Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система для організації та управління груповими подорожами. Клієнтська частина
(тема)

Виконав:
здобувач 4 року навчання
групи ПЗПІ-21-3

Еліна ЧЕРКАСОВА
(Власне ім'я, ПРІЗВИЩЕ)

Спеціальність
121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. Дмитро КОЛЕСНИКОВ
(посада, Власне ім'я, ПРІЗВИЩЕ)

Допускається до захисту
Зав. кафедри _____ Кирило СМЕЛЯКОВ
(підпис) (Власне ім'я, ПРІЗВИЩЕ)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Черкасовій Еліні Сергіївні
(прізвище, ім’я, по батькові)

1. Тема роботи Програмна система для організації та управління груповими подорожами. Клієнтська частина
Затверджена наказом по університету від 397Ст від 19.05.2025
2. Термін подання студентом роботи до екзаменаційної комісії 13.06.2025
3. Вихідні дані до роботи Розробити програмну систему, котра надаватиме функції для мандрівників, мовою програмування JavaScript з використанням бібліотеки React.js.
4. Перелік питань, що потрібно опрацювати в роботі
Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	20.05.2025	*виконано*
2	Створення специфікації ПЗ	21.05.2025	*виконано*
3	Проектування ПЗ	22.05.2025	*виконано*
4	Розробка ПЗ	23.05.2025	*виконано*
5	Тестування ПЗ	30.05.2025	*виконано*
6	Оформлення пояснювальної записки	31.05.2025	*виконано*
7	Підготовка презентації та доповіді	04.06.2025	*виконано*
8	Попередній захист	10.06.2025	*виконано*
9	Нормоконтроль, рецензування	10.06.2025	*виконано*
10	Здача роботи у електронний архів	10.06.2025	*виконано*
11	Допуск до захисту у зав. кафедри	11.06.2025	*виконано*

Дата видачі завдання « 19 » « травня » 2025р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Дмитро КОЛЕСНИКОВ
(підпис) (посада, Власне ім’я, ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра, 93 стор., 33 рис., 16 джерел.

ВЕБЗАСТОСУНОК, ГРУПОВІ ПОДОРОЖІ, ІНТЕРФЕЙС, МОНІТОРИНГ, РЕАКТ, КОМПОНЕНТНИЙ ПІДХІД

Об’єкт розробки - клієнтська частина вебзастосунку TravelMate для планування та супроводу групових подорожей.

Мета роботи – створити функціональність, що забезпечує взаємодію з серверною частиною системи, включаючи управління подорожами, профілем, відображення маршруту та чату.

Методи реалізації включають використання технологій React, бібліотек Leaflet для роботи з інтерактивними картами, а також REST API для взаємодії з бекендом.

В результаті роботи реалізовано клієнтський інтерфейс вебзастосунку, який забезпечує реєстрацію та автентифікацію користувачів, формування та перегляд маршрутів, обмін повідомленнями в групі, перегляд статистики стану учасників подорожі.

ABSTRACT

COMPONENT APPROACH, GROUP TRAVEL, INTERFACE, MONITORING, REACT, WEB APPLICATION

The object of development is the client part of the TravelMate web application for planning and supporting group trips.

The purpose of the work is to create functionality that provides interaction with the server part of the system, including travel management, profile, route display and chat.

Implementation methods include the use of React technologies, Leaflet libraries for working with interactive maps, as well as REST API for interacting with the backend.

As a result of the work, a client interface of the web application was implemented, which provides user registration and authentication, formation and viewing of routes, exchange of messages in the group, and viewing statistics of the status of trip participants.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API - Application Programming Interface - інтерфейс прикладного програмування

DOM - Document Object Model - об’єктна модель документа

GPS - Global Positioning System - глобальна система позиціонування

HTTP - Hypertext Transfer Protocol - протокол передачі гіпертексту

ID - Identifier - ідентифікатор

JWT - JSON Web Token - вебтокен на основі JSON

REST API - Representational State Transfer API - інтерфейс прикладного програмування на основі REST-архітектури

SPA - Single Page Application - односторінковий застосунок

SOS - Save Our Souls (міжнародний сигнал тривоги)

TCP - Transmission Control Protocol - протокол керування передачею

UI - User Interface - користувацький інтерфейс

UML - Unified Modeling Language - уніфікована мова моделювання

UX - User Experience - користувацький досвід

ВСТУП

Розвиток нових інформаційних технологій суттєво впливає на те, як плануються та організовуються види дозвілля, такі як подорожі. На тлі глобалізації та цифровізації суспільства зростає попит на ресурси, які сприяють легкому плануванню, організації та проведенню групових турів. Об'єднання людей для туру, організація маршрутів та сприяння легкому спілкуванню між мандрівниками під час руху стають основними проблемами.

Незважаючи на велику кількість туристичних послуг, вони зазвичай орієнтовані на мандрівників-одинаків або не призначені для забезпечення ефективної групової взаємодії на задовільному рівні.

З огляду на ці проблеми, розробка програмного забезпечення для підтримки групових турів - від планування до перегляду статистики стану здоров'я мандрівників - є актуальним завданням.

Темою кваліфікаційної роботи є розробка клієнтської частини програмної системи TravelMate для планування групових турів.

Основною функцією програмного продукту є надання користувачам можливості:

- планувати маршрут та узгоджувати деталі поїздки з іншими мандрівниками;
- надсилати повідомлення між мандрівниками в режимі реального часу;
- отримувати статистику поїздки;
- створювати свої особисті подорожі та шукати мандрівників для спільної поїздки.

Метою роботи є розробка практичного та зручного веб-застосунку, що представляє клієнтську частину програмної системи TravelMate та реалізований з використанням сучасних веб-технологій.

Для реалізації програмного продукту використовуються мова програмування JavaScript для програмного забезпечення, бібліотека ReactJS, система контролю версій GitHub та середовище розробки VS Code.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Сфера туристичних технологій активно розвивається під впливом глобальної цифровізації. Згідно з аналітичним звітом [1], глобальний ринок програмного забезпечення для подорожей у 2023 році оцінювався в 9.04 мільярда доларів США, а до 2032 року прогнозується його зростання до 22.48 мільярда, що відповідає середньорічному темпу приросту 10.65%. Зростає популярність групових подорожей - спільних вилазок друзів, туристичних клубів, студентських груп та сімейних турів. Разом із цим постають нові виклики: координація дій великої кількості учасників, своєчасний обмін інформацією, контроль за здоров’ям та ретельне планування.

Більшість сучасних інструментів для подорожей (Travefy, Wanderlog, Let's Jetty тощо) орієнтовані переважно на індивідуальних мандрівників або мають обмежену підтримку групової взаємодії. Функціонал супроводу подорожі в реальному часі - реалізовано частково або взагалі відсутній [2].

Організація групової мандрівки вимагає оперативного вирішення таких завдань, як: планування маршруту, спільне бронювання транспорту та житла, обмін повідомленнями, а також збереження медичних даних та даних з фітнес-трекерів. Однак у реальних умовах ці процеси часто реалізуються через декілька різних застосунків і вручну, що створює плутанину й підвищує ризики.

У відповідь на ці виклики виникає потреба в створенні цілісного, інноваційного цифрового рішення, що дозволить:

- планувати маршрут і активності групи;
- моніторити пульс або інші біометричні показники через смарт-пристрої;
- мати швидкий доступ до повідомлень для групової безпеки;
- забезпечити комунікацію всередині групи.

Згідно з систематичним оглядом мобільних туристичних застосунків [3], до основних смарт-функцій сучасних рішень належать: використання геолокації, персоналізовані маршрути, рекомендації на основі штучного інтелекту, інтеграція з календарями та сенсорами мобільних пристроїв. Дослідники також звертають увагу на зростаюче значення таких функцій, як офлайн-доступ, динамічне оновлення маршруту та інтерфейси для групової взаємодії. Ці тенденції підтверджують потребу у всеохоплюючому програмному забезпеченні, яке б відповідало очікуванням сучасного користувача та дозволяло ефективно організовувати й супроводжувати групові подорожі.

TravelMate орієнтований на організаторів подорожей, що відповідальні за координацію групи, планування маршруту, безпеку та зв’язок з учасниками, а ткож на учасників мандрівок, які потребують навігації, сповіщень про зміну маршруту, можливості повідомити про проблеми, переглядати маршрут.

У центрі предметної області - безпека, зручність та координація дій під час колективних подорожей. Саме тому інтеграція таких технологій, як GPS-моніторинг, чат, пульсометри та геомітки є не лише доцільною, а й необхідною для створення сучасного рішення.

Отже, актуальність дослідження предметної галузі полягає у необхідності цифровізації та автоматизації процесів, пов’язаних з організацією групових мандрівок, забезпеченням безпеки учасників та ефективної комунікації. Застосування інтелектуальних рішень у цій галузі дозволить підвищити комфорт і зменшити ризики під час подорожей.

1.2 Виявлення та вирішення проблем

У процесі аналізу предметної галузі та існуючих рішень для планування подорожей виявлено низку проблем, які суттєво впливають на зручність і безпеку групових мандрівок. Більшість наявних застосунків орієнтовані на

індивідуальних туристів або не забезпечують достатнього рівня інтеграції функцій, критичних для групової подорожі.

Дослідження показують, що учасники групових подорожей мають специфічні вимоги до комунікації, взаємної координації та спільного прийняття рішень [4]. На відміну від індивідуальних мандрівок, у груповому середовищі виникає потреба в інструментах для колективного планування, розподілу ролей та швидкого обміну інформацією між усіма учасниками. Наявність централізованої платформи з інтегрованими функціями значно підвищує згуртованість групи, зменшує непорозуміння та підвищує загальну ефективність подорожі.

1.2.1 Проблеми предметної галузі

По-перше, існуючі сервіси пропонують часткову функціональність: планування маршруту, бронювання, ведення журналу подорожі.

Усі зміни до маршруту, зупинок, часу збору тощо доводиться передавати вручну. Це створює плутанину, особливо в динамічних умовах подорожі. Потенціал сучасних технологій не використовується достатньо. Сучасні технологічні тренди, такі як концепція Tourism 4.0 [5], що охоплює використання Інтернету речей (IoT), відкривають нові можливості для розвитку інтелектуальних туристичних сервісів.

TravelMate інтегрує ці технології для створення розумної системи супроводу групових подорожей, що дозволяє в режимі реального часу моніторити стан учасників, оптимізувати маршрути, аналізувати дані про поведінку групи та швидко реагувати на непередбачені ситуації.

TravelMate дозволяє створювати спільний маршрут, розподіляти завдання, додавати учасників, бронювати місця. Вбудований чат дозволяє швидко координувати дії всієї групи.

Таким чином, TravelMate відповідає на реальні потреби групових мандрівників, пропонуючи зручний, надійний і безпечний інструмент супроводу подорожей. Його впровадження підвищує ефективність комунікації, покращує взаємодію між учасниками та мінімізує ризики, пов’язані з подорожами.

1.2.2 Аналіз існуючих рішень

У рамках аналізу існуючих рішень для групових подорожей були розглянуті популярні застосунки, які відповідають на потреби організації групових мандрівок: Travefy, Wanderlog, і Let's Jetty. Кожен з цих сервісів пропонує різний функціонал для планування подорожей, але всі вони мають свої обмеження, особливо у контексті забезпечення безпеки та інтеграції всіх необхідних функцій для групових подорожей.

Travefy [6] - один з популярних платформ для планування групових подорожей (див. рис. 1.1). Застосунок дозволяє створювати детальні плани, організовувати спільне бронювання, ділитися інформацією про маршрути та активності. Основні функції включають:

- планування маршруту (детальний план подорожі, місць для відвідування);
- спільне бронювання готелів;
- спільний календар і нагадування.

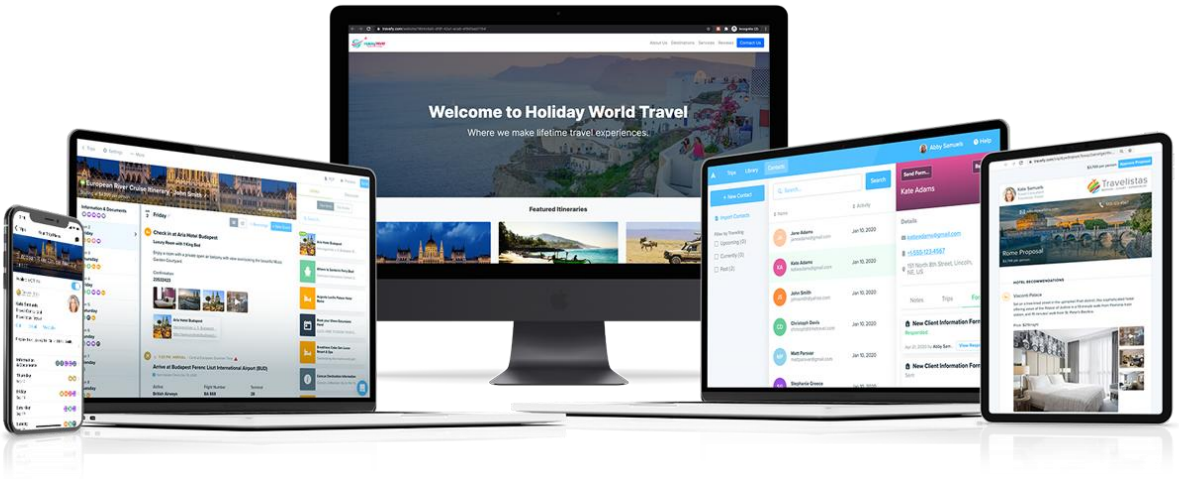

Рисунок 1.1 - Інтерфейс програмної системи “Travefy” (рисунок виконаний самостійно)

Wanderlog [7] є ще одним популярним інструментом для планування подорожей (див. рис. 1.2). Він пропонує вебінтерфейс із мапами, розкладом і спільним редагуванням, але більшість функцій доступні лише після авторизації, інтерфейс перевантажений і не завжди оптимізований для слабких пристроїв. Цей застосунок дозволяє створювати інтерактивні маршрути, зберігати місця для відвідування, а також ділитися планами з іншими учасниками. Основні функції:

- дозволяє будувати маршрути з можливістю додавання точок інтересу, активностей та деталізованої інформації про подорож;
- користувачі можуть обмінюватися інформацією, бронювати спільно готелі та транспорту;
- може генерувати рекомендації щодо маршрутів і розкладів на основі даних інших користувачів.

Однак, як і Travefy, Wanderlog має обмежену функціональність - там немає інтеграції з пристроями для моніторингу здоров’я учасників.

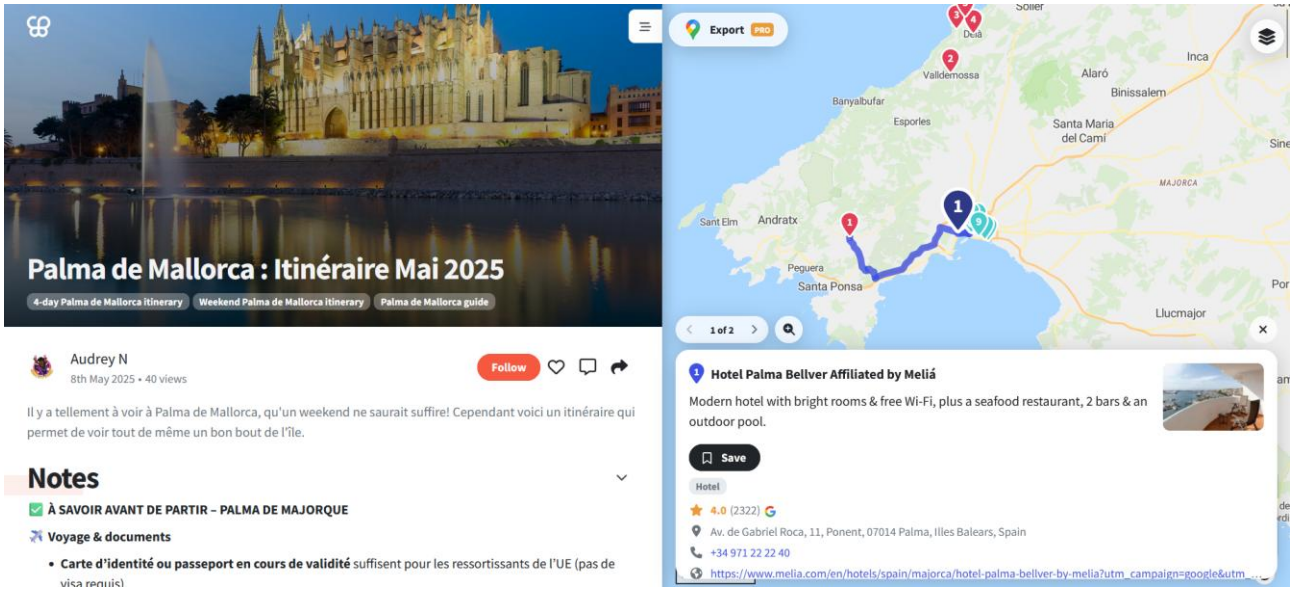

Рисунок 1.2 - Інтерфейс програмної системи “Wanderlog” (рисунок виконаний самостійно)

Ще один застосунок, Let's Jetty [8] - орієнтований на організацію групових подорожей для молодіжних та соціальних груп (див. рис. 1.3). Він дозволяє створювати плани подорожей, проводити опитування, організовувати спільні бронювання, а також забезпечує зручний обмін повідомленнями між учасниками. Основні можливості включають:

- планування маршруту: Let's Jetty дозволяє створювати маршрути і бронювати основні послуги;
- вбудований чат для комунікації між учасниками;
- організація групових поїздок.

Попри зручність для молодіжних груп, Let's Jetty має значні недоліки, коли мова йде про групи з різними потребами:

- відсутність моніторингу здоров’я та безпеки учасників;
- немає функції відстеження локації в реальному часі;
- відсутність офлайн-функціональності для роботи в умовах обмеженого інтернет-зв'язку;

– немає функцій SOS, що робить цей сервіс не найкращим вибором для подорожей у важких умовах (наприклад, в горах чи в екзотичних місцях).

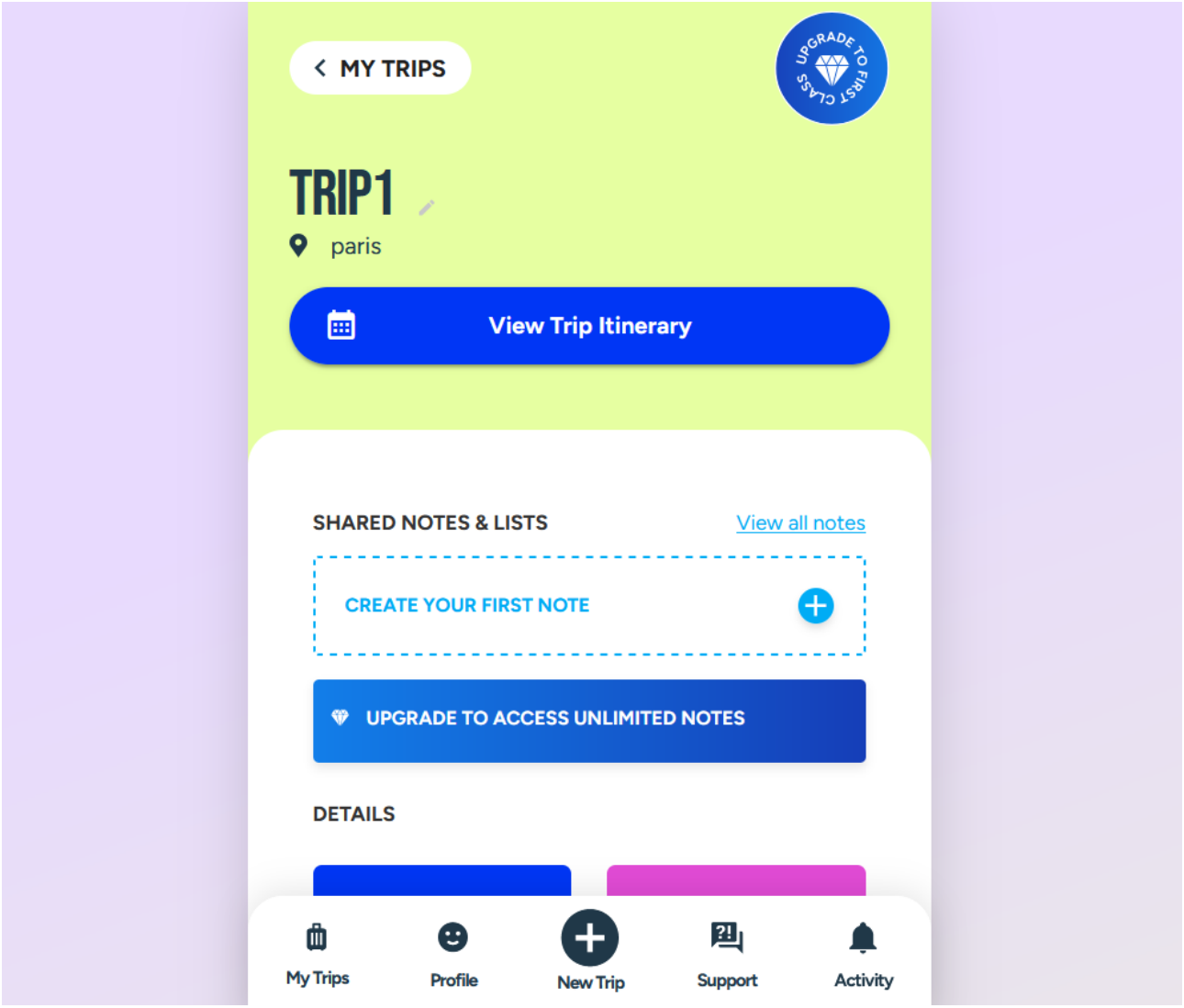

Рисунок 1.3 - Інтерфейс програмної системи “Let's Jetty” (рисунок виконаний самостійно)

Загалом, жоден з розглянутих застосунків не забезпечує комплексної підтримки для групових подорожей, особливо в контексті безпеки та моніторингу здоров’я учасників. TravelMate має явні переваги в плані інтеграції функцій безпеки, відстеження локації учасників і моніторингу їхнього стану здоров’я, що робить його найбільш перспективним рішенням для організації групових подорожей у реальному часі.

1.3 Постановка задачі

Метою цієї кваліфікаційної роботи є розробка клієнтської частини вебплатформи TravelMate, що забезпечує зручну та безпечну взаємодію користувачів із системою для організації групових подорожей. Застосунок має реалізовувати сучасний, інтуїтивно зрозумілий інтерфейс для створення та керування подорожами, обміну повідомленнями та перегляду статистики подорожі. Клієнтська частина тісно взаємодіє із серверною через REST API, надаючи кінцевим користувачам доступ до всіх функцій платформи.

Основним завданням клієнтської частини є забезпечення повноцінного інтерфейсу користувача, що дозволяє взаємодіяти з основними компонентами системи: обліковими записами, групами подорожей, чатами та участями.

Для реалізації цієї мети вебзастосунок має забезпечувати реалізацію таких ключових функцій:

- реєстрація та автентифікація користувачів із підтримкою JWT-токенів;
- створення, перегляд, редагування та видалення подорожей;
- приєднання користувачів до подорожей;
- відображення маршруту подорожі на інтерактивній карті;
- перегляд статистичних даних про серцебиття та інші телеметричні показники учасників під час подорожі, що минула;
- реалізація групового текстового чату між учасниками подорожі;
- перегляд статистики та підсумків завершених подорожей;
- забезпечення зручної навігації;
- виведення повідомлень про помилки, статуси операцій та інструкцій.

У межах розробки клієнтської частини необхідно вирішити такі задачі:

- спроєктувати загальну архітектуру клієнтського застосунку на базі JavaScript-фреймворку React з урахуванням модульності та повторного використання компонентів;
- реалізувати систему авторизації та автентифікації з використанням JWT;

- забезпечити інтеграцію з API серверної частини для отримання та відправлення даних щодо користувачів, подорожей та чату;
- реалізувати роботу HTTP-запитів (polling) для оновлення чату у режимі реального часу;
- створити UI-компоненти для візуалізації маршруту, учасників, чату, профілю і карти;
- впровадити обробку помилок, станів завантаження та тайм-аутів під час взаємодії з сервером;
- задокументувати структуру інтерфейсів, маршрути, ключові компоненти та логіку взаємодії з API.

Розробка клієнтської сторони системи здійснюватиметься з використанням бібліотеки React, мови програмування Javascript. Для роботи з API буде застосовано Axios.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Загальна характеристика

Клієнтська частина програмної системи TravelMate є вебзастосунком, що забезпечує графічний інтерфейс для взаємодії користувача з функціоналом серверної частини платформи. Система орієнтована на учасників групових подорожей і дозволяє створювати маршрут, переглядати статистичні дані про стан здоров’я учасників, надсилати повідомлення та керувати подорожами.

Вебінтерфейс повинен бути адаптивним, зручним для користувача, безпечним і продуктивним при роботі з реальними даними.

2.2 Функціональні вимоги до програмної системи

Функціональні вимоги, які повинен задовольняти продукт наведено нижче.

2.2.1 FR-01 Реєстрація користувача

Користувач має можливість створити новий обліковий запис шляхом заповнення форми реєстрації з введенням обов’язкових полів (ім’я, прізвище, email, ім’я користувача і пароль). Дані надсилаються через REST API на сервер, де перевіряється унікальність email та створюється запис у базі. При успішній реєстрації користувач перенаправляється на сторінку входу.

2.2.2 FR-02 Авторизація користувача

Користувач вводить свої облікові дані (email, пароль) у форму входу. Після успішної перевірки на сервері клієнт отримує JWT-токен [9], що зберігається у локальному сховищі браузера. Усі запити до захищених маршрутів перевіряють наявність і дійсність токена. При його відсутності або завершенні терміну дії користувач автоматично перенаправляється на екран авторизації.

2.2.3 FR-03 Перегляд списку подорожей

Після входу користувач бачить перелік всіх подорожей, що існують. Дані отримуються з сервера у вигляді списку з короткою інформацією (назва, дати, кількість учасників, статус).

2.2.4 FR-04 Створення нової подорожі

Організатор може створити нову подорож через форму, де вводяться: назва подорожі, дата початку та завершення, короткий опис, кількість учасників, екстремальність, локації старту, фінішу та проміжні. Після надсилання форми на сервер створюється відповідний запис.

2.2.5 FR-05 Редагування / видалення подорожі

Організатор має змогу змінити параметри подорожі (дати, назву, опис, маршрут, екстремальність, кількість учасників, статус подорожі). Якщо подорож ще не розпочалась, її також можна скасувати. Ці дії доступні лише користувачам

із роллю «організатор», що мають права на відповідну подорож. Зміни синхронізуються з базою даних через API.

2.2.6 FR-06 Приєднання до подорожі

Користувач може приєднатися до вже створеної подорожі подавши запит на участь, який має бути схвалений або відхилений організатором.

2.2.7 FR-7 Чат учасників подорожі

Вебінтерфейс містить вбудований текстовий чат, що дозволяє учасникам обмінюватися повідомленнями під час подорожі. Повідомлення можуть бути відправлені у спільну групу, сортуються за часом і зберігаються з позначкою про автора.

2.2.8 FR-8 Перегляд статистики по подорожі

Після завершення подорожі користувач може переглянути зведену інформацію: загальну тривалість, пройдений маршрут на карті, його відстань, середній пульс, середню швидкість.

2.2.9 FR-9 Повідомлення про статус операцій

Інтерфейс клієнта повідомляє користувача про результат усіх важливих дій: помилки авторизації, втрату з’єднання, успішне створення подорожі тощо. Повідомлення відображаються у вигляді діалогових вікон.

2.2.10 FR-10 Редагування профілю користувача

Користувачу повинен бути доступний функціонал для зміни своїх даних, таких як – ім’я, прізвище, номер телефону, біографія, країна та місто проживання, пароль і аватар.

2.2.11 FR-11 Функціонал, що доступний тільки адміністратору

Адміністратор бачить на екрані список всіх користувачів та подорожей. Він має можливість заблокувати або розблокувати користувача або подорож.

2.2.12 FR-12 Схвалення або відхилення заявки на участь у подорожі

Організатор бачить список бажаючих приєднатися до подорожі і може прийняти або відхилити заявку на приєднання від іншого користувача.

2.2.13 FR-13 Відображення пунктів призначення на карті

Організатор подорожі створює точки на карті з ім’ям, описом та порядковим номером, що бачать всі користувачі. Кожна точка означає місце відвідування мандрівників.

2.3 Нефункціональні вимоги до програмної системи

Нефункціональні вимоги встановлюють функціональну якість роботи програмного продукту. Загалом, вони мають підвищити надійність,

продуктивність та зручність використання програмного забезпечення, а також покращити його якості, такі як масштабованість та відновлюваність. Ці аспекти є основою для побудови архітектури програмних систем [10].

В ході роботи сформульовано такий перелік нефункціональних вимог:

- продуктивність (час відгуку на запити користувача не повинен перевищувати 5 секунд в межах нормального навантаження; оновлення координат та біометричних даних учасників має виконуватись не рідше ніж раз на 20 секунд);
- масштабованість (архітектура застосунку побудована з використанням компонентної моделі React із підтримкою lazy loading, що дозволяє легко додавати нові функціональні модулі й маршрути без порушення структури проєкту);
- надійність (система повинна мати механізми обробки втрати з’єднання з сервером: виведення повідомлень, повторні спроби відправлення запитів);
- портативність та сумісність (система має коректно функціонувати в останніх стабільних версіях браузерів Google Chrome, Mozilla Firefox, Microsoft Edge та Safari);
- безпека (всі запити повинні здійснюватись по протоколу HTTPS; авторизація реалізується через JWT з перевіркою ролі користувача; доступ до приватних сторінок та біометричних даних дозволений лише після успішної автентифікації);
- зручність використання (застосунок має інтуїтивну навігацію по елементах).

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування програмного забезпечення

Перед початком безпосередньої реалізації клієнтської частини вебзастосунку TravelMate необхідно визначити структуру взаємодії користувачів із системою. Використання UML забезпечує можливість формалізованого опису структури та поведінки системи, дозволяє візуалізувати взаємодію користувача з інтерфейсом, описати основні сценарії використання та структуру програмних компонентів. Завдяки цьому покращується розуміння архітектури проєкту серед розробників, дизайнерів та тестувальників.

У рамках проєктування були створені кілька UML-діаграм, які ілюструють ключові функціональні аспекти клієнтської частини системи. Зокрема, діаграма прецедентів [11] дає змогу наочно представити основні сценарії взаємодії користувачів із системою через графічний інтерфейс. Вона також чітко відображає розмежування прав доступу між звичайними користувачами та адміністраторами, що враховується при розробці UI/UX для кожної ролі.

Після успішної реєстрації або автентифікації користувач (див. рис. 3.1) отримує доступ до основного функціоналу вебінтерфейсу. Головне меню поділене на три основні блоки: керування профілем, керування подорожами та участь у подорожах. Кожен із блоків представлений окремим розділом із відповідними формами, таблицями, кнопками та іншими елементами взаємодії.

Функціональність керування профілем включає редагування персональних даних, зміну пароля та видалення облікового запису. Усі зміни супроводжуються формами з клієнтською валідацією та підтвердженням дій.

Організатор може створювати подорожі через покрокову форму, редагувати існуючі маршрути, переглядати наявні групи, надсилати або обробляти запити на участь.

Після приєднання до групи для участі в подорожі користувач отримує доступ до функціоналу, що дозволяє обмінюватися повідомленнями в чаті, переглядати статистику подорожі.

Інтерфейс адміністратора представлений окремим розділом, недоступним звичайним користувачам. Адміністратор має доступ до списку користувачів, подорожей та інструментів для блокування акаунтів або подорожей (див. рис. 3.2). Всі адміністративні дії реалізуються через захищений вебінтерфейс із підвищеним рівнем прав доступу.

На діаграмі прецедентів представлено всі основні сценарії взаємодії користувачів із системою відповідно до їхніх ролей, що дозволяє систематизувати функціональні вимоги та забезпечити коректне проектування логіки доступу на клієнтській стороні.

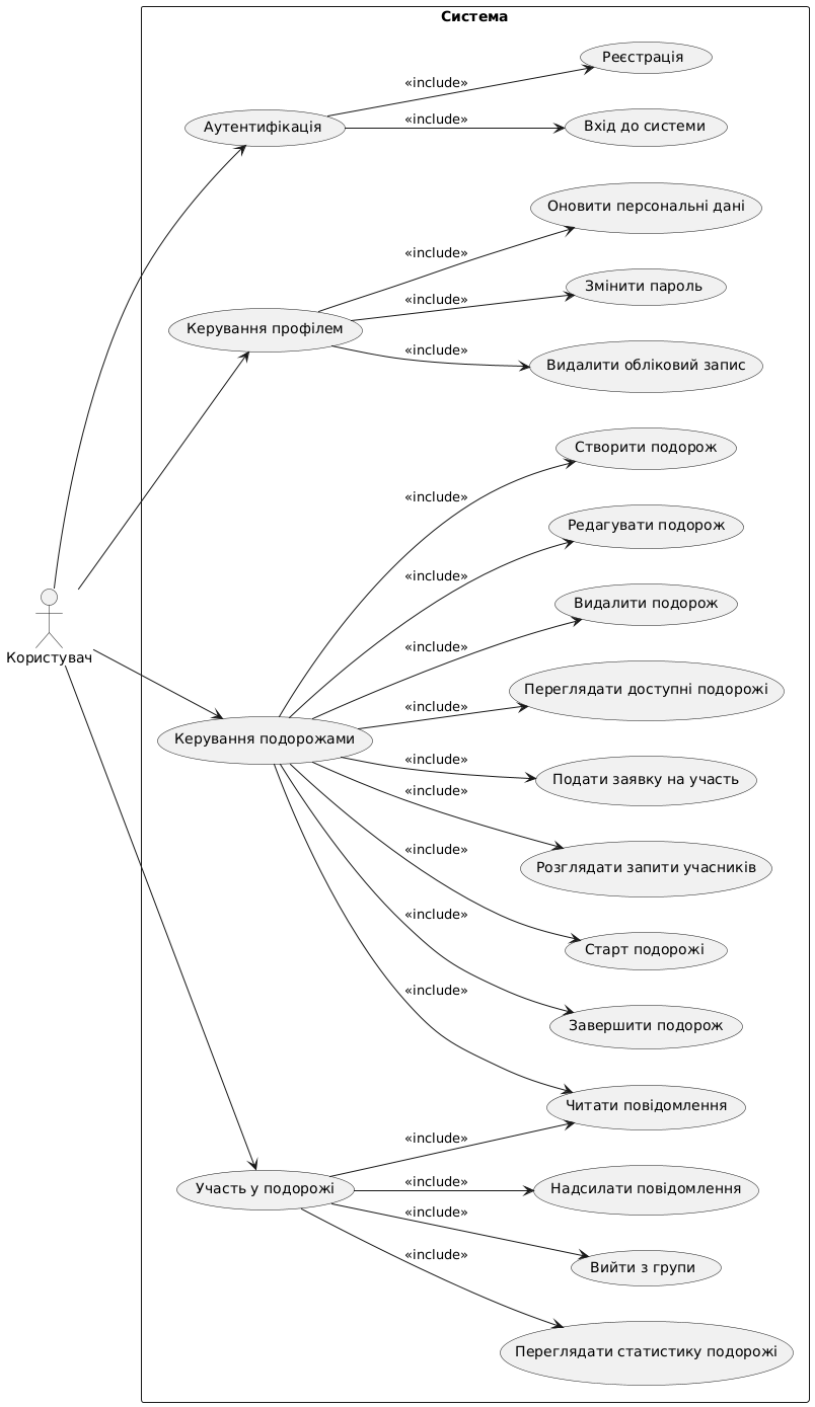

Рисунок 3.1 - Use-case діаграма застосунку «TravelMate» для користувача (рисунок виконаний самостійно)

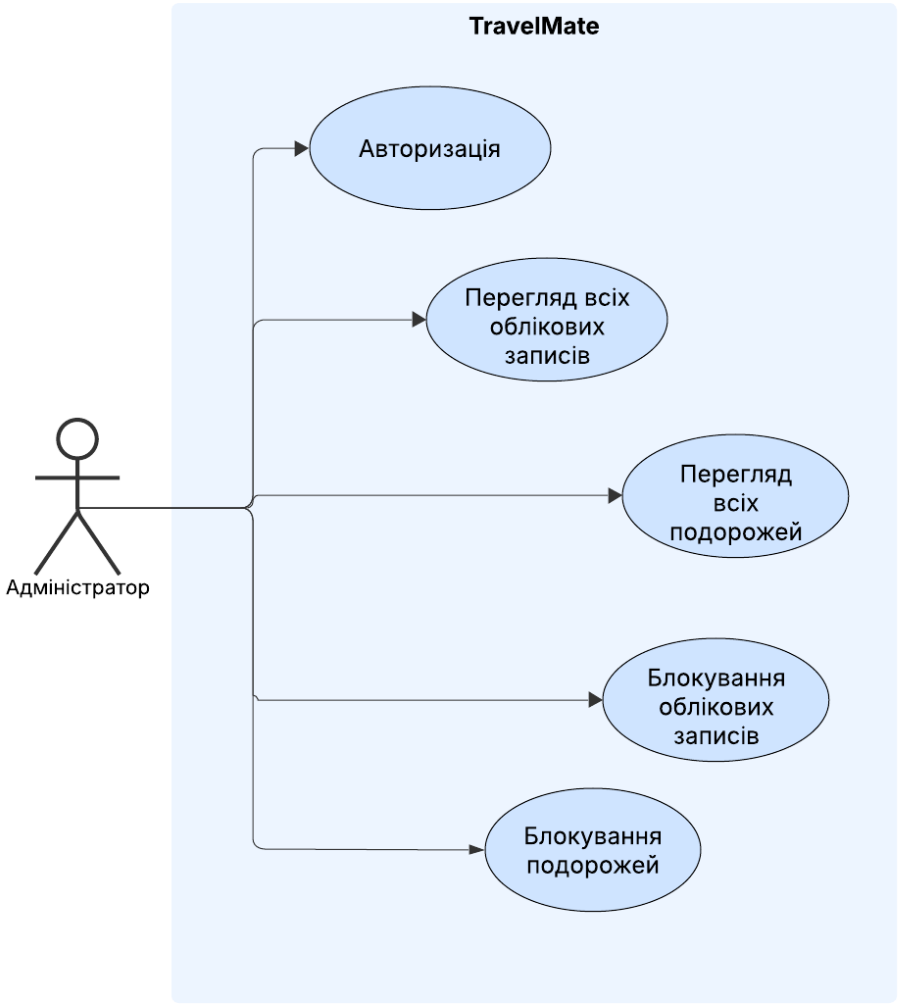

Рисунок 3.2 - Use-case діаграма застосунку «TravelMate» для адміністратора (рисунок виконаний самостійно)

На діаграмі послідовності (див. рис. 3.3) зображено процес створення, підтвердження та запуску подорожі у вебзастосунку TravelMate. Цей сценарій охоплює ключові етапи взаємодії клієнта з сервером.

Користувач надсилає запит авторизації до відповідного контролера. Контролер, у свою чергу, звертається до сервісу автентифікації, який перевіряє облікові дані в базі даних. У разі успішної перевірки користувач отримує JWT токен для подальших запитів.

Після автентифікації користувач надсилає запит на створення подорожі. Контролер подорожей ініціює створення, звертаючись до відповідного сервісу,

який зберігає дані подорожі в базу даних і повертає її унікальний ідентифікатор (ID).

Наступним кроком є підтвердження участі користувачів у подорожі. Контролер надсилає дані до сервісу подорожей, який оновлює статус у базі даних, підтверджуючи участь.

Коли всі учасники підтверджені, користувач надсилає запит на запуск подорожі. Контролер змінює статус подорожі на активний. Сервіс подорожей зберігає оновлений статус у базу даних.

Після запуску подорожі клієнт починає періодично надсилати координати GPS і дані пульсу. Контролер трекінгу передає їх у відповідний сервіс, який записує ці дані до бази даних.

Якщо під час подорожі виявлено критичний стан (наприклад, тривожний пульс або натискання SOS-кнопки), активується альтернативна гілка сценарію. Сервіс SOS ініціює збереження SOS-події та повідомляє організатора подорожі. У нормальному стані клієнт отримує стандартну відповідь 200 OK [12].

Цей сценарій демонструє тісну взаємодію між клієнтською частиною та кількома мікросервісами серверної частини, що відповідають за автентифікацію, збереження подорожей, обробку телеметрії та реагування на надзвичайні ситуації. Такий підхід забезпечує модульність системи, її гнучкість та масштабованість у майбутньому.

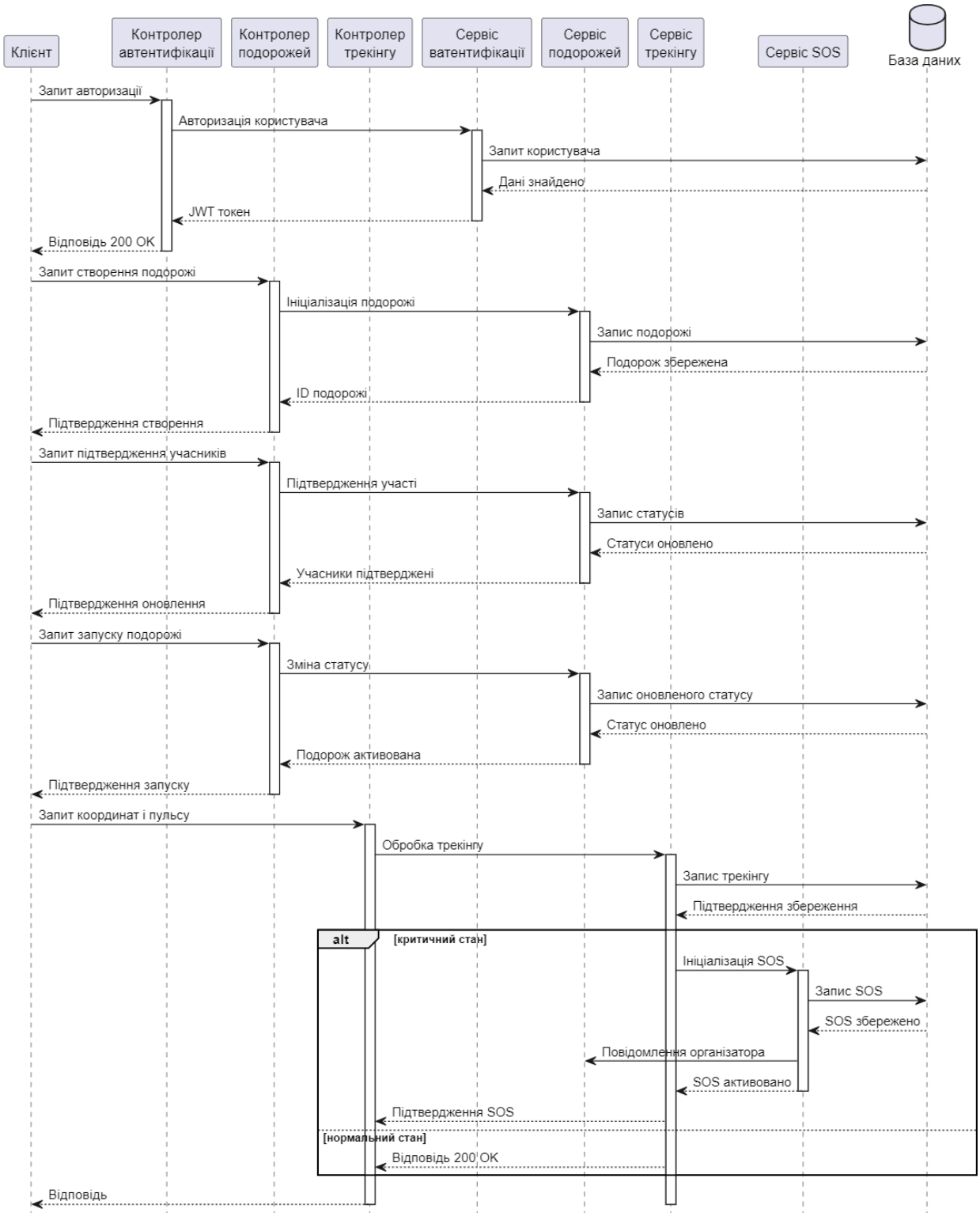

Рисунок 3.3 - Діаграма послідовності сценарію створення та запуску подорожі (рисунок виконаний самостійно)

Для моделювання поведінки користувача в контексті участі у подорожі було розроблено діаграму станів, яка описує можливі стани учасника та переходи між ними протягом життєвого циклу користування продуктом (див. рис. 3.4).

Діаграма включає такі основні стани:

- незареєстрований – початковий стан користувача;
- зареєстрований – користувач, що зареєструвався;
- авторизований – користувач, що ввів валідні ім’я користувача/email та пароль;
- заблокований – користувач не матиме доступу до додатку, тому що адміністратор його заблокував, при цьому користувача можна розблокувати назад;
- не приєднаний – стан користувача, який ще не брав участі в жодній подорожі;
- запрошений – користувач отримав запрошення приєднатися до подорожі;
- очікує підтвердження – після прийняття запрошення учасник очікує підтвердження з боку організатора;
- активний учасник – підтверджений учасник, який бере участь у подорожі;
- виключений – учасник був виключений з подорожі організатором;
- організатор подорожі – користувач створив свою власну подорож;
- завершив участь – подорож завершено, і участь користувача відповідно припинено.

Перехід між станами відбувається відповідно до дій як самого користувача, так і організатора. Наприклад, користувач може прийняти або відхилити запрошення, організатор може підтвердити або відкликати запрошення, а також виключити учасника. У випадку надзвичайної ситуації користувач може ініціювати сигнал SOS, який повертає його до звичайного стану після відповідної обробки.

Розробка цієї діаграми дозволяє чітко формалізувати логіку роботи з учасниками у межах системи, забезпечуючи передбачуваність та контроль над життєвим циклом участі в подорожі.

3.2 Проєктування архітектури програмного забезпечення

Архітектура клієнтської частини вебплатформи TravelMate базується на компонентному підході та реалізована з використанням JavaScript-бібліотеки React. Такий підхід забезпечує модульність, повторне використання коду, легкість масштабування та підтримки проєкту.

Застосунок умовно поділено на кілька основних модулів:

- модуль автентифікації та авторизації – відповідає за процес реєстрації, входу в систему та перевірки прав доступу користувача з використанням JWT-токенів;
- модуль управління подорожами – забезпечує створення, редагування, перегляд та приєднання до подорожей. Передбачає перевірку ролі користувача (мандрівник чи організатор);
- модуль інтерактивної карти – відповідає за візуалізацію маршруту подорожі. Комунікація з сервером здійснюється через REST API;
- модуль обміну повідомленнями (чат) – реалізовано за допомогою довгого опитування для забезпечення швидкого оновлення вхідних і вихідних повідомлень;
- модуль профілю користувача – відповідає за дані користувача, їх зміну;
- модуль навігації та повідомлень – відповідає за маршрутизацію між сторінками, відображення статусів, помилок і підказок користувачу.

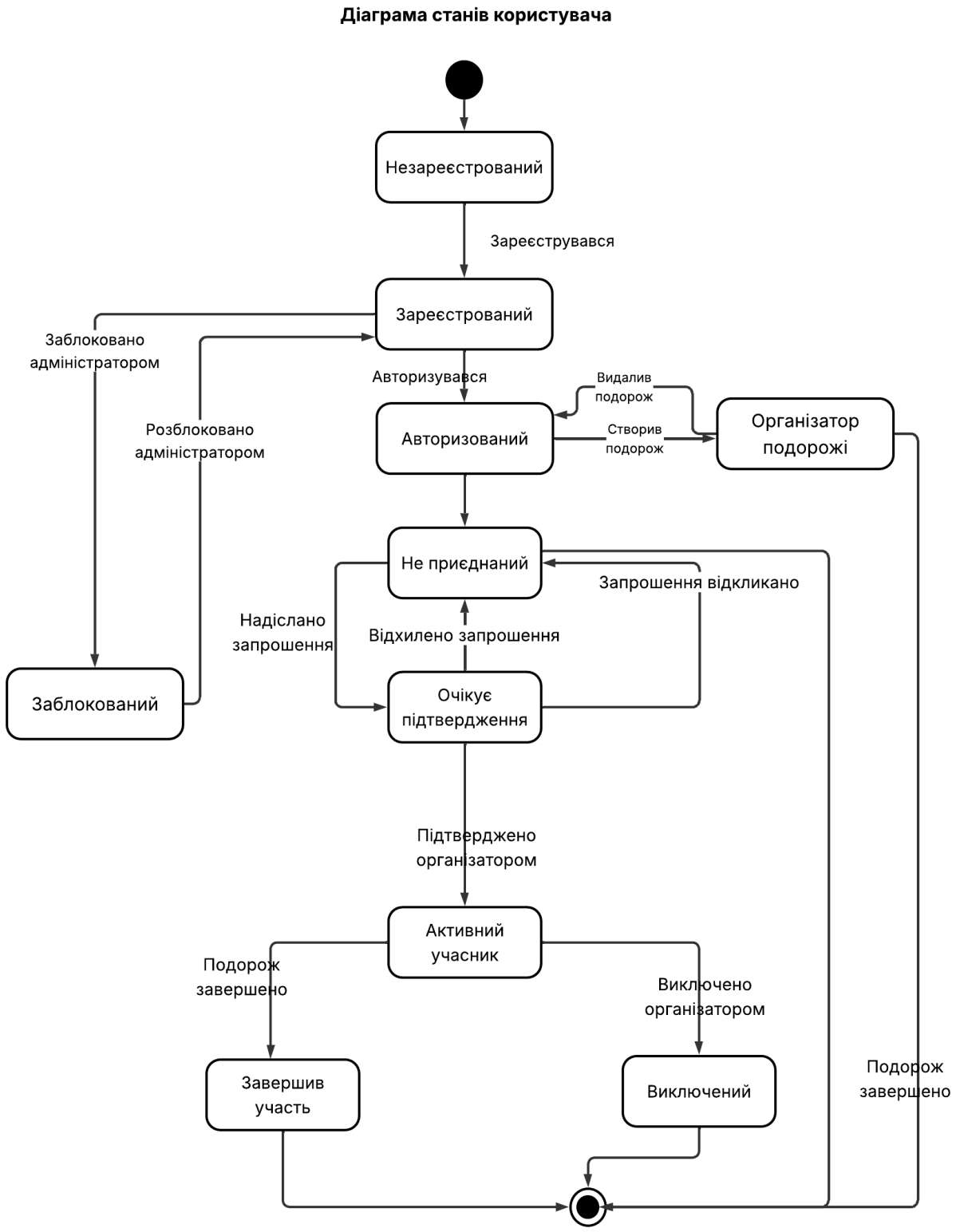

Рисунок 3.4 - Діаграма станів користувача (рисунок виконаний самостійно)

Для формалізації архітектурної структури клієнтського застосунку веб-платформи TravelMate було побудовано діаграму компонентів UML (див. рис. 3.5). Ця діаграма відображає основні функціональні блоки (компоненти), реалізовані у фронтенді, а також показує, як ці блоки взаємодіють один з одним та із зовнішніми сервісами (REST API).

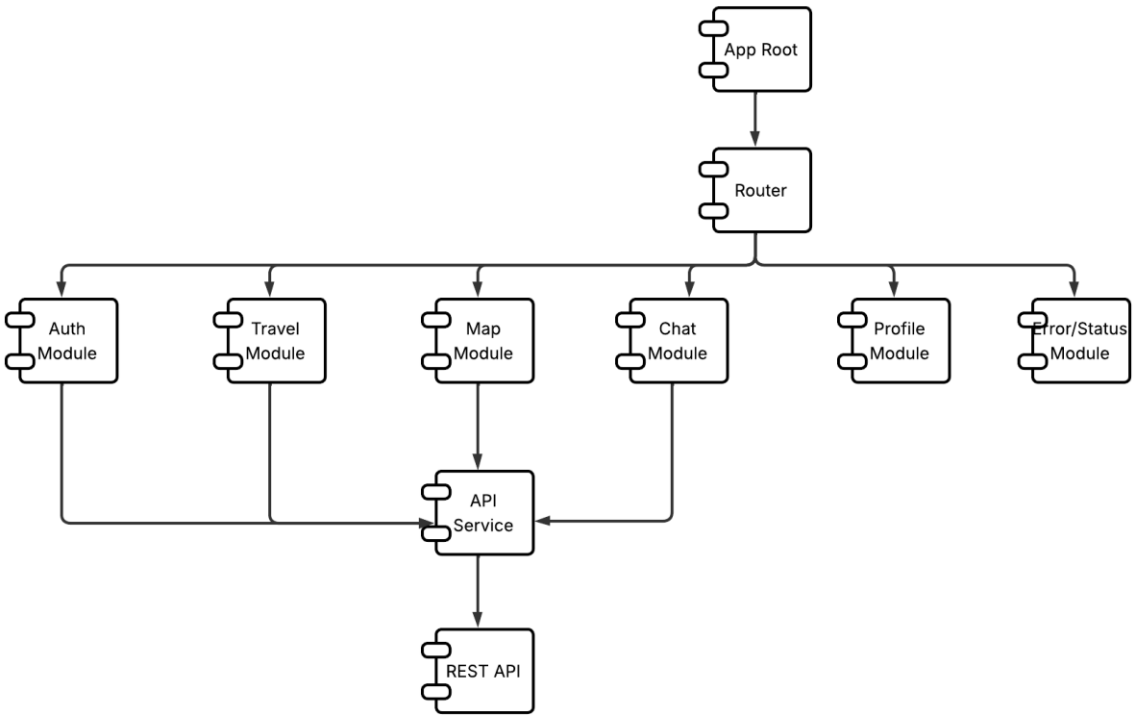

Рисунок 3.5 - Діаграма компонентів (рисунок виконаний самостійно)

Архітектура побудована за принципом SPA. Для організації логіки маршрутизації використовується бібліотека react-router-dom, а для обміну даними з API - Axios. Обробка глобального стану (наприклад, даних користувача, статусу сесії) реалізована через Context API.

Для забезпечення гнучкості та масштабованості проєкту, структура папок застосунку відповідає доменній моделі - логіка поділена за функціональними доменами (components, admin, ui, pages, hooks тощо). Кожен домен містить власні компоненти, сервіси, хуки та стилі.

Застосовано Lazy Loading для оптимізації продуктивності - неактивні частини інтерфейсу завантажуються динамічно при потребі.

Таким чином, обрана архітектура забезпечує ефективну підтримку функціональних вимог системи, високий рівень зручності для користувачів, а також можливість подальшого розвитку проєкту.

3.3 Проєктування структури зберігання даних

Процес створення бази даних для програмної системи TravelMate охоплює кілька ключових етапів: концептуальне, логічне й фізичне моделювання. Його метою є формування послідовної, ефективної та узгодженої структури даних, що забезпечує повноцінне функціонування як клієнтської, так і серверної частин програми.

На етапі концептуального моделювання були ідентифіковані основні сутності, що репрезентують об’єкти реального світу: користувачі, подорожі в групах, повідомлення, сигнали тривоги, дані трекінгу, маршрутні точки, журнали активності тощо. Між цими сутностями встановлені зв’язки, що відображають логіку взаємодії між учасниками в рамках подорожей.

Модель включає такі аспекти:

- один користувач може створити кілька подорожей;
- користувач має можливість брати участь у багатьох подорожах (зв’язок типу «багато до багатьох»);
- зв’язок реалізується через проміжну сутність «Участь», що зберігає дані про участь користувача в подорожі;
- маршрут визначається послідовністю точок, упорядкованих за певним алгоритмом;
- з кожною участю пов’язані повідомлення.

На базі концептуальної схеми сформовано логічну модель, у якій описано структуру бази даних у вигляді сутностей з атрибутами та зв’язками між ними. Основні сутності:

- user (користувач) - містить ID, ім’я, email, хеш пароля, роль, дату народження, дату реєстрації;
- groupTravel (групова подорож) - включає ID подорожі, назву, опис маршруту, дати початку й завершення, регіон, максимальну кількість учасників;
- participation (участь) - містить ID участі, посилання на користувача й подорож, статус заявки, прапорець організатора, дату приєднання;
- routePoint (маршрутна точка) - визначається ID точки, подорожжю, координатами, описом і порядком у маршруті;
- message (повідомлення) - містить ID, ID участі, текст повідомлення та мітку часу;
- trackingData (дані трекінгу) - включає ID, участь, координати, пульс, час отримання, позначку про аномалію;
- alertSignal (сигнал тривоги) - зберігає ID, участь, координати, статус опрацювання та дату події.

На фізичному рівні структура реалізована як реляційна база даних з чітким визначенням типів даних, первинних (PK) і зовнішніх ключів (FK). Всі атрибути нормалізовані згідно з третьою нормальною формою (3НФ), що включає:

- 1НФ - усунення повторюваних груп і складених структур;
- 2НФ - залежність неключових атрибутів лише від повного первинного ключа;
- 3НФ - виключення транзитивних залежностей.

Створена модель бази даних забезпечує цілісність, масштабованість і адаптивність системи TravelMate. Вона підтримує ключові сценарії роботи, включаючи обмін повідомленнями, передачу координат і біометричних даних у

реальному часі, а також реагування на надзвичайні ситуації. Графічне зображення структури бази даних представлено на рисунку 3.6.

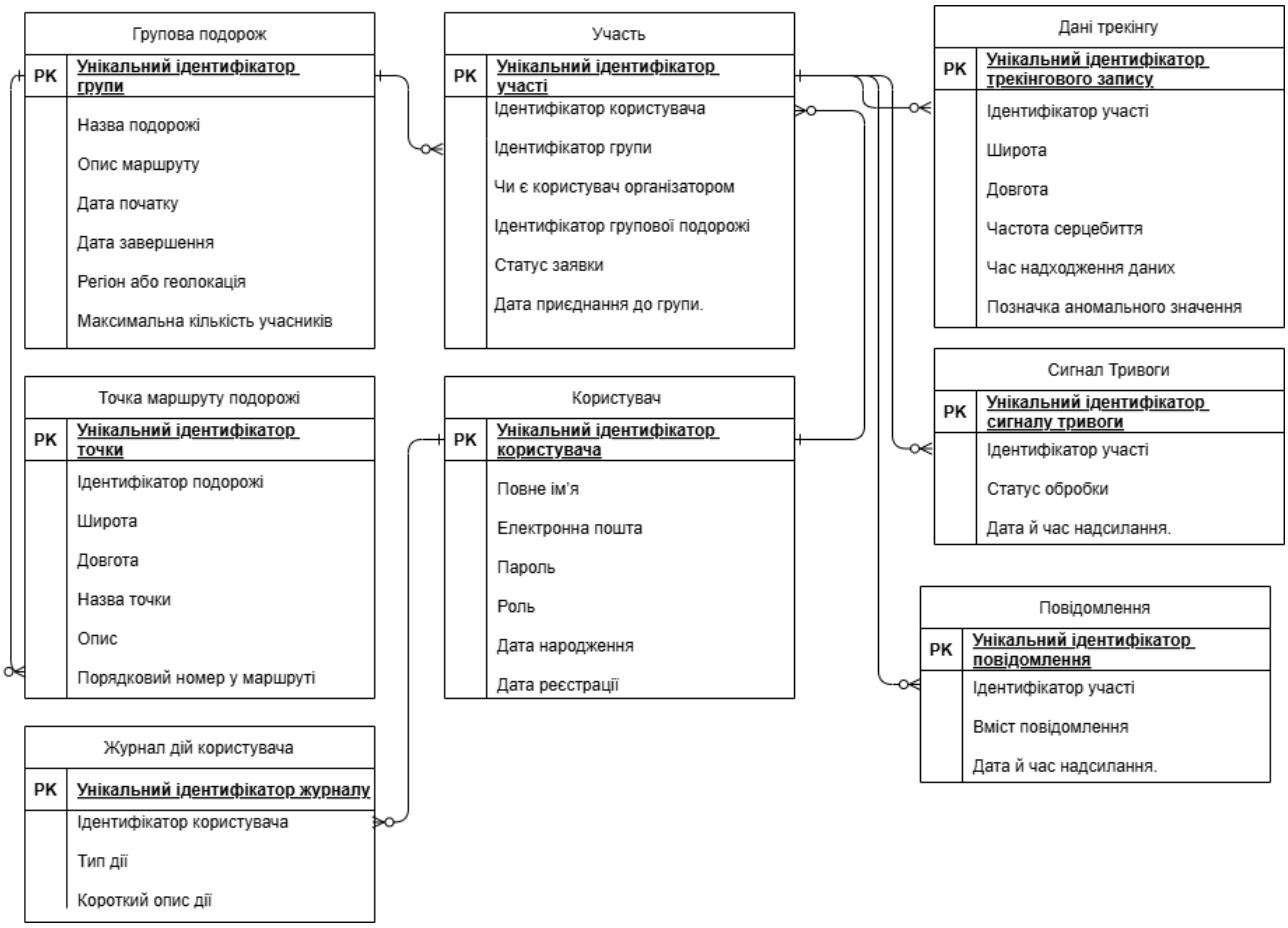

Рисунок 3.6 - Структура бази даних (рисунок виконаний самостійно)

3.4 Створення UI / UX

У контексті розробки TravelMate особливу увагу було приділено створенню зручного, інтуїтивного та естетично привабливого інтерфейсу користувача, адже саме він формує перше враження та безпосередньо впливає на залучення і утримання користувачів. Ефективність взаємодії між користувачем і системою, а також візуальна зрозумілість функціоналу мають вирішальне значення.

При проєктуванні UI/UX використовувалися такі ключові принципи:

- зрозумілість - кожен елемент інтерфейсу повинен мати очевидне призначення;
- простота навігації - інтерфейс поділений на логічні блоки (дашборд, подорожі, повідомлення, профіль), які легко знаходити та між якими зручно перемикатися;
- мінімалізм - дизайн не перевантажено зайвими елементами, акцент зроблено на функціональність.

Для візуального оформлення інтерфейсу обрана світла тема з акцентами у фіолетових і пурпурових тонах, що асоціюються з надійністю та творчістю [13].

Центральним елементом інтерфейсу є дашборд користувача (див. рис. 3.7), який надає швидкий доступ до основних функцій:

- огляд всіх подорожей;
- перегляд поточного статусу подорожі;
- можливість виходу с акаунту;
- профільні налаштування.

Кожен блок має власну панель з чітко вираженим заголовком, іконками та взаємодіючими кнопками.

Для кожної подорожі передбачена окрема сторінка з детальною інформацією (див. рис. 3.8):

- назва, дати та опис подорожі;
- максимальна кількість учасників;
- організатор і статус;
- екстремальність;
- вбудований чат для комунікації;
- кнопки для організатора, що ведуть на сторінки редагування або забезпечують видалення подорожі;
- кнопки для учасників для подання запиту на приєднання.

Інтерфейс адаптується залежно від ролі користувача. Звичайний користувач бачитиме лише ті функції, які стосуються участі в подорожах:

перегляд маршрутів, чат, сповіщення. Організатор подорожей додатково буде мати доступ до функцій редагування та завершення подорожей, а також управління учасниками.

Логічно побудований інтерфейс відповідає потребам обох основних типів користувачів.

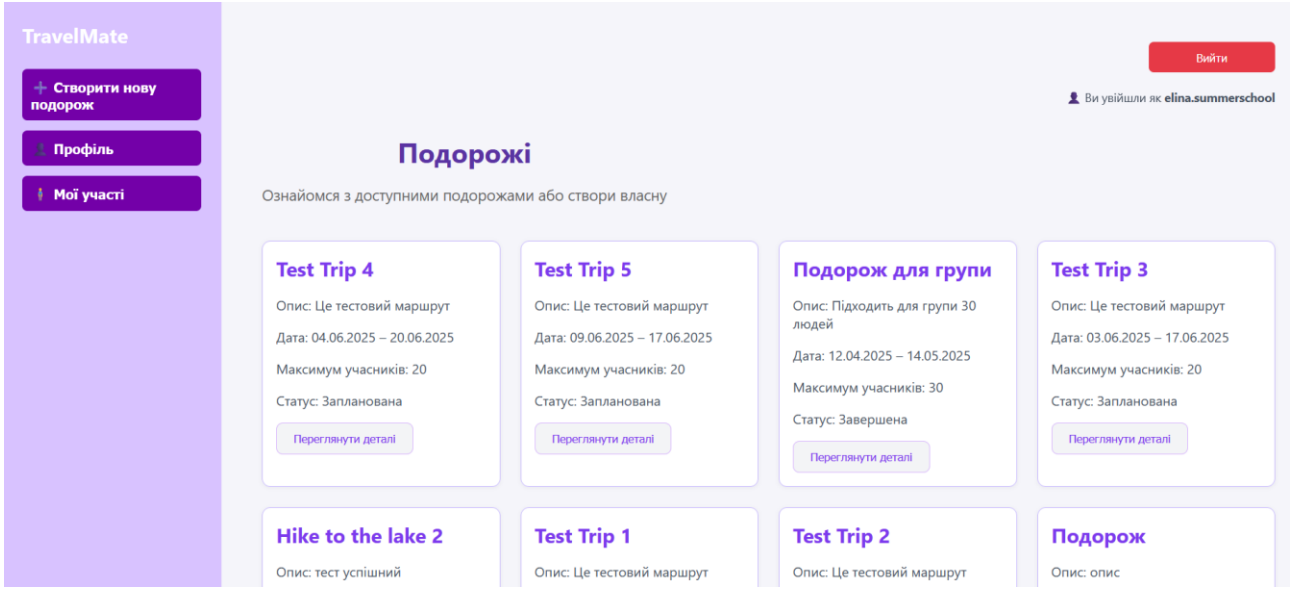

Рисунок 4.7 - Головна сторінка користувача (рисунок виконаний самостійно)

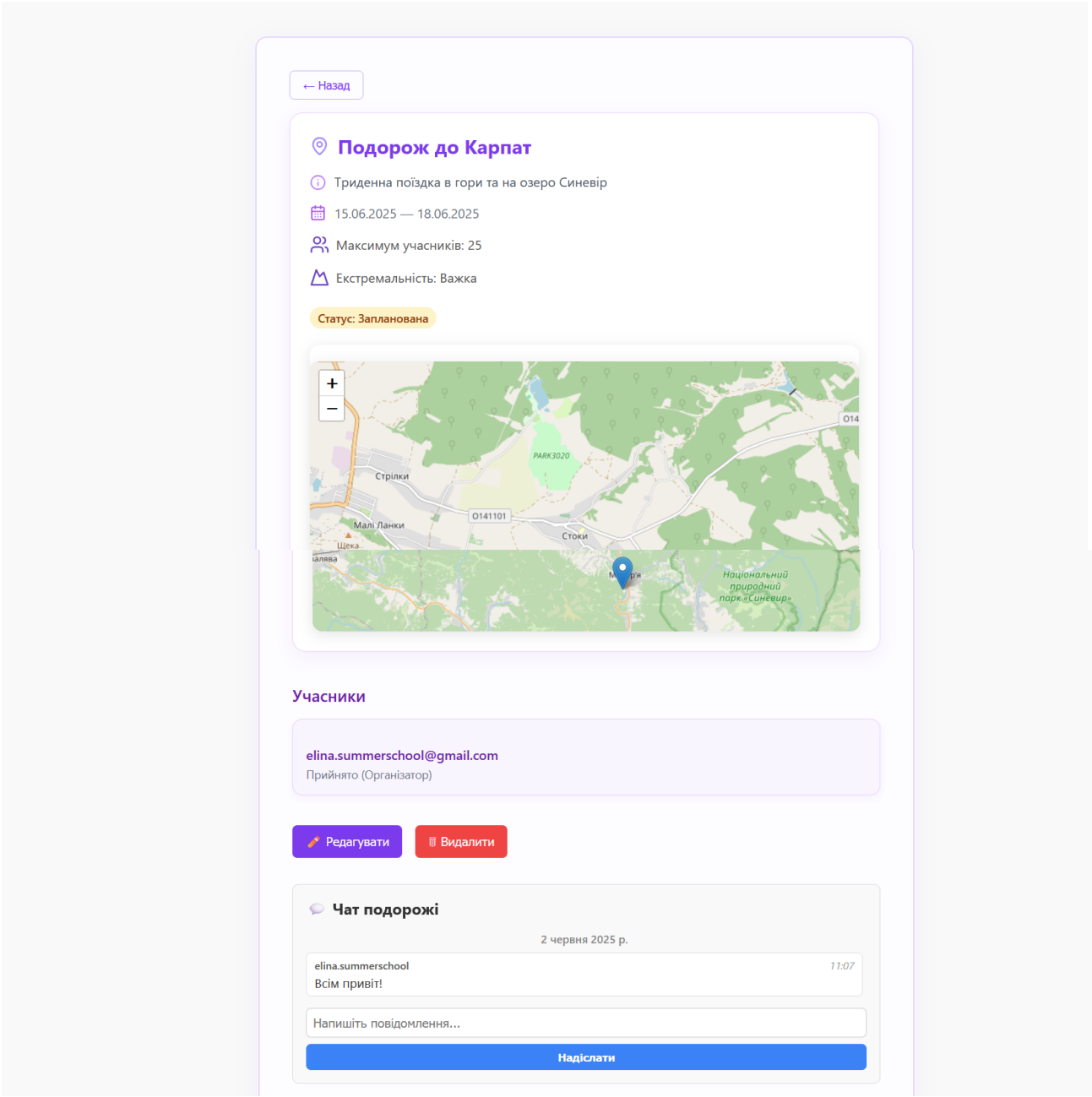

Рисунок 4.8 - Розділ “Деталі подорожі” (рисунок виконаний самостійно)

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Вибір технологій для front-end

У процесі розробки клієнтської частини програмної системи TravelMate - вебзастосунку для планування групових подорожей – було обрано сучасний технологічний стек, що забезпечує гнучкість, масштабованість і високу продуктивність інтерфейсу. До складу обраного стеку увійшли:

- мова програмування JavaScript;
- бібліотека React.js;
- бібліотека для HTTP-запитів Axios;
- система маршрутизації React Router;
- контекст API для керування автентифікацією (AuthContext);
- стилізація через CSS-модулі.

JavaScript – динамічна, інтерпретована мова програмування, що є основою для розробки клієнтської частини вебзастосунків. Її широке поширення, сумісність з усіма сучасними браузерами, активна спільнота та гнучка синтаксична структура роблять її основним інструментом для побудови інтерфейсів.

Серед переваг використання JavaScript:

- інтерактивність інтерфейсу в реальному часі;
- підтримка великої кількості бібліотек та фреймворків;
- повна інтеграція з HTML/CSS;
- розвинений інструментарій розробника.

React.js – популярна JavaScript-бібліотека [14] для створення інтерфейсів користувача на основі компонентного підходу. Вона надає можливість побудови динамічних SPA (Single Page Application), де оновлення сторінки відбувається без повного перезавантаження.

Основними перевагами React є:

- віртуальний DOM, що оптимізує перерендеринг;
- JSX – декларативний синтаксис для шаблонів;
- висока повторюваність компонентів;
- активна спільнота та широке розширення екосистеми.

React Router [15] використовується для реалізації маршрутизації в межах SPA. Завдяки цій бібліотеці реалізовано логіку переходів між сторінками (авторизація, список подорожей, профіль користувача тощо) без повного перезавантаження сторінки.

Перевагами цієї бібліотеки є:

- вкладені маршрути;
- маршрути з динамічними параметрами (наприклад, ID подорожі);
- гнучке перенаправлення та захищені маршрути (PrivateRoutes);
- зручна інтеграція з хуками React (useNavigate, useParams, useLocation).

Axios – бібліотека для виконання HTTP-запитів до backend API. У застосунку TravelMate вона використовується для взаємодії з серверною частиною та обробки запитів до бази даних, що включають:

- отримання списку подорожей;
- реєстрацію та автентифікацію користувача;
- відправлення запитів на приєднання до подорожі тощо.

Axios підтримує глобальну обробку помилок через interceptors, передачу токена авторизації, асинхронне виконання з використанням async/await.

Також у роботі використано Context API (AuthContext) – вбудований інструмент React для створення глобального контексту стану. У цьому застосунку використовується для керування процесом автентифікації: збереження імені користувача, токена авторизації, статусу адміністратора тощо. Такий підхід дозволив уникнути додаткових бібліотек для керування станом, зберігаючи простоту реалізації.

4.2 Архітектурні рішення

Проєктування архітектури програмного забезпечення є ключовим етапом, який визначає подальший підхід до розробки, масштабованість та зручність підтримки системи. У ході реалізації клієнтської частини застосунку для організації групових подорожей було прийнято низку архітектурних рішень, спрямованих на забезпечення зручності використання, гнучкості та ефективності.

По перше, було обрано архітектуру Single Page Application (SPA), яка передбачає завантаження єдиної HTML-сторінки з динамічним оновленням її вмісту у браузері без повного перезавантаження сторінки при переході між розділами. Це дозволяє створити більш плавний і швидкий користувацький досвід, схожий на роботу настільних додатків.

Серед основних переваг SPA:

- швидша взаємодія з інтерфейсом, завдяки зменшенню кількості запитів до сервера;
- менше навантаження на мережу, оскільки більшість ресурсів завантажується один раз;
- покращене UX, завдяки швидкому переходу між сторінками;
- можливість створення офлайн-функціоналу при використанні відповідних механізмів зберігання даних у браузері.

Саме ці переваги стали вирішальними для побудови веб-застосунку.

Клієнтська частина застосунку реалізована відповідно до принципів компонентної архітектури, що передбачає поділ інтерфейсу на незалежні, ізольовані та повторно використовувані частини – компоненти. Це дозволяє підвищити гнучкість і повторне використання коду, забезпечити кращу підтримку і масштабування застосунку, а також реалізувати чітку структуру проєкту із поділом логіки за функціональними областями.

Компоненти поділено на сторінки (pages), які відповідають окремим маршрутам (наприклад, /login, /admin, /profile), та використовувані компоненти (components), що включають навігаційні панелі, форми, таблиці тощо.

Для управління переходами між сторінками використано React Router. У застосунку реалізовані захищені маршрути (PrivateRoute), доступ до яких обмежено лише для авторизованих користувачів або адміністраторів. Також є, розділення доступу за ролями (звичайний користувач або адміністратор) та обробка навігації через параметри URL, що дозволяє реалізувати динамічні маршрути.

Для зберігання даних про стан користувача (ім’я, токен, роль користувача) застосовується React Context API. Це рішення дозволяє уникнути використання сторонніх бібліотек типу Redux та централізовано зберігати дані про авторизацію.

Стан зберігається у AuthContext, який реалізує логіку логіну, реєстрації, перевірки ролі користувача та обробки виходу із системи.

Крім цього, у застосунку передбачено універсальні компоненти, які використовуються як для читання даних, так і для взаємодії з API (наприклад, форми для створення, редагування та перегляду подорожі, керування користувачами).

Однією з ключових функцій застосунку є можливість створення нової подорожі. Для цього був реалізований компонент CreateTripPage, який дозволяє користувачеві заповнити форму, що відправляє дані на сервер через API. Нижче наведено фрагмент коду компонента:

```
function CreateTripPage() {
  const navigate = useNavigate();
  const [routePoints, setRoutePoints] = useState([]);

  const formik = useFormik({
    initialValues: {
      title: '',
      description: '',
      startTime: '',
      endTime: '',
```

```
            maxParticipants: '',
            difficulty: 0,
          },
          validationSchema: Yup.object({
            title: Yup.string().required('Назва обов’язкова'),
            startTime: Yup.date().required('Початкова дата обов’язкова'),
            endTime: Yup.date()
                 .min(Yup.ref('startTime'), 'Кінцева дата повинна бути пізнішою за початкову')
              .required('Кінцева дата обов’язкова'),
            maxParticipants: Yup.number()
              .required('Максимальна кількість учасників обов’язкова')
              .min(1, 'Має бути хоча б один учасник'),
          }),
          onSubmit: async (values) => {
            try {
              const payload = {
                ...values,
                maxParticipants: parseInt(values.maxParticipants),
                difficulty: parseInt(values.difficulty),
                startTime: new Date(values.startTime).toISOString(),
                endTime: new Date(values.endTime).toISOString(),
                routePoints: routePoints.map((point, idx) => ({
                  latitude: point.lat,
                  longitude: point.lng,
                  name: point.name,
                  description: point.description,
                  order: parseInt(point.order),
                })),
              };

              await axiosInstance.post('/travel-groups', payload);
              alert('Подорож створено!');
              navigate('/');
            } catch (error) {
              console.error('Помилка створення подорожі:', error);
              alert('Помилка при створенні подорожі');
            }
          },
        });
```

Компонент CreateTripPage реалізує сторінку з формою для створення подорожі (див. рис. 4.1). Для управління формою використовується бібліотека Formik, що забезпечує контроль над станом введених даних та їх валідацією.

Валідація полів виконується за допомогою бібліотеки Yup: назва та дати початку/завершення - обов'язкові, є перевірка, щоб кінцева дата не була раніше за початкову, також максимальна кількість учасників має бути числом більше або рівним 1. Після успішного заповнення та підтвердження форми формується об'єкт payload, що містить всі дані подорожі , інформація про точки маршруту

трансформується у формат, очікуваний API і здійснюється відправка запиту через axiosInstance.post() на endpoint /travel-groups. Користувач повідомлений про успішне створення за допомогою вспливаючого повідомлення (див.рис. 4.2).

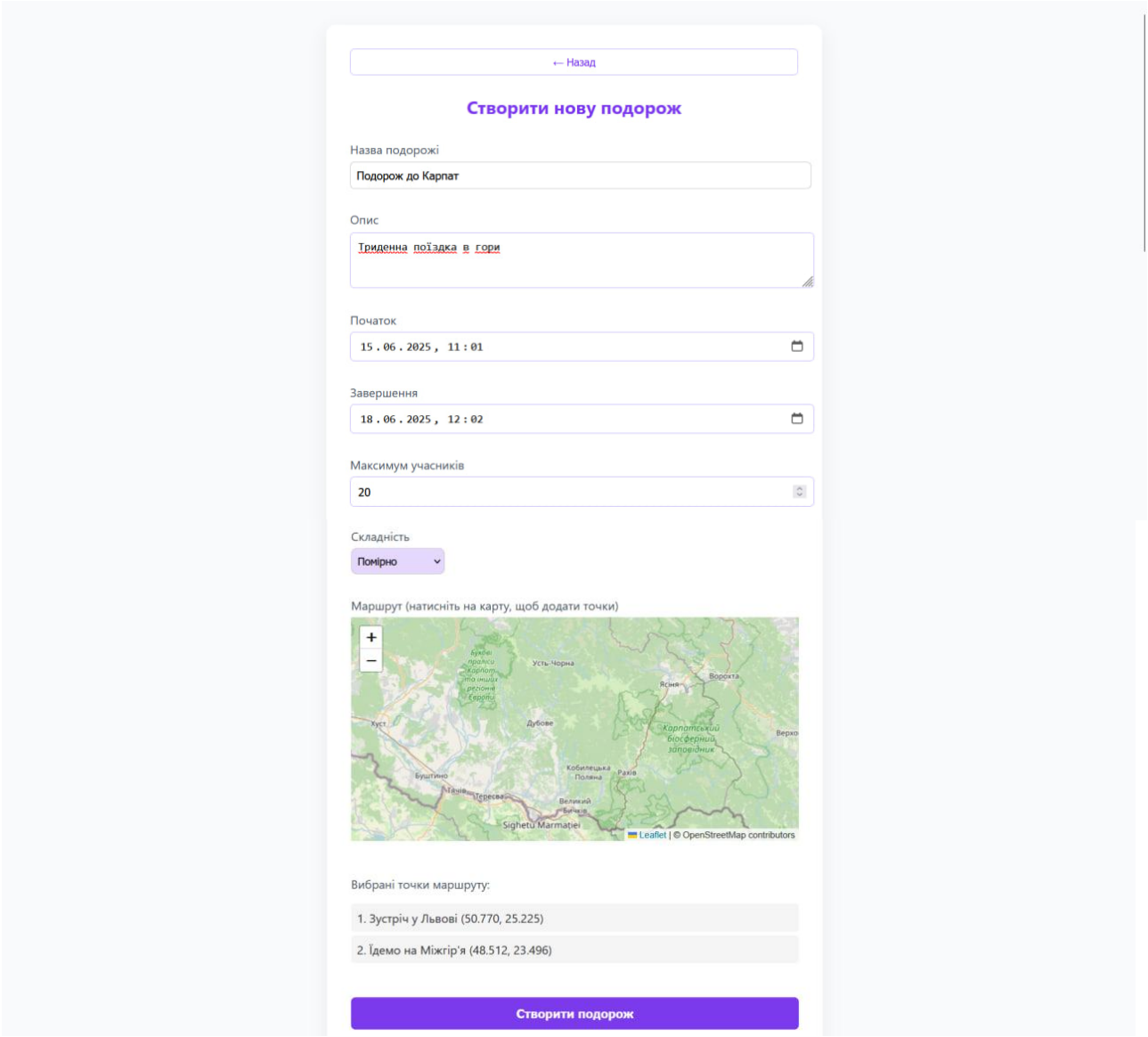

Рисунок 4.1 – Створення подорожі (рисунок виконаний самостійно)

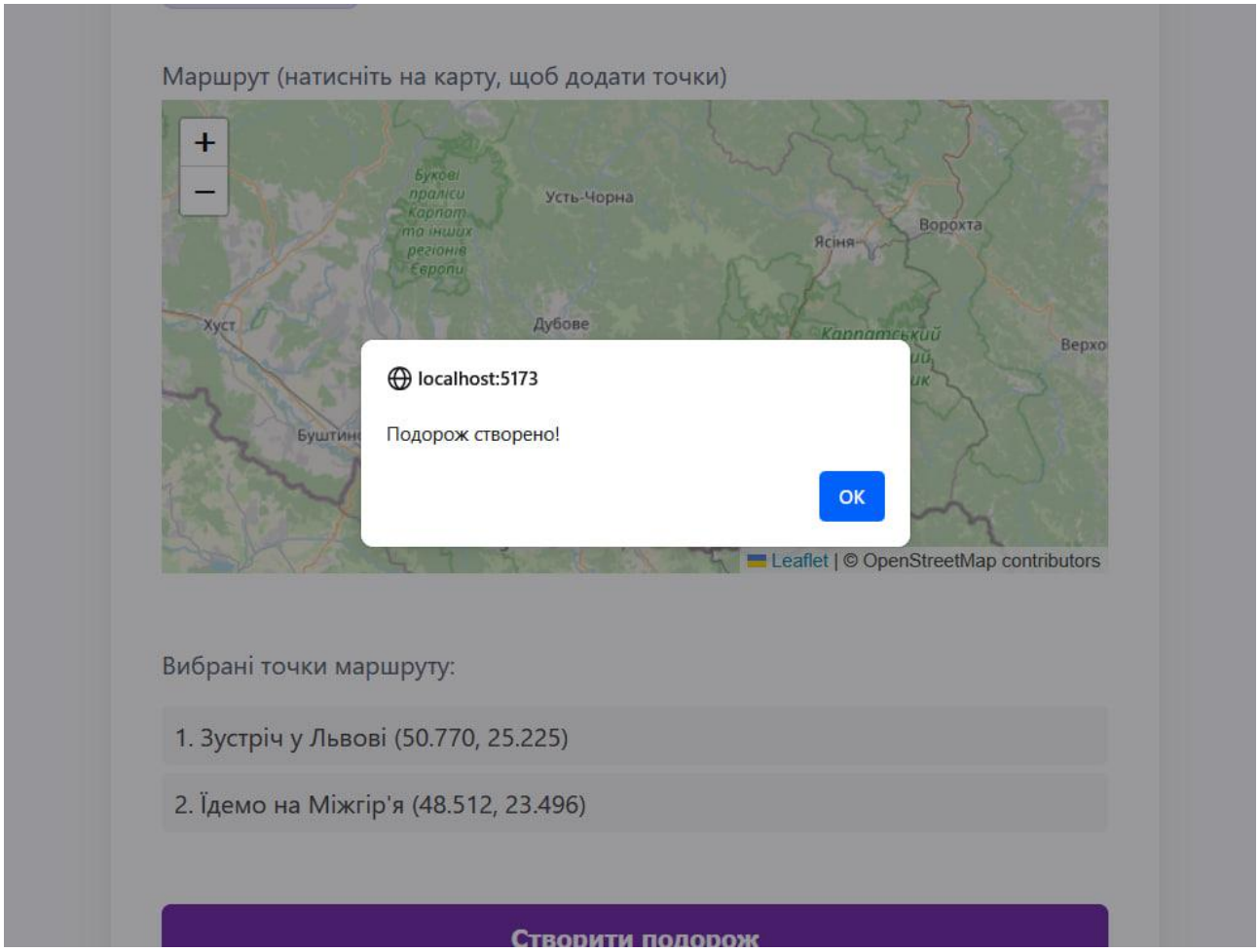

Рисунок 4.2 – Повідомлення про успішне створення подорожі (рисунок виконаний самостійно)

4.3 Інтеграція з Back-end

Інтеграція клієнтської частини застосунку з серверною частиною є важливим етапом, що забезпечує взаємодію користувача з бізнес-логікою та базою даних через інтерфейс. У процесі розробки було реалізовано повноцінну інтеграцію з back-end, що надається через REST API.

Серверна частина проєкту реалізована з використанням ASP.NET, і надає доступ до функціональності через REST API. Обмін даними між front-end та back-end відбувається у форматі JSON за допомогою HTTP-запитів (методи GET, POST, PUT, DELETE).

Для зручності взаємодії з API у проєкті використовується бібліотека axios, яка дозволяє створювати HTTP-запити та конфігурувати їх на глобальному рівні. Було створено окремий екземпляр axios з базовою URL-адресою API та механізмом автоматичної передачі токену авторизації у кожному запиті:

```
import axios from 'axios'
const instance = axios.create({
  baseURL: 'http://localhost:8080/api',
})
instance.interceptors.request.use(
  (config) => {
    const token = localStorage.getItem('token')
    if (token) {
      config.headers.Authorization = `Bearer ${token}`
    }
    return config
  },
  (error) => Promise.reject(error)
)

export default instance
```

Тут використовується метод axios.create() для створення окремого екземпляру з базовою адресою API. За допомогою interceptors.request.use() перехоплюються всі запити перед відправленням. З локального сховища (localStorage) витягується токен авторизації. При цьому, якщо токен присутній, він додається до заголовків запиту у форматі Bearer Token. Таким чином, не потрібно додавати токен вручну при кожному запиті - це відбувається автоматизовано.

Цей підхід дозволяє централізовано контролювати всі запити до сервера, забезпечує безпечну передачу даних та спрощує підтримку коду.

У процесі розробки були реалізовані такі ключові інтеграції між front-end та API back-end:

- авторизація та автентифікація користувачів;
- обробка подорожей (створення, редагування, видалення);
- обробка заявок (підтвердити або відхилити заявку);
- перегляд і редагування профілю користувача;

- керування маршрутом і групами;
- отримання статистики закінченої подорожі;
- блокування користувачів або груп (тільки для адміністратора).

З метою безпеки було реалізовано перевірку авторизаційного токену, захист маршрутів через логіку на front-end, автоматичне перенаправлення на сторінку логіну при втраті доступу.

5. ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Після реалізації функціоналу вебзастосунку важливим етапом є перевірка його коректності, надійності та відповідності вимогам. У процесі тестування клієнтської частини програмного продукту були застосовані як ручні, так і автоматизовані підходи.

5.1 Ручне тестування

Основний обсяг тестування на початкових етапах був здійснений вручну. Цей підхід дозволяє швидко перевірити ключову функціональність, особливо під час активної розробки. Ручне тестування охоплювало такі сценарії:

- функціональне тестування: перевірка правильності роботи основних функцій застосунку: авторизації, створення подорожі, перегляду списку подорожей, навігації між сторінками;
- тестування валідації форм: перевірка, чи коректно обробляються помилки користувача (наприклад, порожні поля або некоректні дати);
- тестування взаємодії з API: контроль відповідей сервера при запитах створення, отримання та оновлення даних.

Тестування проводилося в сучасних браузерах (Google Chrome, Mozilla Firefox).

5.2 Валідаційне тестування форм

У більшості форм проєкту (реєстрації, авторизації, створення подорожі) була реалізована валідація введених даних за допомогою бібліотеки Yup у

поєднанні з Formik. Це дозволило реалізувати перевірку введених користувачем значень ще до надсилання запиту на сервер, що зменшує навантаження на бекенд та покращує UX.

Основні перевірки, що проводилися:

- правильний формат електронної скриньки;
- мінімальна довжина пароля;
- збіг паролів у полі підтвердження;
- заповнення обов’язкових полів;
- коректність дат для перевірки, що дата завершення подорожі пізніше за дату початку;
- числові обмеження для кількості учасників.

Наприклад, якщо користувач не заповнив поле “Ім’я” або ввів email у некоректному форматі, форма не буде надіслана, а помилка відобразиться поруч із відповідним полем. Це є результатом успішного проходження тестів валідації.

5.3 Тестування взаємодії з API

Для перевірки коректної інтеграції з бекендом, у багатьох компонентах, наприклад, для реєстрації, входу в акаунт, створення та редагування подорожі, проводилося тестування обробки API-запитів до відповідних ендпоінтів. Було протестовано коректне формування запитів (структура тіла, формати дат, перетворення типів даних), обробка позитивного результату (збереження токена в localStorage, перенаправлення користувача та очищення форм) та обробка помилок (виведення повідомлення).

Використання бібліотеки axios та обгортки axiosInstance дозволило централізовано обробляти запити та помилки. Усі основні сценарії були перевірені вручну: введення неправильних облікових даних, спроба створити подорож без обов'язкових полів, некоректний формат дат тощо.

ВИСНОВКИ

У результаті виконання цієї роботи було реалізовано клієнтську частину вебзастосунку TravelMate, що призначений для організації та супроводу групових подорожей. Було проведено аналіз предметної галузі, виявлено основні проблеми в координації учасників подорожей, зокрема – труднощі з комунікацією, навігацією, безпекою та контролем фізичного стану учасників.

На основі аналізу було сформульовано вимоги до функціоналу застосунку, розроблено його архітектуру та реалізовано інтерфейс користувача. Особливу увагу було приділено зручності використання (UX/UI), безпеці.

У ході роботи були враховані сучасні технології фронтенд-розробки: React, REST API, JWT-автентифікація, односторінкова архітектура (SPA). Завдяки цьому вдалося створити інтуїтивно зрозумілий, швидкий та функціональний інтерфейс для туристичних груп.

Отримані результати свідчать про ефективність такого рішення та його потенціал до масштабування й подальшого впровадження у реальні туристичні сервіси. У майбутньому можливим є розширення функціональності, зокрема додавання голосового сповіщення, офлайн-режиму, рекомендацій маршрутів на основі вподобань користувачів та інтеграції з зовнішніми картографічними сервісами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Travel Software Market Growing at a CAGR of 10.65% From 2024-2032 | Expedia Group, Travelport Worldwide, Amadeus IT Group SA [Електронний ресурс] – URL: https://www.globenewswire.com/news-release/2025/02/19/3028595/0/en/Travel-Software-Market-Growing-at-a-CAGR-of-10-65-From-2024-2032-Expedia-Group-Travelport-Worldwide-Amadeus-IT-Group-SA.html (дата звернення: 28.05.2025).
2. A Survey on Tourist Trip Planning Systems [Електронний ресурс] – URL: https://www.researchgate.net/publication/259823385_A_SURVEY_ON_TOURIST_TRIP_PLANNING_SYSTEMS (дата звернення: 28.05.2025).
3. Systematic review of mobile travel apps and their smart features and challenges [Електронний ресурс] – URL: https://www.researchgate.net/publication/365839591_Systematic_review_of_mobile_travel_apps_and_their_smart_features_and_challenges (дата звернення: 29.05.2025).
4. Designing a Trip Planner Application for Groups: Exploring Group Tourists? Trip Planning Requirements [Електронний ресурс] – URL: https://www.researchgate.net/publication/302074045_Designing_a_Trip_Planner_Application_for_Groups_Exploring_Group_Tourists_Trip_Planning_Requirements (дата звернення: 29.05.2025).
5. Tourism 4.0 [Електронний ресурс] – URL: https://en.wikipedia.org/wiki/Tourism_4.0?utm_source=chatgpt.com
6. Travefy [Електронний ресурс] – URL: https://travefy.com/ (дата звернення: 29.05.2025).
7. Wanderlog [Електронний ресурс] – URL: https://wanderlog.com/home (дата звернення: 29.05.2025).
8. Let's Jetty [Електронний ресурс] – URL: https://www.letsjetty.com/ (дата звернення: 29.05.2025).

9. JSON Web token (JWT) [Електронний ресурс] – URL: https://datatracker.ietf.org/doc/html/rfc7519 (дата звернення: 29.05.2025).
10. Software Architecture: The Ultimate Guide [Електронний ресурс] – URL: https://tecnovy.com/en/software-architecture-ultimate-guide (дата звернення: 29.05.2025).
11. Use Case Diagram - Unified Modeling Language (UML) Guide [Електронний ресурс] – URL: https://www.geeksforgeeks.org/use-case-diagram/ (дата звернення: 29.05.2025).
12. List of HTTP status codes [Електронний ресурс] – URL: https://en.wikipedia.org/wiki/List_of_HTTP_status_codes (дата звернення: 29.05.2025).
13. Фіолетовий колір в дизайні [Електронний ресурс] – URL: https://welovebrands.com.ua/ua/blogs/fioletovyj-kolir-v-dizajni/ (дата звернення: 29.05.2025).
14. React [Електронний ресурс] – URL: https://react.dev/ (дата звернення: 29.05.2025).
15. React Router [Електронний ресурс] – URL: https://reactrouter.com/ (дата звернення: 29.05.2025).
16. Посилання на github – URL: https://github.com/elina-chrk/2025_B_PI_PZPI-21-3_Cherkasova_E_S