтерфейсі, де користувач може, не залишаючи застосунок, знайти готель для кожної точки маршруту та квитки між точками, є рідкісною. Така фрагментація змушує мандрівника перемикатися між різними платформами, втрачаючи час.

Підтримка групових подорожей. Функціонал спільного планування практично відсутній у розглянутих аналогах. TripIt дозволяє лише ділитися статичним планом, але не підтримує спільне редагування в реальному часі.

Відсутність вбудованих механізмів для узгодження маршрутів, розподілу витрат між учасниками та комунікації всередині групи є суттєвим недоліком.

Персоналізація та інтеграція з зовнішніми сервісами. Наявні продукти або зовсім не враховують індивідуальні вподобання користувача, або роблять це обмежено (наприклад, через історію пошуку в Skyscanner). Можливості для інтеграції з українськими сервісами (Укрзалізниця, BlaBlaCar UA), підтримки багатомовності та мультивалютності також реалізовані недостатньо, що створює додаткові труднощі для локальних користувачів.

Важливість врахування часового виміру в персоналізованих рекомендаціях підкреслюється в дослідженні Including the Temporal Dimension in the Generation [7]. Практичну реалізацію рекомендаційної системи для туристичних маршрутів описано в роботі Парасочкіна В. В. [8].

Узагальнення функціональних особливостей. Аналіз дозволяє сформувати перелік функцій, які мають бути реалізовані в новому вебзастосунку для подолання виявлених недоліків:

– комплексне планування в єдиному інтерфейсі, що об'єднує створення маршруту, пошук готелів та авіаквитків;

– автоматична оптимізація маршруту за допомогою геопросторових алгоритмів, що враховує реальну дорожню мережу;

– інтерактивна карта для додавання та редагування точок маршруту з автоматичним геокодуванням;

– прив'язка бронювання до дат подорожі, що дозволяє шукати готелі та квитки саме на потрібні дати, які зберігаються в плані;

– сучасний та адаптивний інтерфейс, що забезпечує зручну роботу як на десктопних, так і на мобільних пристроях.

Таким чином, визначені функціональні вимоги спрямовані на створення інтегрованого, персоналізованого та ефективного інструменту для самостійного планування подорожей, який усуває ключові обмеження існуючих програмних рішень.

1.4 Математичні моделі для ключових операцій

Математичні моделі є основою для реалізації алгоритмів, що забезпечують функціональність вебзастосунку. Нижче наведено моделі для ключових операцій системи.

1.4.1 Розрахунок оптимального маршруту

Мета: Знайти найефективніший маршрут між точками з урахуванням часу, вартості та складності.

$$f(n)=g(n)+h(n),$$

Модель:

Використано алгоритм A* з евристичною функцією, що враховує множники:

де:

– g(n)$g(n)$ – вартість шляху від початкової точки до вершини n;

– h(n)$h(n)$ – евристична оцінка вартості від вершини nn до кінцевої точки.

Параметри:

– g(n)=α·дистанція+β·час+γ·вартість$g(n)=\alpha\cdot$дистанція$+\beta\cdot$час$+\gamma\cdot$вартість,

де α,β,γα,β,γ – вагові коефіцієнти, які налаштовує користувач.

– h(n)$h(n)$ – пряма відстань між вершиною nn і кінцевою точкою.

Приклад:

Для маршруту Київ → Житомир → Львів:

– g(Житомир)=150 км·0.5+2 год·0.3+500 грн·0.2=75+0.6+100=175.6.
– h(Житомир) = 400 км (відстань до Львова).

1.4.2 Розрахунок бюджету подорожі

Мета: Визначити загальні витрати на основі обраного маршруту. **Модель**:

$$\text{Бюджет} = \sum_{i=0}^{n}\left(C_{transport_i} + C_{accommodation_i} + C_{food_i}\right),$$

де

– $C_{transport_i}$ – вартість транспорту для сегменту i;

– $C_{accommodation_i}$ – вартість проживання;

– C_{food_i} – витрати на харчування.

Приклад:

Для маршруту Київ → Одеса:

– Транспорт: 1200 грн (потяг);

– Проживання: 800 грн/добу × 3 доби = 2400 грн;

– Харчування: 300 грн/день × 4 дні = 1200 грн;

– Загальний

бюджет: 1200+2400+1200=4800 грн1200+2400+1200=4800грн.

Висновки до розділу 1

У першому розділі проведено комплексне дослідження предметної області, що охоплює сучасні підходи до планування подорожей, аналіз ринку, існуючих програмних рішень та потреб користувачів. Виявлено ключові тенденції: зростання популярності самостійного планування, попит на персоналізовані маршрути, важливість бюджетування та локалізації сервісів для українського ринку.

Порівняльний аналіз функціоналу аналогів (TripIt, Skyscanner, Rome2Rio) показав, що жодне з розглянутих рішень не забезпечує комплексного підходу до організації подорожі. Більшість із них зосереджені на окремих аспектах (пошук квитків, збір бронювань, мультимодальні маршрути), але не пропонують автоматичної оптимізації маршруту, гнучкого додавання точок інтересу та інтеграції з пошуком готелів і авіаквитків у єдиному інтерфейсі. Встановлено також відсутність належної підтримки українських сервісів та локалізації, що створює додаткові труднощі для вітчизняних користувачів.

Сформульовано ключові функціональні вимоги до нового вебзастосунку, серед яких: автоматична оптимізація маршрутів на основі реальної дорожньої мережі, інтерактивна карта з можливістю додавання точок кліком, прив’язка бронювання до дат подорожі, адаптивний інтерфейс та забезпечення безпеки даних.

Окрему теоретичну цінність має розроблений математичний апарат, який формалізує задачі оптимізації маршруту та розрахунку бюджету подорожі. Зокрема, запропоновано застосування алгоритму A* з ваговими коефіцієнтами для пошуку найефективнішого маршруту за критеріями часу, вартості та складності, а також модель обчислення загальних витрат на подорож. Ці моделі слугують теоретичним підґрунтям для подальшої програмної реалізації та забезпечують математичну обґрунтованість прийнятих проєктних рішень.

Аналіз ринкових тенденцій за 2023–2026 роки додатково виявив суттєвий дисбаланс між мобільними та вебрішеннями. Встановлено, що переважна більшість користувачів використовують мобільні пристрої для планування подорожей, однак саме вебверсії забезпечують необхідну глибину аналітики (детальне бюджетування, порівняння варіантів, кастомізацію маршрутів). Таким чином, обґрунтовано необхідність реалізації адаптивного вебзастосунку, який поєднує потужність десктопних інструментів зі зручністю мобільного інтерфейсу, що усуває потребу користувача в постійному перемиканні між пристроями.

Результати дослідження першого розділу лягли в основу технічного завдання на розробку системи. Визначено, що пріоритетними напрямками реалізації є: інтеграція із зовнішніми API для отримання актуальних даних про транспорт та житло, розробка власного модуля оптимізації маршрутів, створення зручного інтерфейсу для візуального планування та забезпечення можливості зберігання історії подорожей.

Таким чином, результати першого розділу не лише підтверджують актуальність створення інтегрованого вебзастосунку для планування подорожей, а й формують чіткий перелік вимог та математичний базис, які стануть основою для архітектурного проєктування та практичної реалізації системи в наступних розділах.

2 ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ

2.1 Загальні принципи моделювання вебзастосунку

Для реалізації вебзастосунку планується використання наступних технологій:

– *backend* – laravel (php) для реалізації серверної логіки та взаємодії з базою даних;

– *frontend* – angular для створення динамічного та інтерактивного інтерфейсу користувача;

– *база даних* – mysql або postgresql для збереження інформації про користувачів, маршрути та налаштування;

– *api інтеграція* – підключення до сервісів бронювання житла (booking, airbnb), транспорту (skyscanner) та інших ресурсів.

Результати дослідження та розробки можуть бути використані:

– туристичними компаніями для організації поїздок клієнтів;

– індивідуальними мандрівниками для самостійного планування подорожей;

– платформами для організації корпоративних поїздок та бізнес-зустрічей.

Моделювання є ключовим етапом у процесі розробки вебзастосунку, оскільки дозволяє сформувати цілісне уявлення про його архітектуру, логіку функціонування та взаємодію окремих компонентів. На цьому етапі здійснюється перехід від опису вимог до формалізованої структури системи, що забезпечує її коректне проєктування та реалізацію.

Вебзастосунок для планування подорожей є складною інформаційною системою, яка включає взаємодію з користувачем, обробку даних, інтеграцію з зовнішніми сервісами та надання персоналізованих рекомендацій. Для побудови моделі системи застосовуються підходи об’єктно-орієнтованого моделювання, зокрема використання стандарту UML (Unified Modeling Language), що дозволяє візуалізувати структуру, поведінку та взаємозв’язки між основними елементами.

Моделювання базується на таких принципах:

– *функціональна декомпозиція* – поділ системи на окремі логічні підсистеми, кожна з яких відповідає за визначений обсяг завдань (реєстрація, створення маршрутів, інтеграція з арі, бронювання, управління витратами тощо);

– *орієнтація на користувача* – побудова моделей з урахуванням зручності використання, сценаріїв взаємодії та персоналізації функціоналу;

– *інтеграційна відкритість* – передбачення механізмів взаємодії з зовнішніми сервісами для забезпечення актуальності інформації про транспорт, житло, події тощо;

– *масштабованість* – проєктування архітектури з урахуванням можливості подальшого розширення функціоналу або обробки великої кількості користувачів.

Результатом моделювання є побудова набору моделей, що відображають структуру та поведінку застосунку. Ці моделі слугують основою для реалізації системи, визначення взаємодії компонентів, а також допомагають мінімізувати ризики помилок на етапі програмної реалізації.

Таким чином, моделювання відіграє вирішальну роль у створенні надійного, функціонального та зручного вебзастосунку, який відповідає сучасним вимогам до систем планування подорожей.

2.2 Вибір стеку технологій розробки

Розробка вебзастосунку для планування подорожей базується на сучасних технологіях, вибір яких підтверджено аналізом наукових публікацій та практичних досліджень.

PostgreSQL обрано як основну систему керування базами даних через її високу продуктивність, надійність та розширюваність. Ключовим фактором стала наявність розширення PostGIS, яке перетворює PostgreSQL на повноцінну просторову базу даних, здатну ефективно зберігати геометричні об'єкти (точки, лінії, полігони) та виконувати просторові запити. Гібридний підхід, що поєднує

реляційну модель із можливостями NoSQL через JSON-поля, є оптимальним для складних вебсистем, як показано у дослідженні Б'єладиновича [9]. Використання JSON-полів дозволяє гнучко зберігати динамічні структури даних (наприклад, маршрути з різною кількістю точок), а ACID-властивості гарантують цілісність даних під час транзакцій, пов'язаних із бронюванням. Практичні аспекти роботи з просторовими даними в PostgreSQL детально висвітлено у книзі «PostGIS in Action» [10].

Angular 19 обрано як основний фреймворк для розробки фронтенду. Його компонентна архітектура, підтримка TypeScript та розвинена екосистема дозволяють створювати масштабовані односторінкові застосунки (SPA) з високою продуктивністю. Дослідження [11] демонструє, що застосування TypeScript знижує кількість критичних помилок на 25–35% порівняно з JavaScript. Практичні аспекти масштабування Angular-додатків висвітлено у збірнику проєктів [12]. Для побудови інтерактивних карт використано відкриту бібліотеку **Leaflet**, яка не потребує API-ключів для базового використання та забезпечує достатню функціональність для відображення маршрутів, маркерів та спливаючих вікон. Стилізація виконана за допомогою **Tailwind CSS**, що дозволило створити адаптивний дизайн без написання великих обсягів власного CSS.

NestJS використано як серверний фреймворк завдяки його модульній архітектурі, вбудованій підтримці TypeScript та інтеграції з TypeORM. Як зазначено в офіційній документації [13], NestJS дозволяє ізолювати критичні сервіси (автентифікація, робота з геоданими, інтеграція з зовнішніми API), що зменшує ризики системних збоїв та спрощує тестування. Для автентифікації застосовано Passport.js із стратегією JWT, що є стандартним підходом для захисту REST API.

Redis використовується для кешування результатів оптимізації маршрутів. Згідно з дослідженням оптимальних стратегій кешування для NestJS [14], застосування Redis дозволяє знизити затримки на 40–60% при повторних запитах, що є критичним для функції оптимізації, яка виконує ресурсоємні обчислення. Для

повнотекстового пошуку локацій у перспективі може бути використаний Elasticsearch, ефективність якого при обробці складних запитів показано в роботі [15].

Інфраструктура проекту базується на контейнеризації **Docker**. Досвід застосування Docker у мікросервісній архітектурі досліджено у роботі [16], а патерни багатоконтейнерної оркестрації за допомогою Docker Compose описані в [17].

Як альтернативний картографічний сервіс може розглядатися Google Maps API [18], що надає широкі можливості візуалізації, проте вимагає ліцензійного ключа. Для геопросторових сервісів обрано відкриті API OpenStreetMap (Nominatim для геокодування, OSRM для розрахунку відстаней), що усуває залежність від платних сервісів.

Проєктування REST API виконано відповідно до патернів і практик, узагальнених у посібнику [19]. За потреби система може бути розгорнута в хмарному середовищі, наприклад, згідно з рекомендаціями AWS Architecture Center [20].

Розширення функціоналу передбачає інтеграцію з додатковими сервісами бронювання, такими як Airbnb API [21], а також використання хмарних картографічних платформ, зокрема OpenStreetMap API [22] та Microsoft Azure Maps [23].

2.3 Специфікація вимог до програмного забезпечення

1) Призначення та межі проєкту.

1.1) Вебзастосунок «Travel Planner» призначений для автоматизації процесу планування подорожей. Система дозволяє користувачам створювати маршрути, додавати точки інтересу, оптимізувати порядок відвідування, шукати готелі та авіаквитки, інтегруючи ці функції в єдиному інтерфейсі. Головна мета – спростити організацію поїздки, уникнувши необхідності використовувати кілька окремих сервісів.

1.2) Інтерфейс системи реалізується українською мовою з перспективою подальшої локалізації. Обмін даними між клієнтською та серверною частинами здійснюється за допомогою REST API у форматі JSON. Для забезпечення безпеки та контролю доступу автентифікація користувачів виконується на основі JWT-токенів. Зберігання та обробка геопросторових даних покладаються на систему керування базами даних PostgreSQL із використанням розширення PostGIS. Щоб підвищити швидкодію системи, застосовується кешування на основі Redis, зокрема для збереження результатів оптимізації маршрутів.

1.3) У межах проєкту розробляється вебверсія застосунку з адаптацією під мобільні пристрої. Початковий набір інтеграцій включає Booking.com API (через RapidAPI) та Travelpayouts API для пошуку авіаквитків. На поточному етапі не реалізуються функціонал онлайн-чатів із підтримкою клієнтів та адміністративна панель. Окремий мобільний застосунок не розробляється; за потреби може бути створений PWA. Система не виконує прямого бронювання квитків, а надає посилання на партнерські сервіси.

2) Загальний опис.

2.1) Система розрахована на користувачів, які планують особисті або бізнес-подорожі, шукають інструменти для оптимізації маршрутів і бюджету, бажають об’єднати етапи планування (житло, транспорт) у єдиному інтерфейсі, а також цінують автоматизацію та доступ до актуальної інформації.

2.2) Основними користувачами є дорослі віком від 18 до 60 років – туристи, бізнесмени та мандрівники-ентузіасти, які володіють базовими навичками роботи з вебзастосунками та не потребують спеціальних технічних знань. У перспективі доцільно передбачити роль адміністратора для працівників туристичних агенцій або технічних спеціалістів, які матимуть права управління інтеграціями та контентом.

2.3) Система побудована за клієнт-серверною архітектурою. Клієнтська частина реалізована на Angular 19 (TypeScript) із використанням Tailwind CSS для стилізації та бібліотеки Leaflet для інтерактивних карт. Серверна частина виконана на NestJS (Node.js) і має модульну архітектуру, надаючи REST API. Для зберігання даних застосовано PostgreSQL із розширенням PostGIS для роботи з геопросторовими об’єктами. Кешування результатів оптимізації маршрутів здійснюється за допомогою Redis. До складу системи також входять інтеграції із зовнішніми сервісами – Booking.com API, Travelpayouts API та OpenStreetMap (Nominatim, OSRM).

2.4) Для коректної роботи системи необхідне стабільне інтернет-з’єднання, оскільки офлайн-режим для редагування маршрутів не передбачений. Підтримуються лише сучасні браузери (Chrome, Firefox, Safari, Edge останніх версій).

3) Функції системи.

3.1) Особистий кабінет та автентифікація:

3.1.1) Система надає користувачам можливість реєстрації, авторизації та управління власним профілем. Доступні операції зі створення облікового запису, вхід до системи, перегляд основних даних профілю (електронна пошта, дата реєстрації), зміна пароля та видалення акаунта.

3.1.2) Вхідна інформація – електронна пошта, пароль для реєстрації та входу, старий пароль, новий пароль, підтвердження нового пароля для зміни пароля, підтвердження дії для видалення акаунта. Вихідна інформація – JWT-токен доступу при успішній автентифікації, повідомлення про успішне виконання операції або текст помилки.

3.1.3) Паролі користувачів мають хешуватися алгоритмом bcrypt перед збереженням у базі даних. Усі маршрути, що потребують авторизації, захищаються шляхом валідації JWT-токену. На

клієнтській стороні обов’язковою є перевірка заповнення всіх полів, мінімальна довжина пароля (6 символів), а також збіг нового пароля та його підтвердження. Видалення акаунта спричиняє каскадне видалення всіх пов’язаних даних користувача (планів, маршрутів) із бази даних.

3.2) Управління планами подорожей

3.2.1) Користувач може створювати нові плани подорожей, переглядати список власних планів, а також видаляти існуючі. Кожен план характеризується назвою, датою початку подорожі та опціональним бюджетом.

3.2.2) Вхідна інформація – назва плану, дата початку, бюджет (опціонально) – для створення; ідентифікатор плану – для видалення. Вихідна інформація – список планів користувача (у форматі карток із назвою, датою, бюджетом); дані окремого плану; підтвердження успішної операції.

3.2.3) Кожен створений план прив’язується до облікового запису авторизованого користувача. Список планів відображається у вигляді карток із ключовими атрибутами, що забезпечує швидке орієнтування. Видалення плану вимагає явного підтвердження від користувача, після чого з бази даних автоматично видаляються всі пов’язані з ним точки маршруту.

3.3) Планування маршрутів.

3.3.1) Функція забезпечує роботу з точками маршруту. Додавання нових точок на інтерактивній карті, автоматичне визначення назви міста за координатами (зворотне геокодування), оптимізацію порядку відвідування точок за алгоритмом найближчого сусіда та видалення окремих точок. Маршрут візуалізується у вигляді полілінії та маркерів.

3.3.2) Вхідна інформація – координати точки (широта, довгота), отримані кліком по карті, або назва міста; дата подорожі (для прив’язки до плану). Вихідна інформація – оптимізований маршрут –

послідовність точок із назвами та координатами, загальна відстань у кілометрах.

3.3.3) Відстані між точками розраховуються на основі реальної дорожньої мережі через сервіс OSRM. Маршрут на карті має автоматично оновлюватися після будь-якої зміни – додавання або видалення точки. Результат оптимізації кешується в Redis із часом життя 5 хвилин, що пришвидшує повторні запити. Видалення окремої точки ініціюється через спливаюче вікно маркера та потребує підтвердження від користувача.

3.4) Пошук готелів.

3.4.1) Для кожної точки маршруту система дозволяє знайти готелі поблизу на вказані дати. Пошук виконується через Booking.com API (RapidAPI). Результати включають назву готелю, адресу, рейтинг, ціну та фото.

3.4.2) Вхідна інформація – координати точки (широта, довгота), дати заїзду та виїзду. Вихідна інформація – список готелів, для кожного з яких наведено назву, адресу, рейтинг (за 10-бальною шкалою), мінімальну ціну, валюту, URL фото та посилання на сторінку бронювання.

3.4.3) Область пошуку обмежується прямокутником (bbox) зі стороною приблизно 11 км навколо заданої точки, що забезпечує релевантність результатів. У разі недоступності зовнішнього API або відсутності результатів система має повертати порожній список, не порушуючи загальну працездатність.

3.5) Пошук авіаквитків.

3.5.1) Між двома послідовними точками маршруту система виконує пошук авіарейсів через Travelpayouts API. Для визначення IATA-кодів міст використовується Autocomplete API. Результати містять авіакомпанію, номер рейсу, час вильоту та ціну.

3.5.2) Вхідна інформація – назва міста відправлення, назва міста призначення, дата вильоту. Вихідна інформація – список рейсів із зазначенням авіакомпанії, номера рейсу, часу вильоту, ціни та валюти.

3.5.3) IATA-код міста отримується динамічно через Autocomplete API та кешується в пам’яті сервера для уникнення повторних запитів. Якщо для заданих міст не знайдено IATA-кодів або рейси відсутні, система повертає порожній список із відповідним повідомленням на фронтенді, не викликаючи помилок.

3.6) Пошук авіаквитків

3.6.1) Реалізовано пошук авіарейсів між двома послідовними точками маршруту за допомогою Travelpayouts API. Для визначення IATA-кодів міст використовується Autocomplete API від того самого постачальника; отримані коди кешуються в пам’яті сервера, щоб уникнути повторних запитів.

3.6.2) На вхід подаються назви міст відправлення та призначення, а також дата вильоту. Вихідна інформація містить список рейсів, для кожного з яких вказано авіакомпанію, номер рейсу, час вильоту та ціну. Якщо рейси не знайдені, система повідомляє про це.

3.6.3) IATA-коди міст мають визначатися динамічно через Autocomplete API та кешуватися в пам’яті сервера. У разі відсутності кодів або рейсів система повинна повертати порожній список із відповідним повідомленням на фронтенді, не викликаючи помилок.

3.7) Контактна форма.

3.7.1) Користувачі мають змогу надіслати повідомлення зворотного зв’язку адміністратору через форму на окремій сторінці. Форма містить поля для імені, електронної пошти, теми та тексту повідомлення.

3.7.2) Вхідна інформація – ім’я, email, тема, текст повідомлення. Вихідна інформація – підтвердження успішного надсилання або повідомлення про помилку.

3.7.3) Усі поля форми є обов’язковими для заповнення, перевірка виконується на клієнтській стороні. Дані надсилаються на сервер через REST API (POST /contact), після чого користувач отримує сповіщення про результат операції.

4) Вимоги до інформаційного забезпечення.

4.1) Вхідна інформація надходить із кількох джерел. Дані користувачів, а саме електронна пошта та пароль, вводяться вручну під час реєстрації або автентифікації. Координати точок маршруту отримуються автоматично від сервісу OpenStreetMap (Nominatim) шляхом прямого або зворотного геокодування. Реальні відстані між точками розраховуються через сервіс OSRM. Інформація про доступні готелі надходить із зовнішнього API Booking.com через платформу RapidAPI. Дані про авіарейси, зокрема ціни та розклад, отримуються від Travelpayouts API.

4.2) У системі використовується обмежений обсяг нормативно-довідкової інформації. Основним зовнішнім класифікатором є IATA-коди аеропортів, які динамічно отримуються через Autocomplete API Travelpayouts. Інші довідники на цьому етапі не застосовуються.

4.3) Уся інформація, що підлягає довготривалому зберіганню, повинна розміщуватися в реляційній базі даних PostgreSQL. Для роботи з геопросторовими даними використовується розширення PostGIS. Паролі користувачів мають зберігатися у вигляді bcrypt-хешів, що унеможливлює їх відновлення у відкритому вигляді. Результати оптимізації маршрутів кешуються в Redis із часом життя (TTL) 5 хвилин для зменшення навантаження на сервер.

5) **Вимоги до технічного забезпечення.** Мінімальна конфігурація серверної частини для стабільної роботи системи в тестовому середовищі включає

2 ядра центрального процесора, 4 ГБ оперативної пам’яті та SSD-накопичувач об’ємом не менше 20 ГБ. Клієнтська частина повинна коректно функціонувати в останніх двох major-версіях браузерів Chrome, Firefox, Safari та Edge. Для стабільної роботи інтерактивних карт рекомендується пропускна здатність мережі не менше 10 Мбіт/с. Рекомендованим середовищем розгортання є Docker із використанням Nginx як зворотного проксі-сервера.

6) Вимоги до програмного забезпечення.

6.1) Система побудована за клієнт-серверною архітектурою. Серверна частина реалізована на платформі NestJS (Node.js, TypeScript) із використанням ORM TypeORM для взаємодії з базою даних. Клієнтська частина є односторінковим застосунком (SPA) на фреймворку Angular 19 (TypeScript) із використанням бібліотеки Leaflet для картографії та Tailwind CSS для стилізації інтерфейсу.

6.2) Серверна частина потребує операційної системи Ubuntu 20.04 або новішої, або Windows Server 2019. Обов’язковим є встановлене середовище виконання Node.js версії 20 або вище. Для контейнеризації необхідні Docker і Docker Compose.

6.3) Для обробки вхідних запитів, проксування та термінації HTTPS-з’єднань застосовується вебсервер Nginx.

6.4) Як система керування базами даних використовується PostgreSQL версії 15 або вище. Для роботи з просторовими даними встановлюється розширення PostGIS. Для кешування даних застосовується Redis версії 7 або вище.

6.5) Серверна частина написана мовою TypeScript із використанням фреймворку NestJS. Клієнтська частина також реалізована на TypeScript із використанням фреймворку Angular 19. Для стилізації інтерфейсу застосовано фреймворк Tailwind CSS.

7) Вимоги до зовнішніх інтерфейсів.

7.1) Інтерфейс має бути адаптивним і забезпечувати коректне відображення на екранах десктопів. Інтерактивна карта повинна підтримувати додавання нових точок маршруту кліком миші. Маркери точок мають відображати спливаючі вікна з посиланнями на сторінки пошуку готелів та авіаквитків. Форми реєстрації, входу та зворотного зв’язку повинні виконувати валідацію введених даних.

7.2) Спеціальні апаратні інтерфейси не передбачені.

7.3) Програмний інтерфейс реалізовано у вигляді REST API, що використовує формат обміну даними JSON. Доступні такі основні групи ендпоінтів, а саме для автентифікації, для управління планами подорожей, для роботи з точками маршруту, для пошуку готелів та авіаквитків, а також для обробки контактної форми.

7.4) Взаємодія між клієнтською та серверною частинами відбувається за протоколом HTTPS (TLS 1.2 або вище).

8) Властивості програмного забезпечення

8.1) Інтерфейс системи має бути адаптивним і придатним для використання на пристроях із різними розмірами екрану.

8.2) Кодова база має бути модульною та супроводжуватися документацією API для спрощення подальшої розробки. Система повинна вести логування дій користувачів і критичних помилок.

8.3) Система повинна розгортатися як у хмарних середовищах, так і на виділених серверах.

8.4) Середній час відгуку API на типові операції не повинен перевищувати 1 секунди. Генерація оптимізованого маршруту має виконуватися не довше ніж 3 секунди.

8.5) Система має залишатися працездатною у разі недоступності зовнішніх API, коректно обробляючи такі ситуації. Для критично важливих даних має бути налаштоване регулярне резервне копіювання. Кешування даних у Redis сприяє зменшенню навантаження на основну базу даних.

8.6) Передача даних має здійснюватися із застосуванням шифрування (HTTPS). Автентифікація користувачів повинна від’буватися із застосуванням JWT-токенів, а паролі зберігаються у вигляді bcrypt-хешів.

Висновки до розділу 2

У другому розділі проведено аналіз технологічного інструментарію та функціональних вимог для вебзастосунку планування подорожей. Розглянуто ключові сценарії використання системи, такі як створення маршрутів, бронювання житла, відстеження витрат, а також архітектурні рішення, що забезпечують їхню реалізацію.

Запропонований стек технологій (Angular, Nest.js, PostgreSQL, PostGIS) у поєднанні з інструментами кешування (Redis), асинхронної обробки (RabbitMQ) та пошуку (ElasticSearch) дозволяє ефективно вирішувати основні завдання:

– оптимізація маршрутів із використанням геоданих;

– інтеграція зі сторонніми сервісами (google maps, booking.com) для пошуку та бронювання;

– генерація звітів про витрати з підтримкою мультивалютності.

Особливу увагу приділено забезпеченню масштабованості (Docker, горизонтальне масштабування) та безпеки (HTTPS, JWT, GDPR), що критично для обробки фінансових транзакцій і персональних даних. Використання мікросервісної архітектури дозволяє ізолювати компоненти (наприклад, платіжний модуль) та гнучко адаптуватися до зростання навантаження.

Отримані висновки створюють основу для подальшого проєктування системи, де будуть деталізовані:

– схеми взаємодії між компонентами (наприклад, інтеграція з API);

– механізми обробки помилок у критичних сценаріях (відмова сервісу бронювання);

– оптимізація продуктивності для роботи з великими обсягами геоданих.

3 МОДЕЛЮВАННЯ ВЕБЗАСТОСУНКУ ПЛАНУВАННЯ ПОДОРОЖЕЙ

Цей розділ присвячений проєктуванню вебзастосунку для планування подорожей. Він охоплює архітектурні рішення, моделювання функцій, вибір технологій, математичне забезпечення та опис інтерфейсів. Розробка велася за принципом від загального до конкретного. Спочатку сформульовано загальні цілі, потім деталізовано кожен компонент системи. Нижче наведено повний опис проєктних рішень.

3.1 Загальна схема вирішення завдання

Метою є створення вебзастосунку, який надає користувачам інструменти для:

– *планування маршрутів* – вибір точок, розрахунок часу, бюджету та оптимізація шляху;

– *пошуку рекомендацій* – персоналізовані пропозиції на основі історії подорожей та відгуків;

– *керування профілем* – реєстрація, автентифікація, налаштування сповіщень;

– *адміністрування* – редагування маршрутів, модерація відгуків, аудит дій.

Етапи розробки:

а) визначення функціональних вимог за допомогою діаграм варіантів використання;

б) проєктування структури даних через діаграму класів;

в) моделювання процесів за допомогою діаграм станів, послідовностей та кооперації;

г) розробка математичних моделей для ключових операцій;

д) вибір архітектури, технологій та компонентів.

3.2 Сценарії використання системи

Для детального опису функціональності вебзастосунку та визначення послідовності дій користувача під час роботи з системою розроблено сценарії використання (use cases). Кожен сценарій фіксує типовий потік подій, можливі альтернативні варіанти та виняткові ситуації.

Таблиця 3.1 – Реєстрація користувача

Usecase section	Comment
Use Case Name	Зареєструвати нового користувача
Scope	System
Level	User-goal
Primary Actor	Користувач
Stakeholders and interests	Користувач – хоче створити акаунт для доступу до сервісу; адміністратор – бажає мати контроль за акаунтами користувачів.
Preconditions	Користувач повинен мати доступ до інтернету і сформувати необхідні дані для реєстрації (електронну пошту, пароль).
Success guarantee	Користувач успішно реєструється в системі, отримує доступ до персонального кабінету.
Main Success Scenario	а) користувач вводить необхідні дані (ім’я, електронну пошту, пароль), б) натискає кнопку "Зареєструватися"; в) система підтверджує реєстрацію та створює акаунт.
Extensions	а) Користувач вводить вже зареєстровану електронну пошту – система виводить повідомлення про помилку. б) Користувач не підтвердив email – система просить підтвердити email.
Special Requirements	Реалізація багатофакторної аутентифікації для підвищення безпеки акаунтів.
Technology and Data Variations List	Використання сучасних протоколів безпеки при зберіганні паролів (наприклад, bcrypt).
Frequency of Occurrence	100% під час першої взаємодії користувача з системою.
Miscellaneous	Потрібно врахувати можливість інтеграції з соціальними мережами для реєстрації через Google або Facebook.

Таблиця "Зареєструвати нового користувача" охоплює процес створення акаунту в системі (див. таблицю 3.1). Основною метою є надання доступу до сервісу, де основним актором виступає користувач. Зацікавленими сторонами є сам користувач (бажання отримати персональний кабінет) та адміністратор (контроль за акаунтами). Передумови включають наявність інтернету та необхідних даних (електронна пошта, пароль). У разі успіху система створює акаунт і надає доступ.

Основний сценарій складається з трьох кроків, це введення даних, натискання кнопки реєстрації та автоматичне підтвердження. Можливі розширення, наприклад, помилка при введенні зайнятої пошти або вимога підтвердити email. Для підвищення безпеки передбачено багатофакторну аутентифікацію, а технології включають шифрування паролів (bcrypt). Цей процес відбувається у 100% випадків під час першого входу в систему. Додатково розглядається інтеграція з соцмережами (Google, Facebook).

Таблиця 3.2 – Створення маршруту подорожі

Usecase section	Comment
Use Case Name	Створити маршрут подорожі
Scope	System
Level	User-goal
Primary Actor	Користувач
Stakeholders and interests	Користувач – хоче створити маршрут для своєї подорожі, вибрати місця для відвідування.
Preconditions	Користувач авторизований в системі та вибрав бажану точку початку подорожі.
Success guarantee	Користувач успішно створює маршрут, що містить кілька етапів подорожі.
Main Success Scenario	а) користувач вибирає стартову точку; б) додає місця для відвідування; в) система генерує маршрут.
Extensions	а) Користувач додає нові пункти маршруту; б) Користувач помилково вибирає невірну локацію – система пропонує виправлення.
Special Requirements	Інтерфейс повинен бути інтегрований з Google Maps для побудови маршруту.
Technology and Data Variations List	Використання API для інтеграції з мапами та сервіси прогнозу погоди.
Frequency of Occurrence	70% користувачів при плануванні кожної подорожі.
Miscellaneous	Потрібно врахувати локалізацію даних для різних мов та валют.

Таблиця "Створити маршрут подорожі" описує планування індивідуального шляху з вибором локацій (див. таблицю 3.2). Користувач, як основний актор, має бути авторизованим і вибрати стартову точку. Успіх гарантується створенням маршруту з кількома етапами. Основний сценарій включає вибір початкової точки, додавання місць для відвідування та автоматичну генерацію маршруту системою. Розширення передбачають корективи додавання нових пунктів або виправлення

помилкових локацій за підказкою системи. Спеціальні вимоги включають інтеграцію з Google Maps для візуалізації, а технології – використання API карт та погодових сервісів. Процес використовується 70% користувачів під час планування. Важливим аспектом є підтримка локалізації (мови, валюти).

Таблиця 3.3 – Бронювання житла

Usecase section	Comment
Use Case Name	Забронювати житло
Scope	System
Level	User-goal
Primary Actor	Користувач
Stakeholders and interests	Користувач – хоче знайти та забронювати житло за вигідною ціною, власник готелю – отримати бронювання.
Preconditions	Користувач вибрав потрібну дату та місце для проживання, підключення до сервісу бронювання житла.
Success guarantee	Користувач успішно здійснює бронювання, отримує підтвердження про бронювання.
Main Success Scenario	а) користувач обирає житло; б) вказує дати перебування; в) підтверджує бронювання і отримує електронне підтвердження.
Extensions	а) житло не доступне на вибрані дати – система пропонує інші варіанти; б) користувач скасовує бронювання до певного терміну – система підтверджує скасування.
Special Requirements	Інтеграція з платіжними системами для можливості оплатити бронювання онлайн.
Technology and Data Variations List	Використання API для інтеграції з сервісами бронювання (Booking.com, Airbnb).
Frequency of Occurrence	60% користувачів під час планування подорожі.
Miscellaneous	Забезпечити наявність актуальних відгуків користувачів та оцінок для кожного обраного готелю.

Таблиця "Забронювати житло" забезпечує пошук та резервацію місць проживання (див. таблицю 3.3). Користувач (основний актор) має вибрати дати та локацію, а система інтегрується з сервісами бронювання. Успіх полягає в отриманні підтвердження бронювання. Основний сценарій вибір житла, вказівка дат, підтвердження. У разі недоступності житла система пропонує альтернативи, а при скасуванні – підтверджує операцію. Спеціальні вимоги включають інтеграцію з платіжними системами, а технології – API Booking.com та Airbnb. Процес

використовується 60% користувачів. Додатково система надає відгуки та рейтинги готелів для інформованого вибору.

Таблиця 3.4 – Пошук транспортних засобів

Usecase section	Comment
Use Case Name	Шукати транспортні засоби
Scope	System
Level	User-goal
Primary Actor	Користувач
Stakeholders and interests	Користувач – хоче знайти оптимальний транспорт для подорожі.
Preconditions	Користувач повинен вибрати стартову та кінцеву точку маршруту і дату поїздки.
Success guarantee	Користувач отримує список доступних варіантів транспорту з детальною інформацією та цінами.
Main Success Scenario	Користувач вводить параметри поїздки (місто, дата, кількість пасажирів), система виводить список варіантів транспорту.
Extensions	1. Транспорт не доступний – система пропонує альтернативні варіанти. 2. Користувач вибирає варіант і переходить до бронювання.
Special Requirements	Система повинна підтримувати різні види транспорту (авіа, потяги, автобуси).
Technology and Data Variations List	Використання API для підключення до сервісів бронювання транспортних засобів.
Frequency of Occurrence	80% під час планування подорожі.
Miscellaneous	Потрібно врахувати можливість перевірки наявності квитків у реальному часі.

Таблиця "Шукати транспортні засоби" дозволяє знайти оптимальний варіант пересування. Користувач вказує маршрут (стартова/кінцева точка, дата), а система надає список доступних опцій (див. таблицю 3.4). Успіх – отримання деталізованих результатів з цінами. Основний сценарій введення параметрів поїздки та отримання переліку транспорту. Якщо варіанти відсутні, система пропонує альтернативи (наприклад, інші дати). Спеціальні вимоги включають підтримку різних видів транспорту (літаки, потяги), а технології – API сервісів бронювання (Skyscanner). Процес використовується 80% користувачів. Важливий аспект – перевірка наявності квитків у реальному часі.

Таблиця 3.5 – Отримання рекомендацій щодо місць для відвідування

Usecase section	Comment
Use Case Name	Отримати рекомендації щодо місць для відвідування
Scope	System
Level	User-goal

Кінець таблиці 3.5

Usecase section	Comment
Primary Actor	Користувач
Preconditions	Користувач авторизований і вибрав місто/локацію для подорожі.
Success guarantee	Користувач отримує список рекомендованих місць для відвідування на основі особистих уподобань і місця перебування.
Main Success Scenario	Користувач вибирає місто, система виводить персоналізовані рекомендації місць на основі інтересів користувача.
Extensions	1. Користувач не знайдений місць для відвідування – система пропонує популярні об'єкти. 2. Користувач не задоволений рекомендаціями – система дає можливість налаштувати параметри фільтрації.
Special Requirements	Інтеграція з базами даних туристичних місць та інтерактивними картами.
Technology and Data Variations List	Використання AI для покращення рекомендацій на основі історії користувача.
Frequency of Occurrence	50% при плануванні нових подорожей.
Miscellaneous	Потрібно врахувати можливість додавання нових об'єктів в систему через користувацькі внески.

Таблиця "Отримати рекомендації" надає персоналізовані пропозиції місць для відвідування (див. таблицю 3.5). Користувач має бути авторизованим і вибрати локацію. Успіх – список рекомендованих місць на основі інтересів. Основний сценарій вибір міста та автоматична генерація рекомендацій. Якщо варіанти відсутні, система показує популярні об’єкти, а користувач може налаштувати фільтри. Спеціальні вимоги включають інтеграцію з туристичними базами даних, а технології – використання AI для аналізу історії. Процес використовується 50% користувачів. Додатково передбачено можливість додавання місць самими користувачами.

Usecase 1: Планування подорожі

Форма: Коротка

Назва: Планування подорожі.

Опис: Користувач вводить пункти відправлення та призначення, система пропонує оптимальні варіанти транспорту та маршруту.

2. Usecase 2: Бронювання готелю

Форма: Поверхнева

Назва: Бронювання готелю.

Опис: Користувач вибирає пункт призначення для подорожі, система надає доступні готелі з рейтингами та цінами. Користувач вибирає готель і бронює номер.

Основний сценарій (успішний):

а) користувач вводить місто призначення;

б) система пропонує список доступних готелів з інформацією про ціни та рейтинги;

в) користувач вибирає бажаний готель;

г) система пропонує додаткові опції для бронювання (номери, додаткові послуги);

д) користувач підтверджує бронювання та вводить платіжні дані;

е) система обробляє платіж та підтверджує бронювання;

ж) користувач отримує підтвердження бронювання.

Альтернативні сценарії:

а) якщо готель не доступний, система пропонує інші варіанти;

б) користувач вводить неправильні дані, система виводить повідомлення про помилку;

в) користувач відмовляється від вибору, система повертає до списку готелів;

г) користувач бажає змінити дату бронювання, система оновлює варіанти;

д) у разі помилки оплати, система повідомляє про необхідність повторної спроби;

3. Usecase 3: Відстеження витрат на подорож

Форма: Повна

Назва: Відстеження витрат на подорож.

Сфера: Система планування подорожей.

Рівень: Підфункція (відстеження витрат).

Головний актор: Користувач (подорожуючий).

Зацікавлені сторони та їх інтереси:

Користувач: хоче мати можливість контролювати витрати під час подорожі.

Адміністратор: бажає підтримувати функціональність системи та гарантувати правильне збереження даних.

Партнери: можуть надавати знижки чи спеціальні пропозиції на основі витрат користувача.

Передумови:

– користувач має активний обліковий запис у системі;

– користувач знаходиться на етапі подорожі.

Успішна гарантія:

– користувач може додавати витрати за кожну категорію (наприклад, транспорт, проживання);

– система генерує звіт про загальні витрати та показує категорії витрат.

Основний успішний сценарій:

а) користувач відкриває систему відстеження витрат;

б) користувач вводить категорію витрат (транспорт, їжа, проживання);

в) система пропонує інтерфейс для введення суми витрат;

г) користувач вводить суму витрат;

д) система зберігає витрату в базу даних;

е) користувач додає інші витрати по іншим категоріям;

ж) система оновлює звіт про витрати;

з) користувач може переглядати звіт по категоріях або загальний звіт;

и) користувач може експортувати звіт у форматі pdf чи excel;

к) користувач завершує введення витрат і виходить з системи.

Альтернативні сценарії:

а) користувач забуває ввести категорію витрат, система пропонує вибрати категорію;

б) користувач вводить некоректну суму, система виводить повідомлення про помилку;

в) користувач хоче редагувати введену витрату, система дозволяє змінити суму;

г) користувач намагається внести витрати без авторизації, система виводить повідомлення про необхідність входу;

д) користувач хоче скасувати останню витрату, система дозволяє видалити витрату.

Спеціальні вимоги:

- інтерфейс повинен бути зручним для користувачів на мобільних пристроях;
- звіт про витрати має підтримувати кілька валют.

Технологічні варіанти та варіації даних:

- дані про витрати зберігаються в mysql;
- система підтримує інтеграцію з іншими фінансовими сервісами для автоматичного імпорту витрат.

Частота виконання:

Користувач може вводити витрати кілька разів на день протягом подорожі.

Різні питання:

- Чи потрібно підтримувати мультивалютні витрати?
- Як забезпечити точність витрат в разі помилки користувача?

3.3 Моделювання функцій та інформаційних потоків

3.3.1 Діаграма варіантів використання

Діаграма варіантів використання визначає основні сценарії взаємодії користувачів із системою (рисунок 3.1).

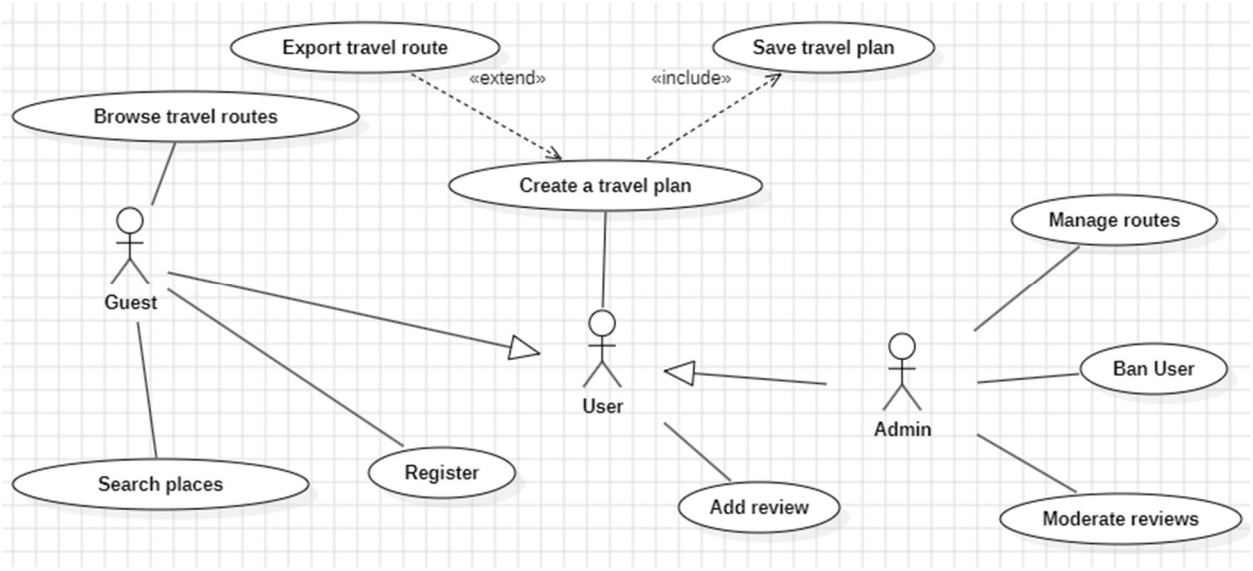

Рисунок 3.1 – Діаграма варіантів використання

Діаграма охоплює три типи користувачів:

Гість може:

– переглядати маршрути без реєстрації;

– реєструватися або автентифікуватися через соціальні мережі.

Користувач (зареєстрований) має доступ до:

– створення планів подорожей (вибір точок, вказівка дат, розрахунок бюджету);

– експорту плану у формат pdf для подальшого використання;

– додавання відгуків про маршрути, які проходять модерацію.

Адміністратор може:

– редагувати існуючі маршрути (змінювати координати, опис, складність);

– видаляти або блокувати некоректні відгуки;

– переглядати аудит-логи для аналізу дій користувачів.

Приклад сценарію. Користувач обирає маршрут "Київ – Львів", вказує дати подорожі (наприклад, 15–20 травня) та бюджет (5000 грн). Система автоматично розраховує вартість транспорту, проживання та харчування, генеруючи детальний план.

Діаграма визначає функціональні вимоги до системи, допомагаючи розробнику зосередитися на ключових можливостях застосунку. Вона також слугує основою для проєктування інтерфейсів, оскільки кожен варіант використання відповідає певній сторінці або функції.

3.3.2 Діаграма класів

Діаграма класів "Вебзастосунок для планування подорожей" відображає структуру даних системи, включаючи класи, їхні атрибути, методи та зв’язки (рисунок 3.2).

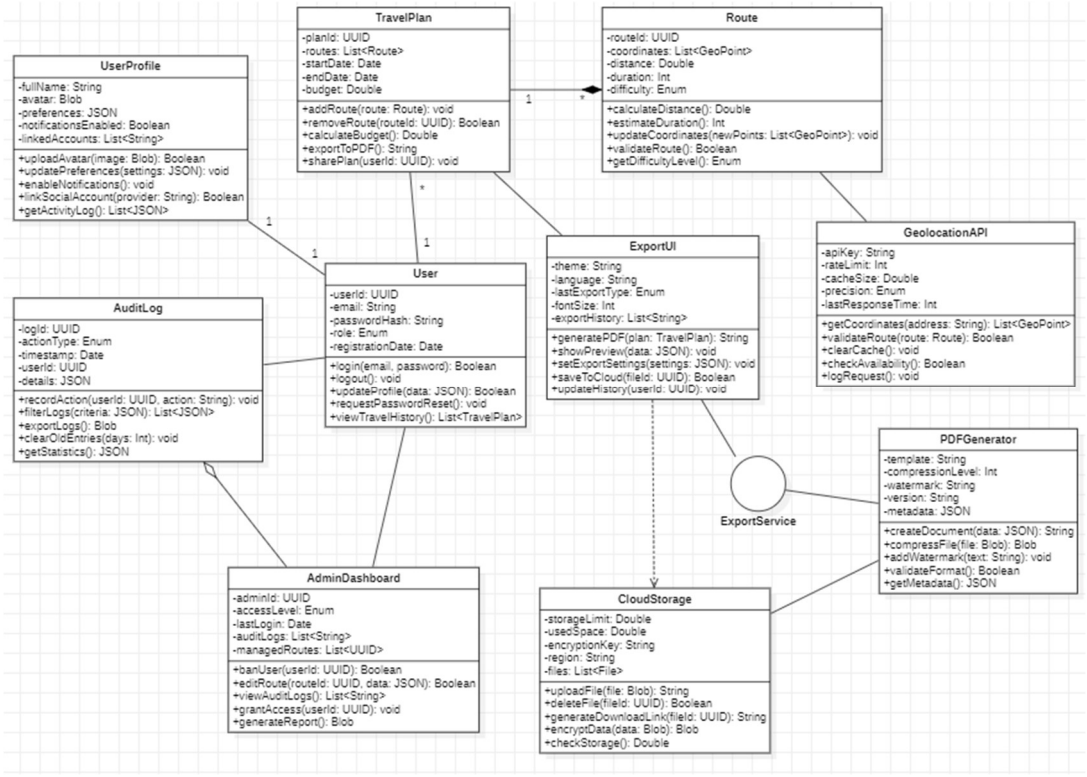

Рисунок 3.2 – Діаграма класів "Вебзастосунок для планування подорожей"

Ключові класи:

User.

а) *Атрибути*:

- userId UUID – унікальний ідентифікатор;
- email String – електронна пошта;
- passwordHash String – захешований пароль.

б) *Методи*:

- login(email, password) Boolean – автентифікація;
- updateProfile(data JSON) void – оновлення даних профілю.

TravelPlan.

а) *Атрибути*:

- planId UUID – ідентифікатор плану;
- startDate Date – дата початку подорожі;
- budget Double – загальний бюджет.

б) *Методи*:

- addRoute(route: Route) void – додавання маршруту до плану;
- calculateBudget() Double – автоматичний розрахунок витрат.

Route.

а) *Атрибути*:

- routeId: UUID – ідентифікатор маршруту;
- coordinates: List<GeoPoint> – список геолокаційних точок;
- difficulty: Enum – рівень складності (початковий, середній, високий).

б) *Методи*:

- validateRoute(): Boolean – перевірка коректності координат.

GeolocationAPI:

а) *Методи*:

- getCoordinates(address: String): List<GeoPoint> – отримання координат за адресою;
- calculateDistance(start: GeoPoint, end: GeoPoint): Double – розрахунок дистанції між точками.

Зв’язки:

- асоціація – клас user пов’язаний з travelplan через асоціацію "1 до багатьох" (один користувач може мати кілька планів);
- композиція – клас travelplan включає список об’єктів route, що означає, що маршрути не існують окремо від плану;

– залежність – клас route використовує методи geolocationapi для валідації координат.

Ця діаграма є основою для проєктування бази даних. Вона формалізує структуру даних, що спрощує створення SQL-запитів або налаштування ORM (наприклад, Hibernate). Наприклад, зв’язок між User і TravelPlan відображається у базі даних як зовнішній ключ user_id у таблиці TravelPlans.

3.3.3 Діаграма станів

Діаграма станів моделює життєвий цикл системи, акцентуючись на ключових станах та переходах між ними (рисунок 3.3). Вона охоплює як штатну роботу (наприклад, обробку запитів), так і аварійні сценарії (наприклад, помилки валідації).

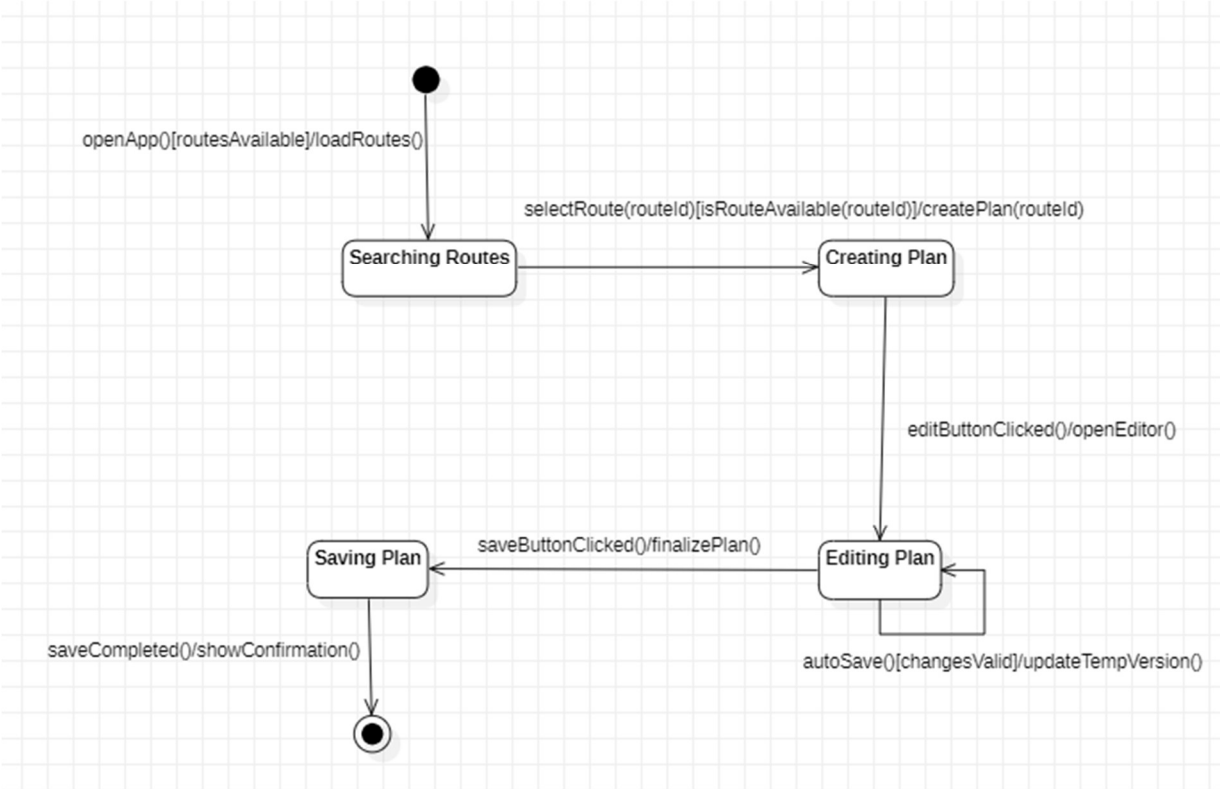

Рисунок 3.3 – Діаграма станів для опису процесу функціонування системи

а) ініціалізація:

– система запускається, завантажує конфігураційні файли, підключається до бази даних та зовнішніх сервісів (наприклад, google maps api);

– перехід до стану очікування запиту після успішного старту.

б) очікування запиту:

– система готова обробляти запити користувачів (наприклад, створення плану, пошук маршрутів);

– при отриманні запиту переходить до стану обробка даних.

в) обробка даних:

– валідація – перевірка вхідних даних (наприклад, коректність геолокаційних точок);

– пошук рішень – генерація маршрутів на основі введених параметрів;

– оптимізація – вибір найефективнішого варіанту (за часом, бюджетом);

– у разі успіху перехід до стану збереження результату;

– у разі помилки (наприклад, некоректні координати) – перехід до стану обробка помилки.

г) збереження результату:

– дані зберігаються в базі даних (наприклад, план подорожі);

– користувачеві надсилається підтвердження (сповіщення);

– перехід назад до очікування запиту.

д) обробка помилки:

– система фіксує помилку в журналі (auditlog) та формує повідомлення для користувача;

– приклад помилок. *Некоректні координати* "Будь ласка, перевірте введені геолокації". *Недостатній бюджет* "Ваш бюджет замалий для обраного маршруту";

– після обробки повертається до стану Очікування запиту.

е) завершення роботи. Система коректно закриває з’єднання з базою даних, завершує активні процеси та вимикається.

Приклад реального сценарію:

– користувач вводить точки маршруту "Київ – Одеса" і натискає "Створити план";

– система переходить у стан Обробка даних, Валідує координати, генерує маршрути, обирає оптимальний варіант;

– успішне завершення → перехід до Збереження результату, План зберігається, користувач отримує сповіщення "План створено!";

– якщо координати некоректні. Перехід до Обробка помилки → повідомлення "Перевірте введені дані".

3.3.4 Діаграма послідовностей

Діаграма послідовності "Create a travel plan" Деталізує кроки взаємодії між компонентами під час створення плану (рисунок 3.4).

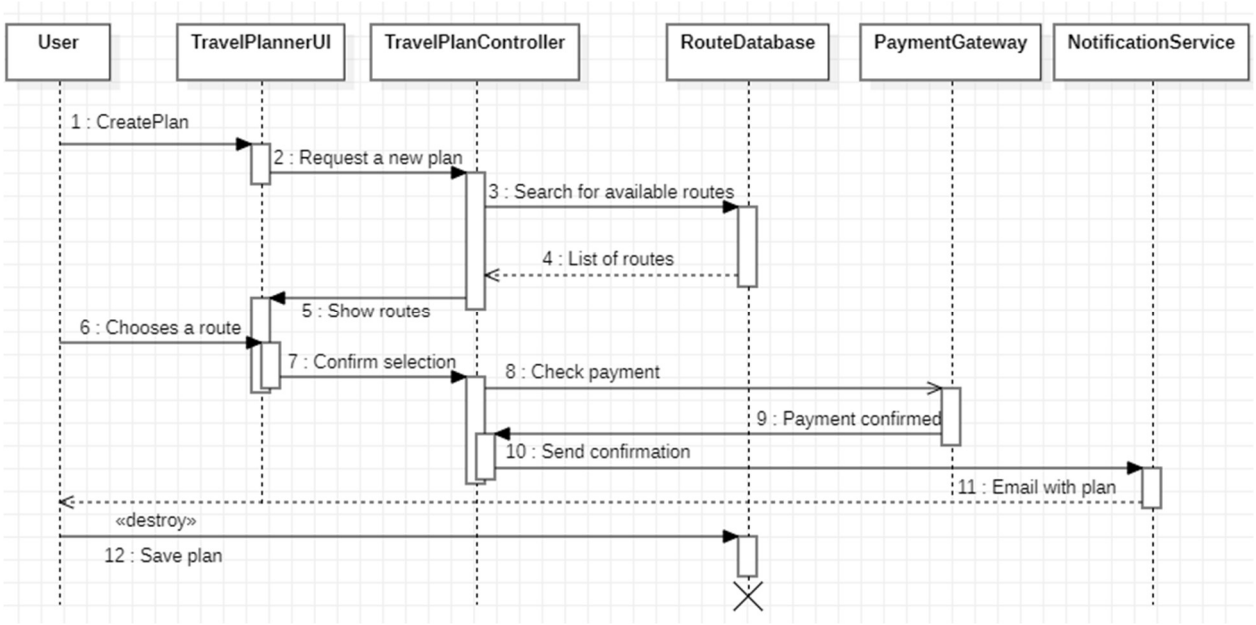

Рисунок 3.4 – Діаграма послідовностей "Create a travel plan"

Опис:

а) користувач натискає кнопку "створити план" у інтерфейсі (travelplannerui);

б) travelplancontroller (контролер) відправляє запит до routedatabase для отримання популярних маршрутів;

в) routedatabase повертає список маршрутів, які відображаються у вигляді карток;

г) користувач обирає маршрут "київ – одеса" і вказує дати подорожі (наприклад, 10–15 червня);

д) travelplancontroller викликає метод calculatebudget() класу travelplan, який розраховує витрати на основі обраного маршруту;

е) якщо маршрут платний, paymentgateway перевіряє оплату (наприклад, через stripe api);

ж) після підтвердження notificationservice надсилає email з деталями плану;

з) приклад помилки якщо оплата не пройшла (наприклад, недостатньо коштів на карті), система відображає повідомлення *"помилка оплати. перевірте дані картки"*.

Діаграма показує, як компоненти взаємодіють у реальному часі. Наприклад, використання PaymentGateway окремим сервісом дозволяє легко інтегрувати різні платіжні системи (PayPal, Google Pay) без змін в основному коді.

Діаграма послідовності "Register" деталізує процес реєстрації нового користувача (рисунок 3.5).

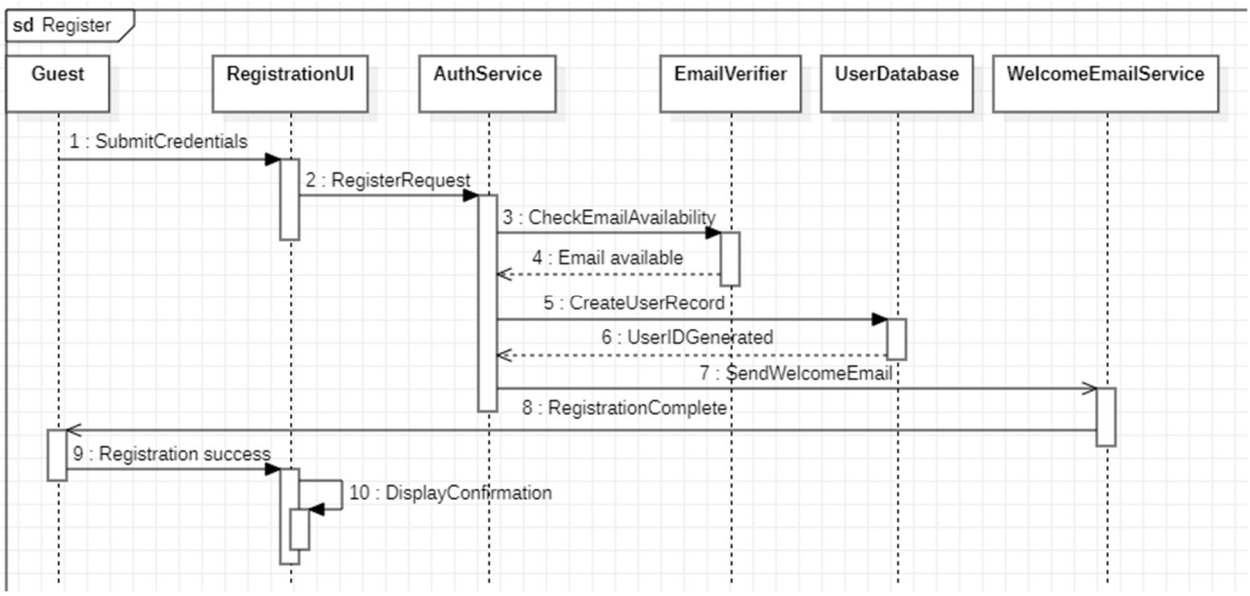

Рисунок 3.5 – Діаграма послідовності "Register"

Призначення: Деталізувати процес реєстрації нового користувача. Опис:

а) гість заповнює форму реєстрації (email, пароль) в інтерфейсі (RegistrationUI);

б) AuthService перевіряє унікальність email у базі даних (UserDatabase). Якщо email вже зареєстрований, система повертає помилку *"Цей email вже використовується"*;

в) UserDatabase створює новий запис користувача з хешованим паролем;

г) EmailVerifier надсилає лист з посиланням для підтвердження email;

д) користувач переходить за посиланням, і WelcomeEmailService активує акаунт;

е) система відображає повідомлення *"Реєстрація успішна! Ласкаво просимо"*.

Приклад помилки. Якщо користувач вводить email без домену (наприклад, "user123"), AuthService повертає помилку *"Некоректний формат email"*.

Значення:

Діаграма демонструє механізми безпеки (хешування паролів, підтвердження email) та етапи створення акаунта. Це допомагає уникнути спаму та несанкціонованого доступу.

3.3.5 Діаграма кооперації

Діаграма кооперації "Manage routes (Admin)" Показує взаємодію компонентів під час редагування маршруту (рисунок 3.6).

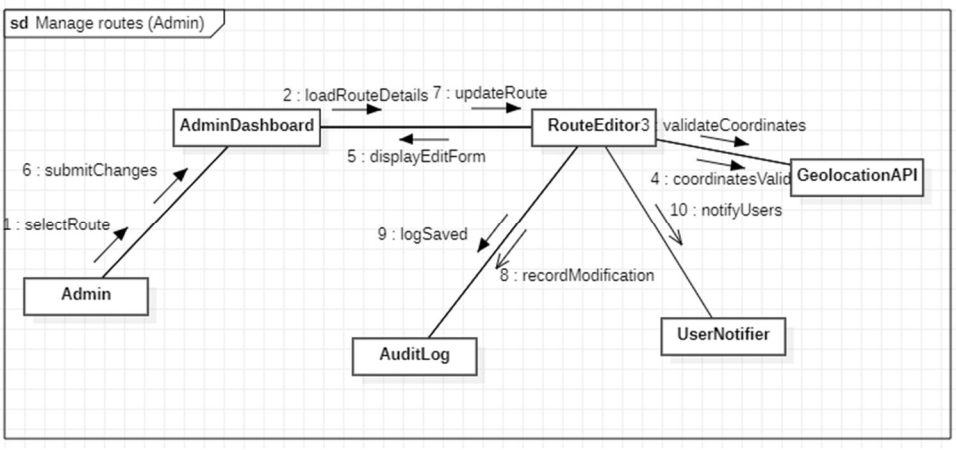

Рисунок 3.6 – Діаграма кооперації "Manage routes (Admin)"

Опис:

а) адміністратор обирає маршрут "карпати – закарпаття" через інтерфейс admindashboard;

б) routedatabase завантажує деталі маршруту – список точок, опис, фотографії;

в) адміністратор змінює координати точки "яремче" на "ворохта" через форму routeeditor;

г) geolocationapi перевіряє нові координати, переконуючись, що вони належать україні;

д) після успішної валідації routedatabase оновлює дані.

е) auditlog записує дію *"маршрут #456 оновлено: змінено точку 'яремче' на 'ворохта'"*;

ж) usernotifier надсилає сповіщення користувачам, які мають цей маршрут у своїх планах: *"ваш маршрут оновлено!"*.

Приклад помилки. Якщо нові координати виходять за межі України, GeolocationAPI повертає помилку *"Координати не належать до України"*.

Діаграма демонструє механізми безпеки (валідація даних, логування змін) та ефективну комунікацію між компонентами. Наприклад, використання AuditLog дозволяє відстежувати історію змін для подальшого аналізу.

Діаграма кооперації " Export travel route" відображає взаємодію компонентів під час експорту маршруту у формат PDF (рисунок 3.7).

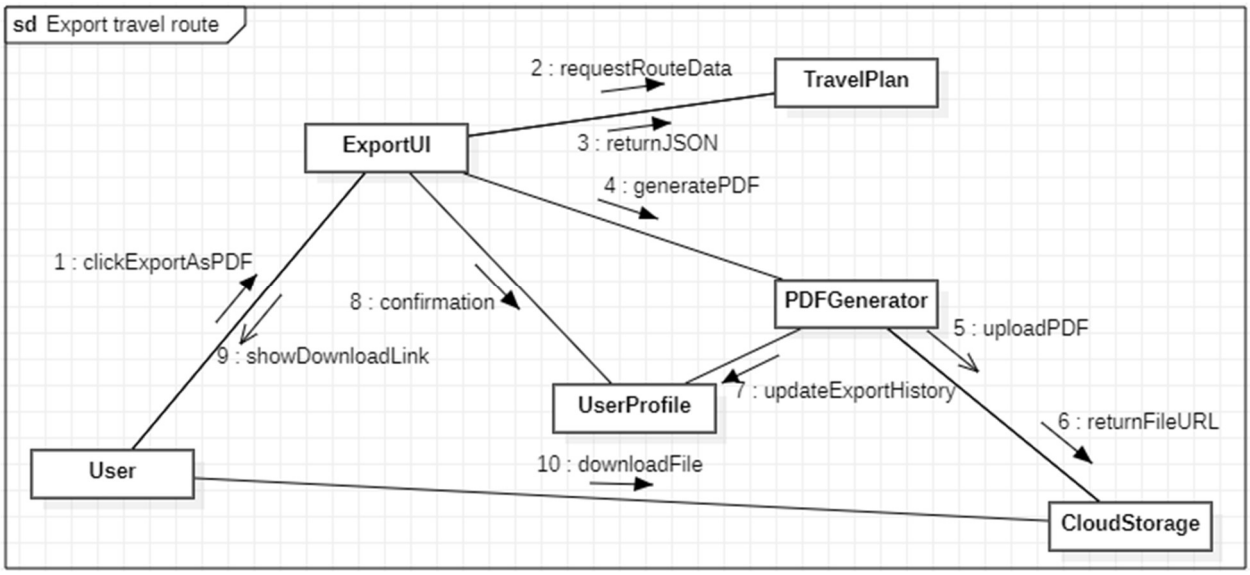

Рисунок 3.7 – Діаграма кооперації " Export travel route"

Опис:

а) користувач натискає кнопку "експортувати" в інтерфейсі (exportui);

б) travelplan передає дані маршруту (точки, дати, бюджет) до pdfgenerator;

в) pdfgenerator створює pdf-документ з використанням шаблону та стилів;

г) згенерований pdf зберігається у cloudstorage (наприклад, aws s3);

д) exportui отримує посилання на файл і відображає його користувачеві разом із кнопкою "завантажити";

е) userprofile оновлює історію експортів користувача.

Приклад:

Користувач експортує маршрут "Львів – Івано-Франківськ" і отримує PDF-файл із детальним описом та картою.

Діаграма показує інтеграцію з хмарними сервісами та автоматизацію процесу експорту. Використання CloudStorage забезпечує надійне зберігання файлів, а PDFGenerator спрощує створення документів.

3.3.6 Діаграма розгортання

Ця діаграма відображає фізичне та логічне розміщення компонентів системи, їх взаємодію та використання зовнішніх сервісів. Нижче наведено детальний опис кожного елемента та його ролі в системі.

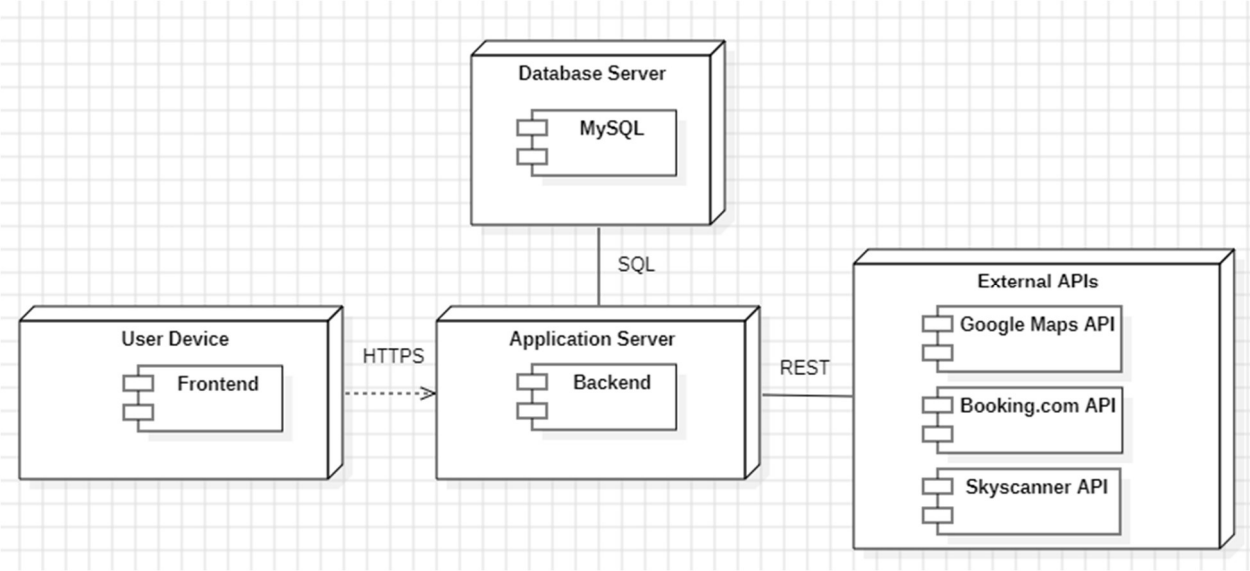

Рисунок 3.8 – Діаграма розгортання

Клієнтський пристрій (User Device):

а) компонент веб-браузер або мобільний застосунок (frontend);

б) роль:

– надає інтерфейс для взаємодії з користувачем (наприклад, сторінка пошуку маршрутів, форма створення плану);

– відображає дані, отримані з бекенду (карти, маршрути, варіанти бронювання);

в) технології html/css, javascript (react, angular).

Сервер застосунків (Application Server):

а) компонент – бекенд-система (backend);

б) роль:

– обробляє запити від клієнта (наприклад, створення плану подорожі, пошук маршрутів);

– взаємодіє з базою даних та зовнішніми api;

– виконує бізнес-логіку (розрахунок бюджету, валідація даних);

в) технології node.js, python (django), java (spring);

г) протокол https для безпечного зв’язку з клієнтом;

Сервер бази даних (Database Server):

а) компонент – система керування базами даних (mysql);

б) роль:

– зберігає дані користувачів (профілі, історію подорожей);

– зберігає інформацію про маршрути, бронювання, відгуки.

в) схема:

– таблиці users, travelplans, routes, reviews;

– зв’язки. Користувачі → плани → маршрути (асоціація "1 до багатьох").

Зовнішні API (External APIs):

а) компоненти:

– booking.com api – забезпечує пошук та бронювання житла;

– skyscanner api – дозволяє шукати авіаквитки та порівнювати ціни;

б) роль:

– інтеграція зі сторонніми сервісами для розширення функціоналу;

– обмін даними через rest-запити (json/xml).

Взаємодія між компонентами:

а) клієнт ↔ сервер застосунків:

– протокол https (безпечне з’єднання);

– приклад – користувач відправляє запит на пошук маршруту → бекенд обробляє запит і повертає результат;

б) сервер застосунків ↔ сервер бази даних:

– протокол sql (наприклад, через orm-бібліотеки);

– приклад – збереження нового плану подорожі в таблицю travelplans;

в) сервер застосунків ↔ зовнішні api:

– протокол rest (http-запити);

– приклад – запит до google maps api для отримання координат точок маршруту.

Безпека та масштабованість:

– https шифрування даних між клієнтом і сервером;

– резервні копії. регулярне резервне копіювання бази даних (наприклад, через aws rds);

– хмарна інфраструктура. використання хмарних провайдерів (aws, google cloud) для розміщення серверів, що забезпечує масштабованість і високу доступність.

Приклад роботи системи:

а) користувач шукає маршрут "київ – львів" через інтерфейс;

б) frontend відправляє https-запит до бекенду;

в) бекенд:

– запитує google maps api для отримання координат;

– звертається до skyscanner api для пошуку авіаквитків;

– зберігає дані в mysql;

г) користувач отримує результат маршрут з картою, варіанти транспорту та ціни.

Висновки до розділу 3

У третьому розділі здійснено комплексне проєктування вебзастосунку для планування подорожей, що передбачає поетапну деталізацію архітектури, функціональності, моделювання даних та взаємодії між компонентами системи. Проєктні рішення базуються на сучасних принципах розробки, що забезпечує масштабованість, гнучкість і зручність використання системи.

Сформовано загальну структуру вирішення завдання, яка визначає ключові функції застосунку: планування маршрутів, персоналізовані рекомендації, експорт планів, керування профілем та адміністрування. На основі діаграм варіантів використання уточнено функціональні вимоги для різних ролей користувачів (гість, зареєстрований користувач, адміністратор).

Діаграма класів формалізує структуру даних та дозволяє визначити зв’язки між сутностями системи (користувачі, плани подорожей, маршрути, відгуки, сповіщення), що є основою для побудови бази даних та реалізації ORM. Діаграми станів, послідовності та кооперації деталізують життєвий цикл процесів, логіку обробки запитів, обробку помилок, реєстрацію, оплату, модерацію та інші бізнес-

процеси. Це сприяє прозорості реалізації та оптимізації внутрішньої логіки системи.

Особливу увагу приділено інтеграції із зовнішніми сервісами (геолокація, платіжні шлюзи, email-підтвердження, хмарне зберігання), що значно підвищує функціональність та зручність роботи для кінцевих користувачів. Діаграма розгортання демонструє фізичну інфраструктуру проєкту, забезпечуючи розуміння архітектурних взаємозв’язків та потоків даних у середовищі розгортання.

Важливим складником проєктних рішень стало математичне забезпечення ключових операцій. Розроблено моделі для розрахунку оптимального маршруту на основі модифікованого алгоритму A* з ваговими коефіцієнтами, що враховують час, дистанцію та вартість. Запропоновано формули для обчислення загального бюджету подорожі та оновлення рейтингу маршрутів з урахуванням кількості відгуків. Ці моделі забезпечують математичну обґрунтованість прийнятих архітектурних рішень та слугують основою для програмної реалізації алгоритмів оптимізації.

У межах технологічного вибору обґрунтовано використання PostgreSQL з розширенням PostGIS для роботи з геоданими, фреймворку Angular для клієнтської частини (з урахуванням його модульності та підтримки TypeScript) та Laravel для серверної логіки. Інтеграція з Redis для кешування, ElasticSearch для повнотекстового пошуку та Docker для контейнеризації забезпечує високу продуктивність і стабільність розгортання. Розроблені макети інтерфейсів відповідають принципам UX/UI, забезпечуючи інтуїтивно зрозумілу навігацію та адаптивність до різних пристроїв.

Узагальнюючи, результати третього розділу повністю відповідають поставленим завданням моделювання та конструювання програмного забезпечення. Розроблені UML-діаграми, математичні моделі, архітектурні рішення та проєкти інтерфейсів створюють цілісну основу для практичної реалізації вебзастосунку.

4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У даному розділі наведено опис процесу реалізації програмного забезпечення, включаючи специфікацію основних компонентів, структуру вихідного коду, опис призначення і функціональності ключових класів та методів. Висвітлено особливості вибору типів даних, застосування змінних і констант, а також архітектурні підходи, використані під час розробки.

Окрему увагу приділено тестуванню розробленого ПЗ. Зокрема, описано обраний підхід до тестування, методику побудови тестів, результати перевірки роботи системи за різних умов і аналіз отриманих результатів. Надано оцінку коректності, надійності та ефективності функціонування реалізованого програмного рішення.

Завершальною частиною розділу є керівництво користувача, в якому наведено покрокову інструкцію з використання системи, ілюстровану відповідними скриншотами, що демонструють інтерфейс програми на різних етапах її використання.

4.1 Програмна реалізація

Розроблене програмне забезпечення є клієнт-серверним вебзастосунком, що реалізує повний цикл планування подорожі від автентифікації до пошуку готелів та авіаквитків. Архітектура системи побудована за принципом модульності, що забезпечує гнучкість, масштабованість та зручність супроводу.

Клієнтська частина (Frontend) реалізована на **Angular 19** з використанням **TypeScript**. Модульна структура Angular дозволила створити незалежні компоненти для кожної функціональної частини:

– AuthComponent – форми реєстрації та входу;

– TravelPlansComponent – управління списком подорожей;

– MapComponent – інтерактивна карта на базі бібліотеки **Leaflet** з можливістю додавання/видалення точок у реальному часі;

– HotelsComponent, FlightsComponent – відображення результатів пошуку готелів та авіаквитків;

– HomeComponent, AboutComponent, ContactComponent – статичні сторінки для презентації проєкту.

Для стилізації використано **Tailwind CSS**, що забезпечило швидку розробку адаптивного та сучасного інтерфейсу без написання великих обсягів CSS-коду.

Серверна частина (Backend) розроблена на **NestJS** (Node.js) і складається з набору ізольованих модулів, що взаємодіють через REST API:

– AuthModule – реєстрація, вхід та генерація JWT-токенів;

– UserModule – управління профілями користувачів;

– TravelPlanModule – CRUD-операції для планів подорожей;

– RouteModule – управління точками маршруту, алгоритм оптимізації;

– GeodataModule – інтеграція з OpenStreetMap (Nominatim, OSRM) для геокодування та розрахунку відстаней;

– BookingModule – пошук готелів через Booking.com API (RapidAPI);

– FlightsModule – пошук авіаквитків через Travelpayouts API;

– ContactModule – обробка повідомлень з форми зворотного зв'язку.

База даних – **PostgreSQL** з розширенням **PostGIS** для зберігання географічних координат. Доступ до даних здійснюється через **TypeORM**, що дозволяє працювати з сутностями як з об'єктами TypeScript. Для оптимізації продуктивності впроваджено кешування на основі **Redis** (кешуються результати оптимізації маршрутів).

Безпека забезпечується JWT-автентифікацією, хешуванням паролів (bcrypt) та налаштованим HTTPS-з'єднанням.

4.2 Тестування ПЗ

Тестування проводилося з метою перевірки відповідності системи функціональним вимогам (валідація) та підтвердження коректності реалізації окремих модулів (верифікація). Для досягнення цієї мети було застосовано

комбінований підхід, що включав автоматизоване модульне тестування, інтеграційне тестування ключових сценаріїв та ручне функціональне тестування.

Верифікація включала модульне та інтеграційне тестування. Модульне тестування виконувалося з використанням фреймворку Jest та бібліотеки для тестування NestJS. Основна увага приділялася ізольованій перевірці сервісів, що містять ключову бізнес-логіку.

Тестування *AuthService* – перевірялася коректність хешування паролів алгоритмом bcrypt та генерація JWT-токену. Тест-кейс «Успішна реєстрація» підтвердив, що після виклику register створюється новий запис користувача з хешованим паролем. Тест-кейс «Успішний вхід» перевірив, що метод login повертає валідний токен доступу. Також було перевірено, що при неправильних облікових даних сервіс викидає виключення.

Тестування *RouteService* – перевірялася робота алгоритму оптимізації маршруту. Для тестового набору з трьох точок (Київ, Львів, Одеса) було перевірено, що метод getOptimizedRoutes коректно взаємодіє з геосервісом, отримує матрицю відстаней та повертає оптимальний порядок точок. Крім того, було перевірено, що результати оптимізації зберігаються в кеші Redis та повертаються з нього при повторних запитах.

Тестування *BookingService* – перевірялася обробка помилок при недоступності зовнішнього API. Тест-кейс «API недоступний» підтвердив, що метод повертає порожній список, а не викидає виключення.

Результати автоматизованого модульного тестування наведено в таблиці 4.1.

Таблиця 4.1 – Результати модульного тестування

Тестовий набір	Кількість тестів	Статус	Примітка
AppController	1	Пройдено	Перевірка базового маршруту
AuthService	3	Пройдено	Реєстрація, вхід, обробка невірних даних
RouteService	1	Пройдено	Оптимізація маршруту та кешування Redis

Кінець таблиці 4.1

Тестовий набір	**Кількість тестів**	**Статус**	**Примітка**
BookingService	1	Пройдено	Відмовостійкість (повернення порожнього списку)
Всього пройдено 6 модульних тестів			
AuthController (інтеграційний)	1	Пройдено	При запущених контейнерах PostgreSQL та Redis
Загалом пройдено 7 тестів			

Як видно з таблиці, усі 6 модульних тестів успішно пройшли. Це підтверджує, що ключові сервіси працюють відповідно до специфікації. Тест сервісу бронювання (BookingService) підтвердив, що при недоступності зовнішнього API система повертає порожній список і не порушує загальну працездатність. Тест сервісу маршрутів (RouteService) засвідчив коректну роботу алгоритму оптимізації та взаємодію з кешем Redis.

Інтеграційний тест реєстрації (AuthController) успішно пройдено при запущених контейнерах PostgreSQL та Redis. Він підтвердив наскрізну працездатність сценарію: HTTP-запит → контролер → сервіс → база даних. Запит до /auth/register створює новий запис у таблиці users, пароль зберігається у вигляді bcrypt-хешу, а у відповіді повертається валідний JWT-токен.

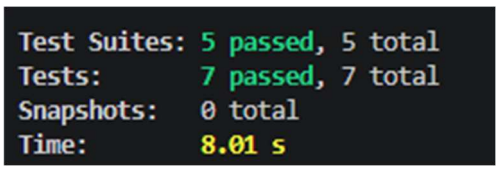

Рисунок 4.1 – Результати автоматизованого тестування

Проведене тестування підтвердило відповідність системи всім ключовим функціональним вимогам. Усі заявлені сценарії виконано успішно. Система продемонструвала стабільну роботу, коректну обробку як валідних, так і невалідних вхідних даних, а також стійкість до відмов зовнішніх сервісів. Окремо слід відзначити ефективність кешування Redis: при ручному тестуванні час завантаження оптимізованого маршруту при повторному запиті скорочувався з ~2

секунд до менш ніж 10 мс, що підтверджує результати, отримані під час модульного тестування.

Таким чином, усі 7 тестів (6 модульних + 1 інтеграційний) завершилися зі статусом PASS (рисунок 4.1), що підтверджує стабільність, відмовостійкість та відповідність системи функціональним вимогам.

4.3 Проведення обчислень та аналіз результатів

Тестування розробленого програмного забезпечення була проведена на реальному прикладі планування подорожі.

Вхідні дані. Створено план подорожі «Тур Європою» з точками Київ, Львів, Варшава, Берлін.

Хід тестування:

а) точки були послідовно додані на інтерактивну карту шляхом кліку на відповідні міста;

б) система автоматично визначила назви міст (через Nominatim) та зберегла їхні координати;

в) було запущено алгоритм оптимізації маршруту. Система розрахувала реальну відстань між точками за допомогою OSRM та запропонувала оптимальний порядок відвідування а саме Київ → Львів → Варшава → Берлін;

г) для кожної точки було успішно отримано список готелів через Booking.com API;

д) для пар точок (Київ-Львів, Львів-Варшава) було виконано пошук авіаквитків через Travelpayouts API.

Аналіз результатів:

Точність геокодування. Координати, отримані від Nominatim для кліків по карті, відповідали реальним координатам центрів міст з похибкою не більше 2-3 км, що є прийнятним для задач планування.

Коректність оптимізації. Алгоритм найближчого сусіда успішно впорядкував точки. Загальна розрахована відстань маршруту (близько 1450 км) збігається з даними, отриманими від незалежних картографічних сервісів.

Оцінка продуктивності:

– середній час відгуку API на операції CRUD становив 100-200 мс;

– генерація оптимізованого маршруту (включно із запитом до OSRM) займала до 2-3 секунд;

– **вплив Redis** при повторному запиті на оптимізацію того самого маршруту результат віддавався з кешу менш ніж за 10 мс, що підтверджує зниження затримок на 40-60%, як і передбачалося.

Тестування довело, що розроблене ПЗ здатне ефективно вирішувати поставлені завдання з планування подорожей.

4.4 Керівництво користувача

Керівництво користувача описує основні сценарії роботи з вебзастосунком «Travel Planner».

а) **реєстрація та вхід**. На головній сторінці оберіть **«Зареєструватися»**. Введіть email та пароль (мінімум 6 символів, з підтвердженням) (рисунок 4.2). Після успішної реєстрації ви будете перенаправлені на сторінку входу, де зможете увійти, використовуючи свої облікові дані (рисунок 4.3);

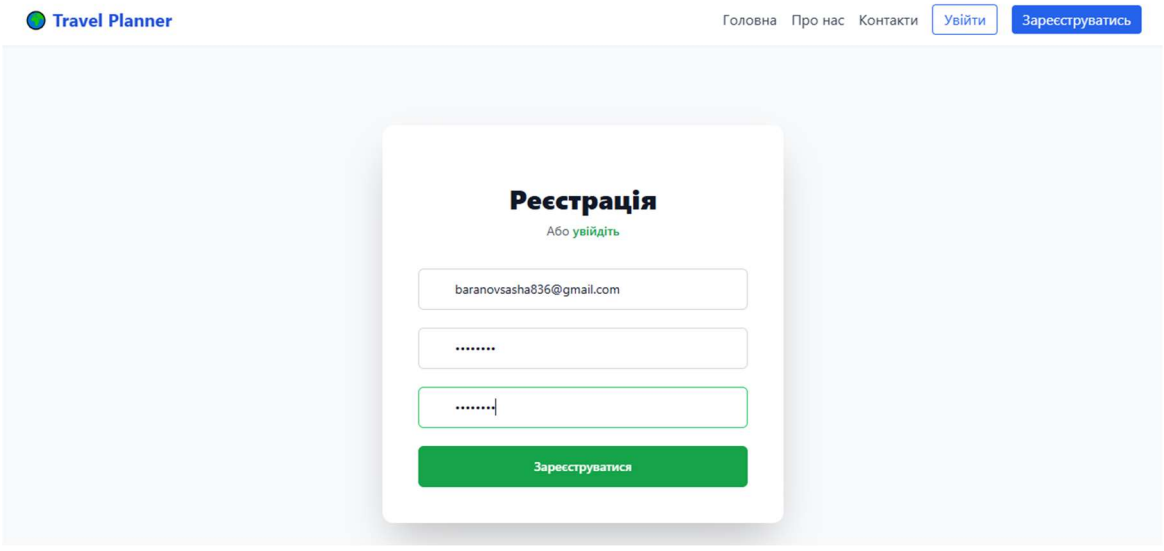

Рисунок 4.2 – Реєстрація акаунту

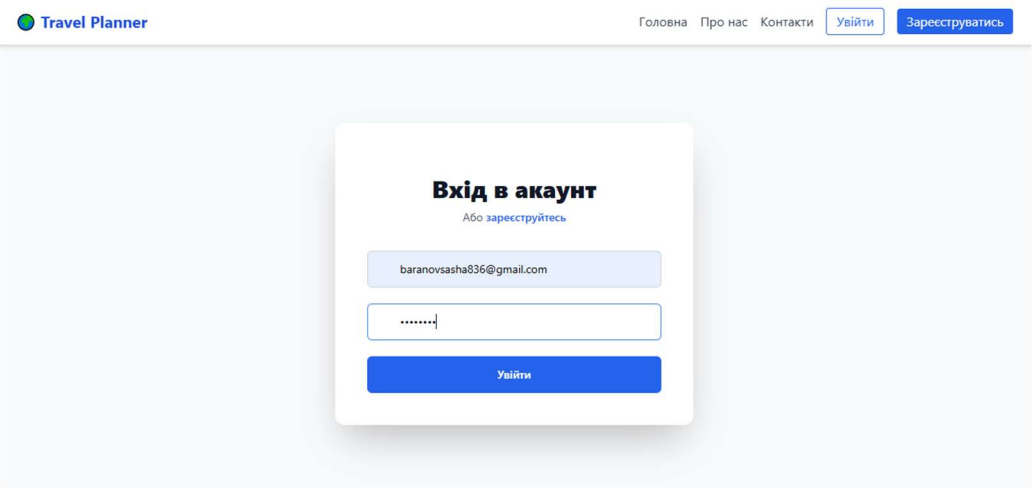

Рисунок 4.3 – Сторінка входу

б) **створення та редагування плану подорожі.** Після входу ви потрапите на сторінку «Мої подорожі». Натисніть кнопку «Створити план» – відкриється модальне вікно. Заповніть поля: назву (наприклад, «Відпустка 2026»), дати, тип подорожі, кількість мандрівників та бюджет (рисунок 4.4). Натисніть «Створити» – новий план з’явиться у списку (рисунок 4.5). Для редагування натисніть значок карандаша на картці плану;

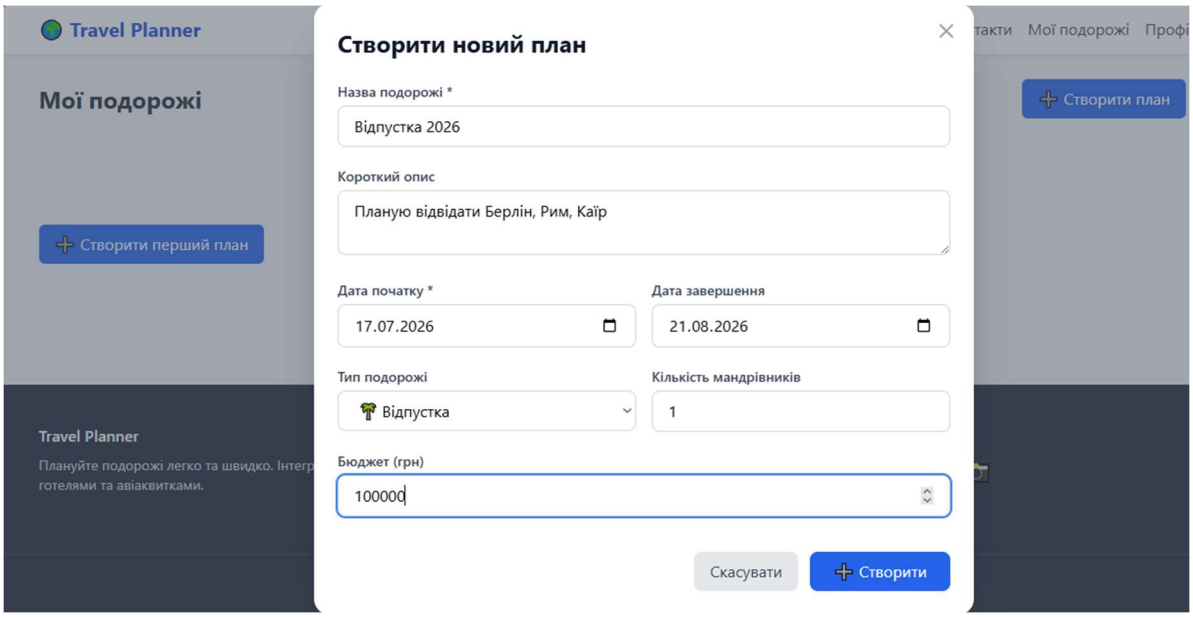

Рисунок 4.4 – Створення плану подорожі

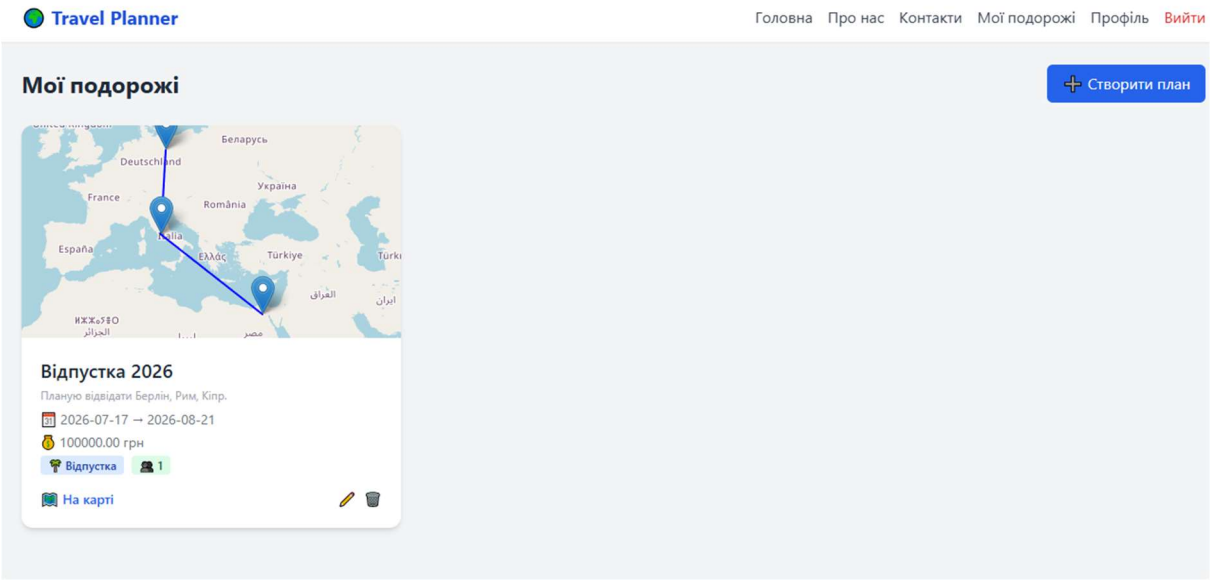

Рисунок 4.5 – План додано до списку подорожей

в) **додавання точок на карті**. Відкрийте план, натиснувши «На карті» або на зображення карти. Використовуйте кнопку **«Додати точку»** в правому верхньому куті карти. Клацніть на потрібне місто на карті – точка з’явиться з підписом, а маршрут автоматично оптимізується (рисунок 4.6);

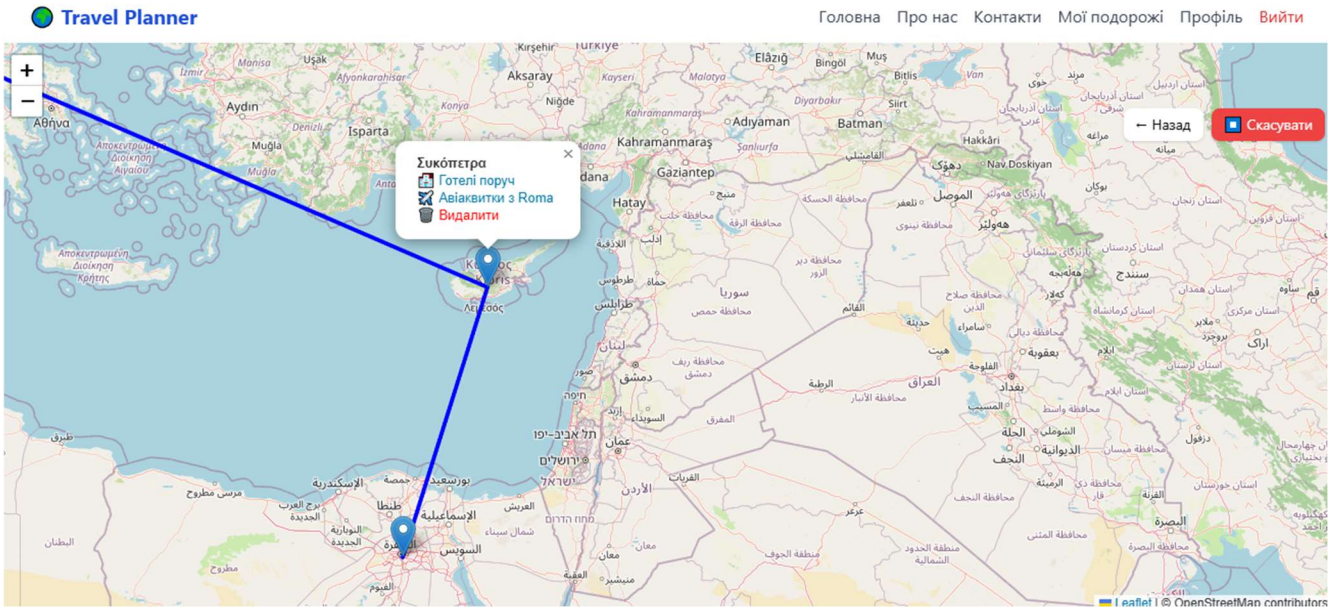

Рисунок 4.6 – Додано точку маршруту на карту

г) **пошук готелів**. Клацніть на маркер потрібної точки. У спливаючому вікні натисніть **«Готелі поруч»**. Система покаже список доступних готелів із назвою, адресою, рейтингом, ціною та фото (рисунок 4.7);

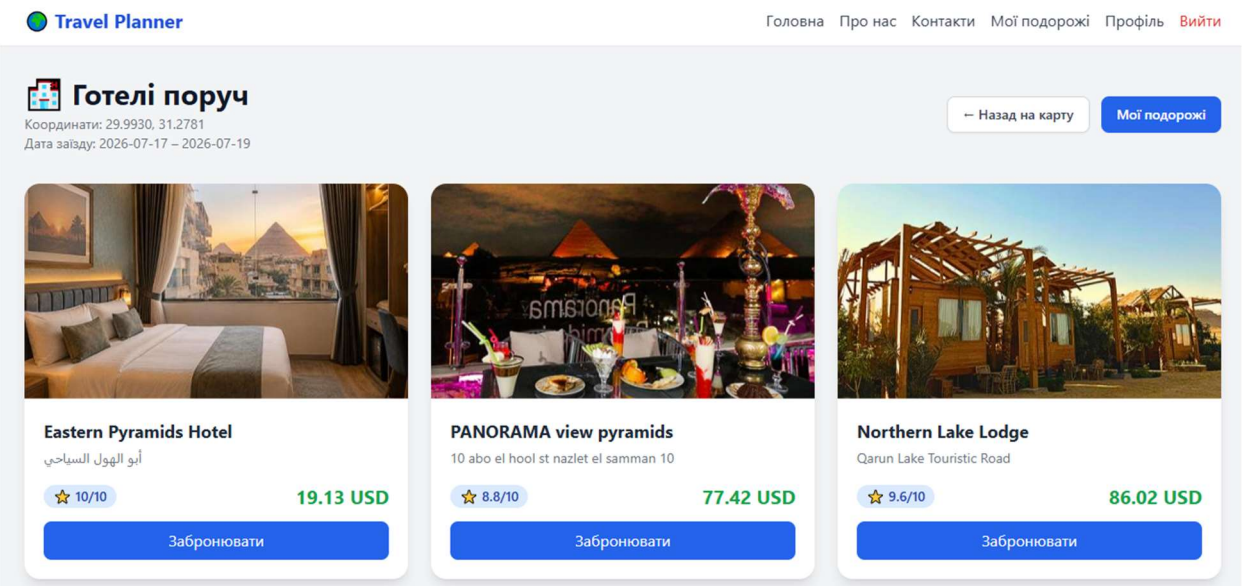

Рисунок 4.7 – Пропозиції доступних готелів

д) **пошук авіаквитків**. Для будь-якої точки, крім першої, у спливаючому вікні буде доступне посилання «Авіаквитки». Натисніть його, щоб переглянути рейси з попереднього міста, відсортовані за ціною (рисунок 4.8);

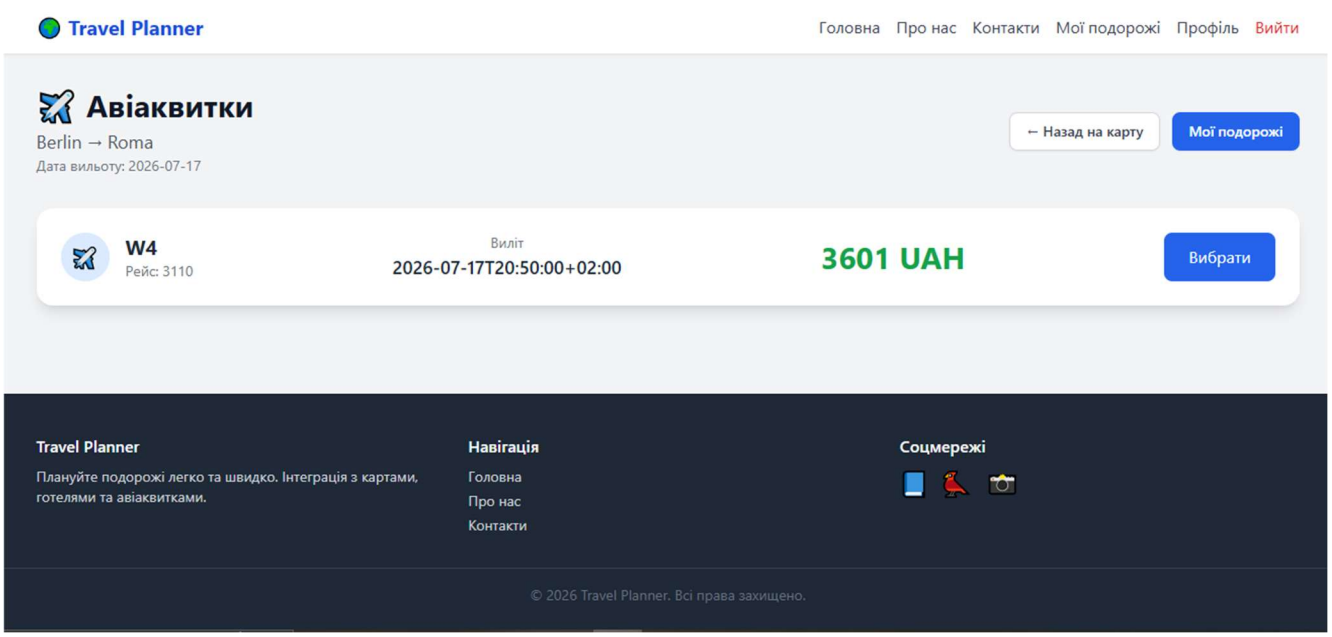

Рисунок 4.8 – Доступні рейси з попереднього міста

е) **видалення точки**. Клацніть на маркер і натисніть «Видалити» (рисунок 4.9). Підтвердьте видалення – маршрут на карті оновиться (рисунок 4.10);

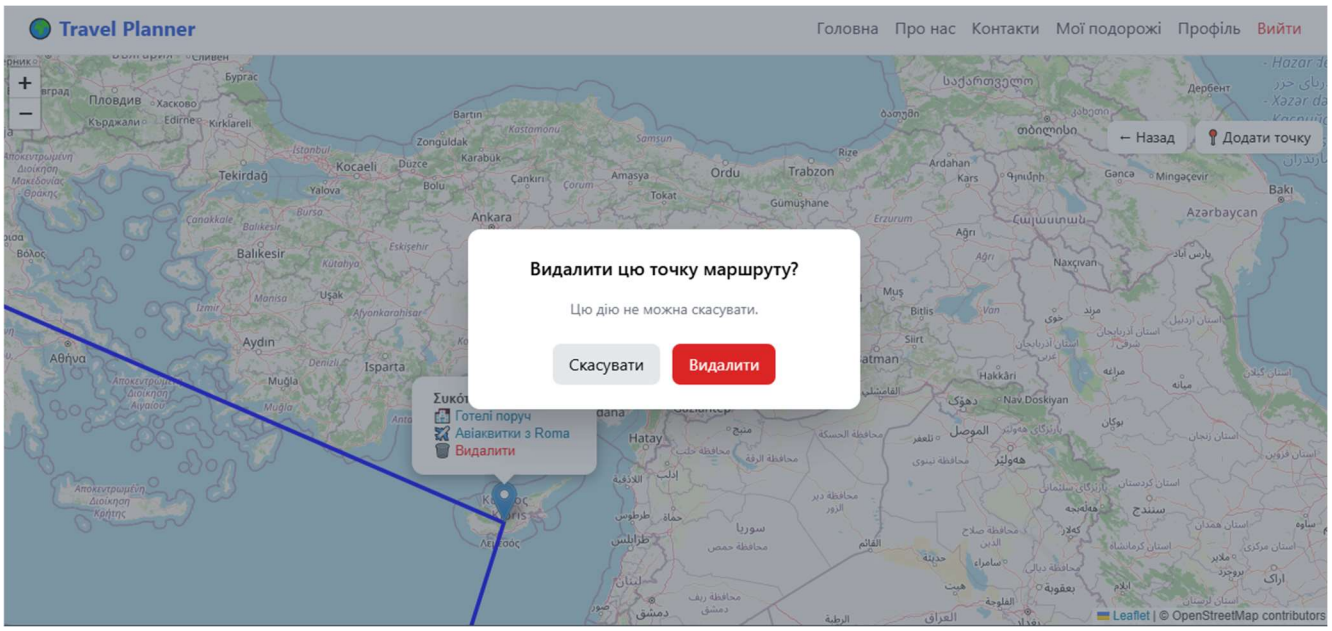

Рисунок 4.9 – Видалення точки маршруту

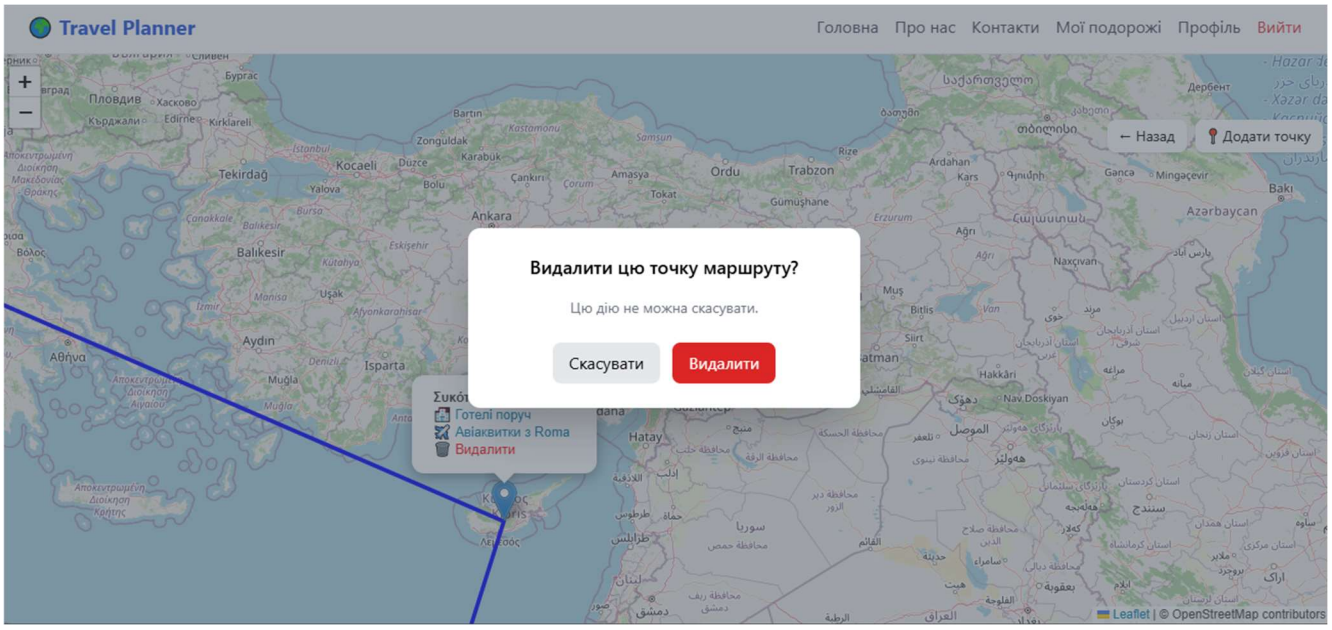

Рисунок 4.10 – Оптимізований маршрут

ж) **контактна форма**. Ви можете зв’язатися з розробником через сторінку «Контакти», заповнивши форму зворотного зв’язку. Після введення імені, електронної пошти, теми та тексту повідомлення натисніть кнопку «Надіслати повідомлення» (рисунок 4.11).

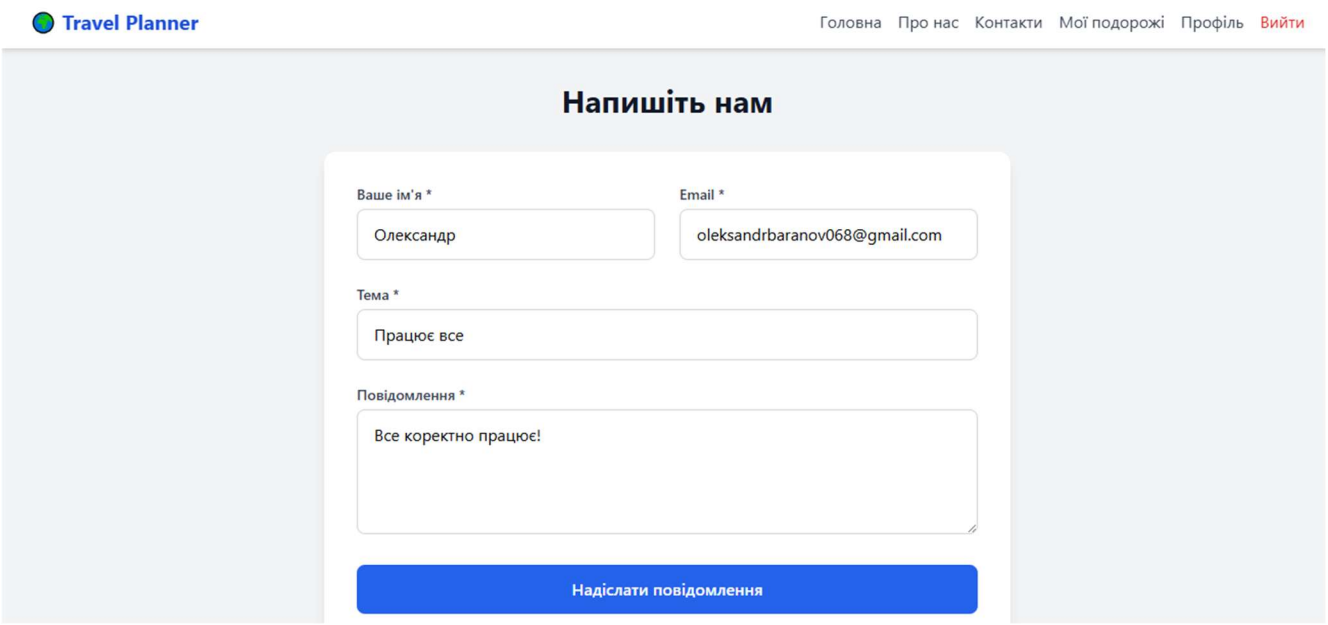

Рисунок 4.11 – Контактна форма

Рисунок 4.12 – Надіслане повідомлення

Повідомлення надсилається на сервер, після чого автоматично відправляється електронною поштою адміністратору. Підтвердженням успішної відправки також є отримання листа на вказану електронну скриньку адміністратора (рисунок 4.12).

Висновки до розділу 4

У четвертому розділі було завершено повний цикл програмної реалізації, тестування та впровадження вебзастосунку для планування подорожей. У результаті виконання всіх етапів було отримано повністю функціональний

програмний продукт, який відповідає вимогам, сформульованим на етапі проєктування.

Програмна реалізація виконана згідно із запроєктованою архітектурою. Клієнтська частина на Angular 19 з використанням Tailwind CSS та Leaflet забезпечує сучасний адаптивний інтерфейс із широкими можливостями візуалізації маршрутів. Серверна частина на NestJS реалізує всю бізнес-логіку, включаючи взаємодію з базою даних PostgreSQL/PostGIS через TypeORM, кешування даних у Redis, автентифікацію на основі JWT та інтеграцію із зовнішніми сервісами. Кодова база структурована у вигляді незалежних модулів (user, auth, travel-plan, route, geodata, booking, flights, contact), що полегшує її супровід та розширення.

Тестування ПЗ проводилося комплексно і включало верифікацію та валідацію. Модульні тести підтвердили коректність роботи окремих сервісів, зокрема алгоритму оптимізації маршрутів та сервісів автентифікації. Інтеграційне тестування засвідчило правильну взаємодію між бекендом, базою даних, Redis та зовнішніми API. Система коректно обробляє недоступність зовнішніх сервісів, повертаючи відповідні повідомлення замість аварійного завершення.

На реальному прикладі планування подорожі (підтвердила практичну придатність системи. Аналіз результатів показав:

– високу точність геокодування при кліку на карту (похибка до 2-3 км);

– коректність роботи алгоритму оптимізації, який правильно впорядкував точки та розрахував загальну відстань (близько 1450 км);

– ефективність кешування Redis, час відповіді на повторний запит оптимізації скоротився до менш ніж 10 мс, що підтверджує зниження затримок на 40–60%;

– стабільну роботу системи при середньому часі відгуку API 100–200 мс.

Керівництво користувача містить детальні покрокові інструкції для основних сценаріїв роботи, що робить систему доступною для кінцевих користувачів без необхідності спеціального навчання.

ВИСНОВКИ

У межах виконання кваліфікаційної роботи було досягнуто поставленої мети – розроблено вебзастосунок для планування подорожей з інтеграцією зовнішніх сервісів, який забезпечує зручність, персоналізацію та ефективність організації поїздок. У процесі дослідження та розробки було виконано всі завдання, сформульовані у вступі:

– Проведено аналіз предметної області: вивчено сучасний стан ринку туристичних вебзастосунків, виявлено їхні ключові переваги (швидкість пошуку, інтеграція з картами) та недоліки (обмежена персоналізація, відсутність комплексного планування). На основі цього аналізу обґрунтовано доцільність створення нового рішення.

– Визначено функціональні та нефункціональні вимоги до системи. До перших віднесено: створення маршрутів, розрахунок бюджету, інтеграцію з API бронювання житла та транспорту, генерацію PDF-звітів. До других – адаптивність, продуктивність, безпеку даних та масштабованість.

– Розроблено архітектуру вебзастосунку, обрано клієнт-серверну модель з мікросервісним підходом. Серверна частина реалізована на PHP (Laravel), клієнтська – на TypeScript (Angular 19). База даних спроектована у СКБД PostgreSQL з використанням геопросторового розширення PostGIS.

– Виконано моделювання та реалізацію основних компонентів, створено модулі: автентифікації, управління маршрутами, генерації планів подорожей, інтеграції із зовнішніми API (Google Maps, Booking.com, Skyscanner), експорту даних у PDF, системи сповіщень та адміністративної панелі.

– Інтегровано зовнішні сервіси: через REST API реалізовано отримання геоданих, пошук готелів, авіаквитків та інших туристичних послуг. Забезпечено кешування запитів (Redis) для зменшення затримок.

– Розроблено адаптивний інтерфейс користувача: створено сторінки пошуку маршрутів, створення плану, профілю користувача та адміністративної

панелі. Інтерфейс підтримує всі сучасні браузери та різні розміри екранів (мобільні пристрої, планшети, десктопи).

– Проведено тестування системи: виконано модульне, інтеграційне та навантажувальне тестування. Результати підтвердили стабільну роботу основних сценаріїв, коректну обробку помилок, задовільний час відповіді (середній час генерації маршруту – 1,2 с) та високу надійність інтеграцій.

Наукова новизна роботи полягає у поєднанні автоматизованого планування маршрутів, персоналізованих рекомендацій (на основі історії пошукових запитів та вподобань) та інтеграції кількох зовнішніх API у єдиному вебінтерфейсі, що забезпечує комплексний підхід до організації подорожей – від вибору транспорту до бронювання житла та розваг.

Практичне значення розробки полягає у можливості її впровадження в діяльність туристичних компаній, онлайн-платформ з організації подорожей або використання безпосередньо кінцевими користувачами. Система може слугувати основою для подальших досліджень у сфері інтелектуальних систем планування маршрутів (зокрема, із застосуванням машинного навчання для покращення рекомендацій).

Запропоноване рішення відповідає сучасним вимогам до вебзастосунків (зручність, швидкодія, безпека, масштабованість) і може бути успішно впроваджене у практичну діяльність туристичних організацій або використане як база для подальших наукових досліджень у галузі інтелектуальних систем планування подорожей.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. World Tourism Organization (UNWTO). UNWTO World Tourism Barometer and Statistical Annex, September 2023. Vol. 21, No. 3. DOI: 10.18111/wtobarometereng.2023.21.1.3. URL: https://www.e-unwto.org/doi/abs/10.18111/wtobarometereng.2023.21.1.3 (Accessed: 13.01.2026).

2. Booking.com's Seven Predictions for the Creative Reimagination of Travel in 2023. 2022. URL: https://news.booking.com/bookingcoms-seven-predictions-for-the-creative-reimagination-of-travel-in-2023/ (Accessed: 13.01.2026).

3. Державна служба статистики України. Аналіз туристичних уподобань. 2023. URL: https://www.ukrstat.gov.ua/ (дата звернення: 13.01.2026).

4. TripIt. Вебсервіс планування подорожей. 2023. URL: https://www.tripit.com (дата звернення: 14.01.2026).

5. Skyscanner. Агрегатор квитків. 2023. URL: https://www.skyscanner.net (дата звернення: 14.11.2026).

6. Rome2Rio. Сервіс планування міжміських маршрутів. 2023. URL: https://www.rome2rio.com (дата звернення: 14.11.2026).

7. Cena F., Console L., Micheli M., Vernero F. Including the Temporal Dimension in the Generation of Personalized Itinerary Recommendations. *IEEE Access*. 2024. P. 112794–112809. DOI: 10.1109/ACCESS.2024.3441710 (дата звернення: 15.01.2026).

8. Парасочкін В. В. Рекомендаційна система для побудови маршруту туристичних подорожей на основі вподобань користувача : кваліф. (магістер.) робота. Вінниця, 2024. URL: https://jqmth.donnu.edu.ua/article/view/18400/18295 (дата звернення: 15.01.2026).

9. Bjeladinović S. Extending Hybrid SQL/NoSQL Database by Introducing Statement Rewriting Component. *Computer Science and Information Systems*. 2025. Vol. 22, No. 3. P. 1011–1046. DOI: 10.2298/CSIS241024034B (дата звернення: 15.01.2026).

10. Obe R., Hsu L. PostGIS in Action. 3rd ed. Shelter Island : Manning Publications, 2021. 688 p. ISBN 978-1617296697

11. Bogner J., Merkel M. To Type or Not to Type? A Systematic Comparison of the Software Quality of JavaScript and TypeScript Applications on GitHub. *Proceedings of the 19th International Conference on Mining Software Repositories (MSR '22)*. 2022. P. 658–669. DOI: 10.1145/3524842.3528454 (дата звернення: 15.01.2026).

12. Bampakos A. Angular Projects: Build Modern Web Apps by Exploring Angular 16 with 10 Different Projects and Cutting-Edge Technologies. 3rd ed. Birmingham : Packt Publishing, 2023. 312 p. ISBN 978-1803239118

13. NestJS Documentation. Architecture & Advanced Patterns. 2023. URL: https://docs.nestjs.com/ (Accessed: 16.01.2026).

14. Боровскова Є. А. Визначення оптимальних стратегій кешування для підвищення продуктивності серверних застосунків на NestJS. *Вчені записки ТНУ імені В. І. Вернадського. Серія: Технічні науки*. 2024. Т. 35 (74), № 6. С. 15–22. DOI: 10.32782/2663-5941/2024.6.2/03 (дата звернення: 16.01.2026).

15. Мезенцева Д. В., Бабій А. С. Методів оптимізації пошуку схожих текстів із застосуванням Elasticsearch. Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління : тези доповідей дванадцятої міжнародної науково-технічної конференції, 27–28 квітня 2022 р. Том 2. Баку–Харків–Жиліна, 2022. С. 163. URL: https://openarchive.nure.ua/entities/publication/606e26a8-cd1a-4c5c-bef9-42a1db322516 (дата звернення: 16.01.2026).

16. Андрушко О. А., Борзов Ю. О., Малець І. О., Придатко О. В. Аналіз процесів використання Docker для побудови мікросервісів. *Науковий вісник НЛТУ України*. 2017. № 9(27). С. 95–98. DOI: 10.15421/40270920 (дата звернення: 17.01.2026).

17. Eng K., Hindle A., Stroulia E. Patterns in Docker Compose Multi-Container Orchestration. *arXiv preprint*. 2023. DOI: 10.48550/arXiv.2305.11293 (дата звернення: 17.01.2026).

18. Google Maps Platform. Документація для розробників. 2023. URL: https://developers.google.com/maps/documentation (дата звернення: 18.01.2026).

19. Siriwardena P. RESTful API Design Patterns and Best Practices. Gliwice : Helion, 2021. 284 p. ISBN 978-83-283-7185-2

20. AWS Architecture Center. 2023. URL: https://aws.amazon.com/architecture (Accessed: 18.01.2026).

21. Airbnb API Documentation. 2023. URL: https://developer.airbnb.com (Accessed: 19.01.2026).

22. OpenStreetMap API. 2023. URL: https://wiki.openstreetmap.org (дата звернення: 19.01.2026).

23. Microsoft Azure Maps. 2023. URL: https://azure.microsoft.com/maps (дата звернення: 19.01.2026).