ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙНАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка програмного забезпечення для організації подорожей автобусом»

на здобуття освітнього ступеня бакалавра

зі спеціальності 121 Інженерія програмного забезпечення

освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Богдан ПІКУШ

(підпис)

Виконав: здобувач вищої освіти групи ПД-43

Богдан ПІКУШ

Керівник: Богдан КАЛИНЮК

асистент кафедри ІПЗ

Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Пікушу Богдану Юрійовичу

1. Тема кваліфікаційної роботи: «Розробка програмного забезпечення для організації подорожей автобусом»
керівник кваліфікаційної роботи асистент кафедри ІПЗ Богдан КАЛИНЮК,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56.

2. Строк подання кваліфікаційної роботи «02» червня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про архітектуру клієнт-серверних веб-застосунків, методи реалізації RESTful API, документація платформ Node.js та Express.js, а також інформація про процеси організації автобусних перевезень.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Огляд застосунку для організації подорожей автобусом та аналіз існуючих аналогів, визначення вимог до програмного забезпечення.

2. Огляд та аналіз засобів реалізації застосунку для організації подорожей автобусом.

3. Проєктування архітектури застосунку для організації подорожей автобусом.

4. Програмна реалізація та тестування застосунку для організації подорожей автобусом.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до застосунку.
3. Програмні засоби реалізації.
4. Діаграма варіантів використання.
5. Діаграма структури бази даних.
6. Діаграма розгортання клієнт-серверної моделі.
7. Екранні форми.
8. Демонстрація роботи застосунку.

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	25.02.-04.03.2024	
2	Аналіз та дослідження існуючих аналогів	05.03-13.03.2024	
3	Огляд та аналіз застосунку для вивчення японської мови	13.03-15.03.2024	
4	Огляд та аналіз засобів реалізації застосунку для організації подорожей автобусом	16.03-21.03.2024	
5	Проєктування архітектури застосунку для організації подорожей автобусом	22.03-03.04.2024	
6	Програмна реалізація та тестування застосунку для організації подорожей автобусом	04.04-28.04.2024	
7	Оформлення роботи: вступ, висновки, реферат	29.04-11.05.2024	
8	Розробка демонстраційних матеріалів	12.05-18.05.2024	
9	Попередній захист роботи	19.05-30.05.2024	

Здобувач вищої освіти _____ Богдан ПІКУШ
(підпис)

Керівник
кваліфікаційної роботи _____ Богдан КАЛИНЮК
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 50 стор., 1 табл., 30 рис., 10 джерел.

Мета роботи - Розширення функціональності системи організації автобусних поїздок шляхом розробки веб-застосунку з використанням сучасних технологій розробки програмного забезпечення.

Об'єкт дослідження - Процес організації автобусних перевезень.

Предмет дослідження - Методи, інструменти та технології розробки веб-застосунку для бронювання автобусних поїздок.

Короткий зміст роботи: У цій дипломній роботі представлено процес розробки веб-застосунку для організації автобусних поїздок з можливістю онлайн-бронювання квитків. Проєкт охоплює розробку клієнтської частини для пасажирів, адміністративної панелі для менеджменту маршрутів та поїздок, а також серверної частини з базою даних.

У вступі обґрунтовано актуальність теми, сформульовано мету, задачі та об'єкт дослідження. У першому розділі наведено аналіз предметної області, визначено технічні та функціональні вимоги до системи. У другому – описано вибрані мови програмування, фреймворки, інструменти розробки та середовище виконання. Третій розділ присвячено архітектурі проєкту, моделі даних та діаграмам UML. У четвертому розділі представлено реалізацію клієнтської й серверної частин, структуру проєкту, інтерфейси користувача та адміністратора, а також описано процес тестування. Завершує роботу перелік джерел та висновки.

КЛЮЧОВІ СЛОВА: BACK-END, FRONT-END, REACT.JS, NODE.JS, RESTful API, ВЕБ-ЗАСТОСУНОК.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – Програмне забезпечення

API – Application Programming Interface

CRUD – Create, Read, Update, Delete

DOM – Document Object Model

IDE – Integrated Development Environment

JSON – JavaScript Object Notation

JWT – JSON Web Token

NoSQL – Not Only Structured Query Language

REST API – Representational State Transfer Application Programming Interface

SPA – Single Page Application

UI – User Interface

ВСТУП

Актуальність: Організація автобусних перевезень є важливим і актуальним завданням у сучасному світі, особливо з огляду на зростаючу мобільність населення та популярність внутрішнього туризму. Використання сучасних інформаційних технологій дозволяє значно спростити процеси бронювання, продажу квитків та організації поїздок і зробити ці послуги більш зручними та доступними для широкого кола користувачів. Веб-застосунки, що реалізують ці функції, допомагають автоматизувати бізнес-процеси, створити прозорість для перевізників та підвищити якість обслуговування пасажирів.

Веб-застосунок - це комплексне рішення, яке об'єднує всі інструменти, необхідні для організації автобусних поїздок, в одному інтерфейсі. Він дозволяє користувачам швидко знаходити потрібні маршрути, бронювати місця, здійснювати платежі та керувати своїми поїздками. Сучасні інструменти, такі як сповіщення в Telegram-ботах та модулі управління поїздками для адміністраторів, дозволяють максимально автоматизувати операційні завдання. Це робить процес організації поїздок більш зручним, ефективним і приємним як для користувачів, так і для менеджерів сервісу.

*Об’єктом дослідже*ння є процес організації автобусних перевезень.

Предметом дослідження є методи, інструменти та технології розробки веб-застосунку для бронювання автобусних поїздок.

Метою роботи є Розширення функціональності системи організації автобусних поїздок шляхом розробки веб-застосунку з використанням сучасних технологій розробки програмного забезпечення.

Методи дослідження: В рамках дослідження було проаналізовано існуючі системи автобусних перевезень (Tickets.ua, FlixBus, BlaBlaCar Bus) для виявлення їхніх переваг та недоліків, а також для формулювання функціональних та нефункціональних вимог до власного рішення. Було обрано сучасний стек

технологій, що включає Node.js та Express.js на серверній стороні, MongoDB Atlas як базу даних, React та Vite на клієнтській стороні, GitHub для контролю версій. Процес розробки включав прототипування, безпосереднє кодування та тестування додатку.

Наукова новизна роботи визначається реалізацією таких функціональних компонентів: Telegram-бот для миттєвих повідомлень о помилок на сервері, адаптований інтерфейс для адміністраторів з можливістю управління маршрутами, рейсами та бронюваннями, а також використання сучасних практик RESTful API.

Практична значущість результатів полягає у можливості застосування створеного програмного рішення реальними перевізниками для автоматизації організації автобусних перевезень, що дозволить їм, оптимізувати бізнес-процеси та покращити взаємодію з клієнтами.

1 АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОРГАНІЗАЦІЇ ПОДОРОЖЕЙ АВТОБУСОМ

1.1 Дослідження предметної області

У наш час автобусні перевезення стають все більш популярними завдяки зростаючій мобільності населення та інтенсифікації внутрішнього туризму. Впровадження сучасних інформаційних технологій суттєво змінює процес бронювання квитків та управління автобусами, надаючи пасажирам зручні інструменти для планування та здійснення поїздок.

Основні переваги ПЗ для організації автобусних подорожей полягають у наступному:

– Автоматизація бізнес-процесів, зменшення кількості помилок та часу, необхідного для здійснення бронювання;

– Зручність та доступність онлайн бронювання, незалежно від місця та часу доби;

– Широкий вибір маршрутів і рейсів, доступних в одній системі завдяки інтеграції з авіакомпаніями;

– Ціни, розклад та умови проїзду можна порівняти без необхідності відвідувати офіс або автовокзал;

– Інтеграція з сучасними платіжними системами забезпечує швидкі та безпечні транзакції.

Аналіз показав, що такі веб-застосунки можуть значно покращити взаємодію між пасажирами та транспортними компаніями, забезпечуючи прозорість та оперативність інформування. Однак до їх недоліків можна віднести залежність від стабільного мережевого з'єднання, відсутність прямого контакту з персоналом транспортної компанії та потребу в регулярній технічній підтримці для забезпечення безперебійної роботи системи.

Тому розробка ПЗ для організації автобусних поїздок є актуальним і важливим завданням, рішення якого можуть кардинально покращити якість транспортних послуг, зробити їх простішими у використанні та комфортнішими для користувачів.

1.2 Специфіка організації автобусних подорожей

Організація автобусних перевезень має певні особливості, які суттєво впливають на розробку ПЗ в цій галузі. До них відносяться

Інформаційні потреби:

– Потреба в точному та актуальному оновленні розкладу руху автобусів;

– Забезпечення прозорості та доступності інформації про ціни та умови бронювання;

– Оперативне інформування пасажирів про зміни маршрутів або рейсів.

Технічна функціональність:

– Вимагає високопродуктивного програмного забезпечення для швидкого обслуговування великої кількості одночасних запитів;

– Впровадити надійні системи безпеки для захисту персональних даних та фінансових транзакцій;

– Необхідно забезпечити масштабованість системи, щоб не відставати від зростаючої кількості користувачів.

Операційні аспекти:

– Зручність використання інтерфейсів пасажира та системного адміністратора;

– Впровадження інструментів моніторингу процесів бронювання та продажу;

– Надання надійної технічної підтримки та швидке реагування на потенційні збої та проблеми.

1.3 Цільова аудиторія

Основними користувачами програмного забезпечення для організації автобусних перевезень є такі цільові групи:

1) Пасажири, які часто користуються автобусним сполученням або щодня їздять до місця проживання.

2) Туристи, які планують внутрішні подорожі та хочуть швидко і зручно бронювати квитки онлайн.

3) Менеджери та оператори автобусних компаній, які зацікавлені в автоматизації своїх маршрутів, управлінні рейсами та процесами контролю бронювання.

4) Власники автобусних перевізників, які бажають підвищити ефективність та прозорість бізнесу за рахунок використання сучасних інформаційних систем.

1.4 Дослідження існуючих аналогів

На сьогоднішній день одними з найпопулярніших вебсервісів у сфері онлайн-бронювання автобусних квитків є Tickets.ua, FlixBus та BlaBlaCar Bus. Розглянемо їх особливості детальніше.

1.4.1 Tickets.ua

Tickets.ua - популярний український онлайн-сервіс, який пропонує бронювання квитків на всі види транспорту, включаючи автобусні, залізничні, авіа та інші подорожі. Платформа орієнтована на широке коло користувачів, які шукають зручний спосіб планувати маршрути та купувати квитки онлайн. Сервіс вирішує проблему централізованого пошуку транспорту, об'єднуючи різні транспортні компанії та напрямки в одному інтерфейсі.

Основна мета сайту - спростити процес планування подорожей для пересічного користувача, дозволивши йому швидко шукати доступні рейси,

порівнювати ціни, бронювати та оплачувати квитки всього в кілька кліків.

Основні можливості Tickets.ua включають:

– Пошук рейсів: пошук автобусних, залізничних та авіаперевезень за заданими параметрами (дата, маршрут, час);

– Онлайн-бронювання: миттєве оформлення замовлення з подальшим автоматичним надсиланням підтвердження на електронну пошту;

– Фільтр: сортування транспорта за ціною, часом відправлення та тривалістю подорожі;

– Мультимодальна підтримка: об'єднання автобуса, поїзда, літака і трансфера в одному інтерфейсі;

– Особистий кабінет: збереження історії замовлень, перебронювання та повернення;

– Інтеграція з мобільними додатками: платформа має мобільну версію та додатки, що дозволяють здійснювати покупки з мобільного телефону;

– Багатомовність: сайт доступний українською та англійською мовами;

– Підтримка клієнтів: функціонал зворотного зв'язку, гаряча лінія, служба підтримки 24/7.

Переваги:

– Зручний інтерфейс: інтуїтивно зрозумілий дизайн та навігація дозволяють навіть недосвідченим користувачам швидко знаходити та бронювати квитки;

– Універсальність: підтримує різні види транспорту та охоплює широкий спектр маршрутів в Україні та за її межами;

– Безпека платежів: безпечна платіжна система з можливістю оплати банківською картою та іншими сервісами;

– Інтеграція з агрегаторами: платформа співпрацює з багатьма перевізниками та пропонує широкий вибір квитків.

Недоліки:

– Залежність від сторонніх систем: у багатьох випадках платформа є не

прямим продавцем, а посередником, що може ускладнити процес повернення або обміну квитків;

– Відсутність відкритого API: сторонні розробники не можуть інтегрувати сервіс у власні проекти або додатки;

– Обмежений функціонал для транспортних компаній: платформа не може надати операторам транспортних компаній комплексне логістичне рішення, наприклад, управління рейсами, маршрутами або аналітикою.

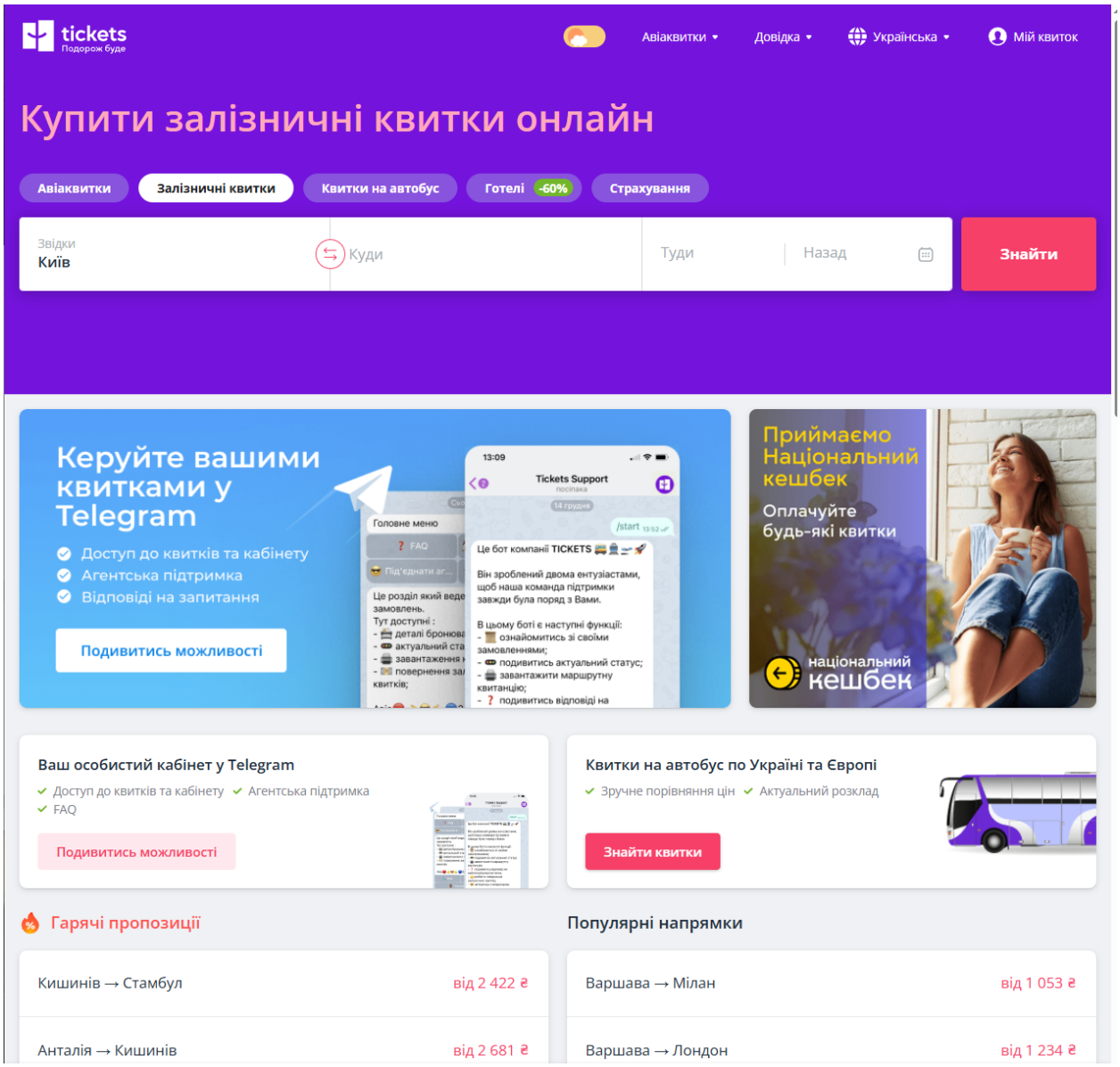


Рис. 1.1 Головна сторінка Tickets.ua

1.4.2 FlixBus

FlixBus - це міжнародна платформа автобусних перевезень, заснована в Німеччині в 2013 році, яка виросла до одного з найбільших операторів міжміських автобусних перевезень в Європі та США. FlixBus поєднує сервіс онлайн-бронювання з мережею партнерських автобусних компаній для забезпечення високоавтоматизованого обслуговування клієнтів та оптимізації маршрутів.

Компанія реалізує концепцію «мобільний як сервіс», надаючи зручний інструмент для бронювання поїздок через свій веб-сайт та мобільні додатки, не маючи власних автобусів, а співпрацюючи з місцевими перевізниками.

Основні можливості FlixBus:

– Онлайн-бронювання: швидкий пошук автобусів з детальною інформацією про маршрути, час, зупинки та ціни;

– Мобільний застосунок: повнофункціональний застосунок для iOS та Android з можливістю продажу електронних квитків, пуш-сповіщень та відстеження поїздок;

– Особистий кабінет: зберігє історію поїздок, дає змогу керувати квитками та швидко перебронювати квитки;

– Інтеграція з Google Maps та іншими сервісами: зручне планування маршруту та визначення місця розташування зупинок;

– Багатомовність: інтерфейс підтримує понад 30 мов;

– Система знижок: Акції, промокоди, знижки для молоді, пенсіонерів тощо;

– Технічна підтримка: Багатомовна підтримка клієнтів через чат, телефон або електронну пошту.

Переваги:

– Широке покриття: понад 2 500 напрямків у більш ніж 35 країнах Європи та США;

– Доступні ціни: Конкурентоспроможні ціни завдяки оптимізованій

моделі співпраці;

– Стабільна якість послуг: Єдині стандарти обслуговування, Wi-Fi, розетки, зручні сидіння в автобусах;

– Відкритий API (обмежений): компанія пропонує API для доступу до розкладів і тарифів, але тільки для офіційних партнерів (доступ через комерційні угоди).

Недоліки:

– Не відкритий для інтеграції сторонніх розробників: відкритий API не має широкого розповсюдження, що ускладнює розробку сторонніх сервісів на основі даних FlixBus;

– Обмежений доступ до внутрішнього функціоналу: адміністративні функції (управління рейсами, аналітика, CRM) недоступні для сторонніх розробників;

– Орієнтація на міжнародні ринки: сервіс має обмежену присутність в Україні і поки що не підтримує внутрішні маршрути у великих масштабах.

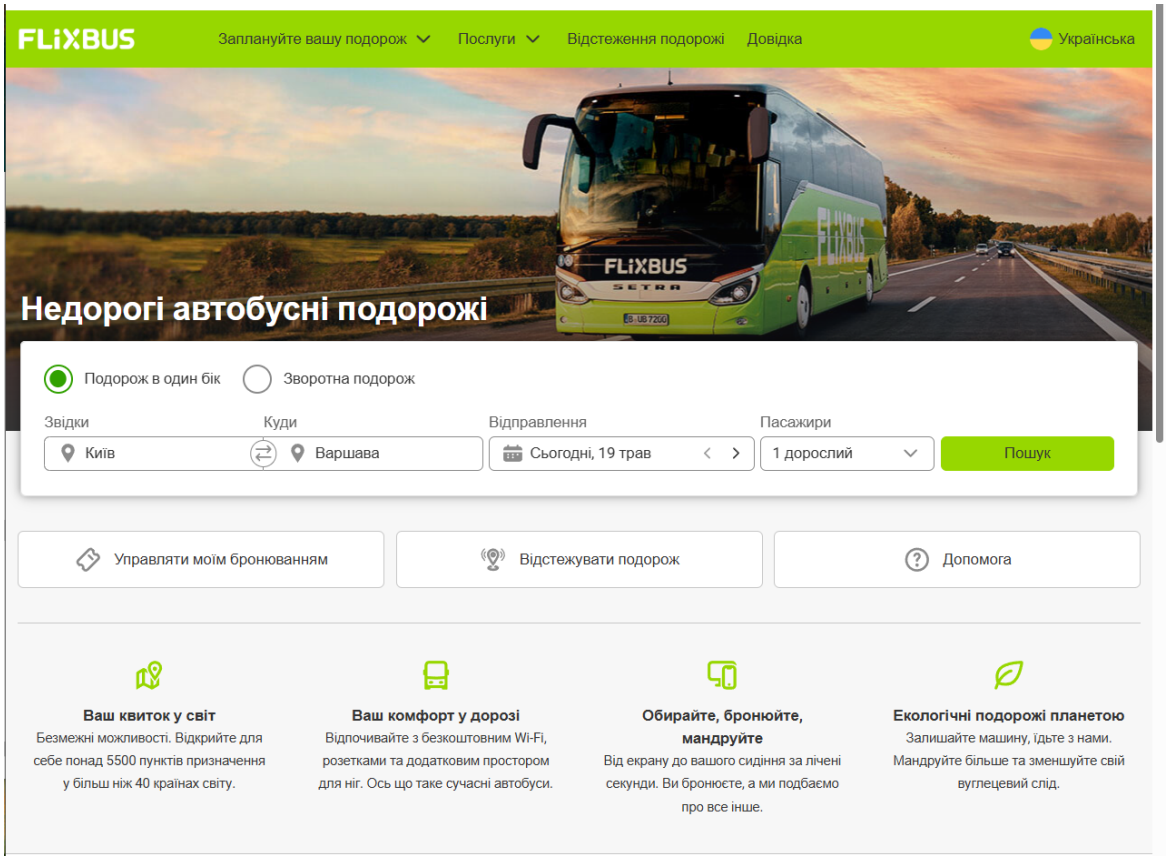


Рис. 1.2 Головна сторінка Flixbus.ua

1.4.3 BlaBlaCar Bus

BlaBlaCar Bus - це автобусний сервіс, який є частиною екосистеми BlaBlaCar, відомої платформи спільного використання автомобілів. Спочатку відомий як Ouibus, сервіс був придбаний BlaBlaCar у 2018 році, щоб розширити свої рішення для мобільності та охопити користувачів, які віддають перевагу організованому транспорту. BlaBlaCar Bus поєднує зручність онлайн-бронювання з недорогими міжміськими автобусними поїздками в Європі.

Сервіс орієнтований на молодь, туристів та пасажирів, які шукають доступний варіант подорожі. Основна увага приділяється зручності завдяки мобільним додаткам та адаптивним інтерфейсам на смартфонах.

Серед ключових особливостей BlaBlaCar Bus:

– Онлайн-бронювання на сайті : простий механізм пошуку та придбання квитків через сайт або мобільний додаток;

– Інтеграція з BlaBlaCar: комбіноване бронювання (автобус + автопоїздка) в одному додатку;

– Мобільний додаток: доступний для iOS та Android, підтримує електронні квитки та push-повідомлення;

– Цифрові квитки: не потрібно друкувати квитки - просто покажіть QR-код на своєму мобільному телефоні;

– Гнучка система знижок: акції та знижки для молоді, груп та тих, хто часто подорожує;

– Багатомовність: додатки та веб-сайти підтримують широкий спектр європейських мов;

– Карта маршрутів: візуалізація доступних напрямків на інтерактивній карті.

Переваги:

– Легко інтегрується з карпулінгом: користувачі можуть поєднувати різні форми поїздок в одній системі;

– Орієнтація на мобільні пристрої: інтерфейс працює зі смартфонами, що робить його особливо зручним для молоді;

– Низька ціна: сервіс позиціонується як бюджетна альтернатива іншим авіакомпаніям з привабливими тарифами;

– Електронні квитки та зручність: швидке бронювання квитків без паперової тяганини.

Недоліки:

– Відсутність відкритого API: API сервісу недоступний для сторонніх розробників, що унеможливлює інтеграцію з іншими платформами або системами бронювання;

– Обмежений функціонал управління: платформа не пропонує інструментів для перевізників або адміністраторів, таких як аналітика, управління рейсами або користувачами;

– Дуже мала присутність в Україні: сервіс переважно працює в країнах ЄС і не охоплює внутрішні маршрути в Україні;

– Обмежена підтримка корпоративного використання: не існує окремого рішення для корпорацій або агентів.

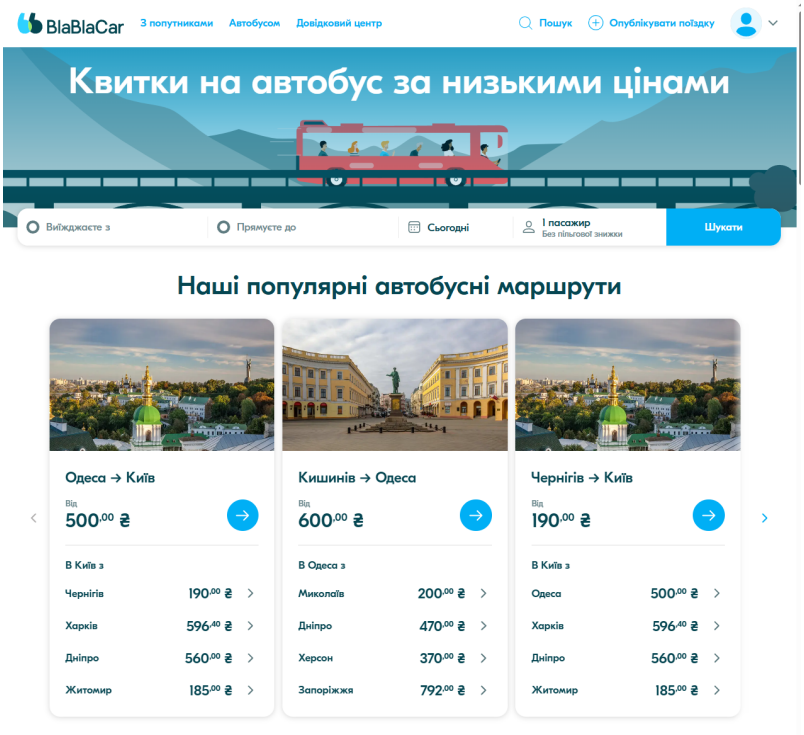


Рис. 1.3 Головна сторінка blablacar.com.ua

В таблиці 1.1 зведено результати аналізу існуючих застосунків для автобусних поїздок.

Таблиця 1.1

Показник	**Tickets.ua**	**FlixBus**	**BlaBlaCar Bus**
Наявність веб-платформи	+	+	+
Наявність мобільного додатку	+	+	+
Відкритий API	-	Частково (тільки для партнерів)	-
Можливість адміністрування маршрутів	-	-	-
Міжнародне покриття	+	+	+
Простота бронювання	+	+	+
Основна цільова аудиторія	**Користувачі з України, які планують подорожі різними видами транспорту**	**Пасажири, що подорожують на далекі відстані по Європі та США**	**Молоді мандрівники та користувачі мобільних сервісів**

1.5 Визначення вимог до застосунку

Додаток для управління автобусними перевезеннями повинен забезпечувати повний цикл взаємодії з користувачем - від пошуку рейсів до бронювання. Основними функціональними вимогами є можливість реєстрації та авторизації користувачів, пошук рейсів за заданими параметрами (місто, дата, час), бронювання.

Важливою є наявність особистого кабінету, де користувачі можуть переглядати історію бронювання. Розділ адміністрування передбачає окремі модулі для управління маршрутами, рейсами та перегляду бронювань.

Нефункціональні вимоги включають простий та інтуїтивно зрозумілий інтерфейс, високу продуктивність сервісу, безпечне зберігання персональних даних та стабільну роботу системи. З технічної точки зору, система повинна базуватися на сучасних веб-технологіях: Node.js на стороні сервера, React на стороні клієнта, а також бази даних MongoDB.

Архітектура проекту має бути клієнт-серверною, реалізованою за допомогою REST API. Інтерфейс також має бути зручним для користувача і інтуїтивно зрозумілим.

2 ЗАСОБИ РЕАЛІЗАЦІЇ

2.1 Середовище розробки

IDE - це програмний комплекс, що поєднує в собі інструменти, необхідні для написання, редагування, компіляції, налагодження та тестування програмного коду. Основна мета IDE - надати розробникам зручне та ефективне середовище для створення програмних продуктів, мінімізуючи час, що витрачається на рутинні завдання. Середовище зазвичай включає в себе редактори коду, засоби автозавершення, термінали, засоби управління файлами, а також засоби налагодження та інтеграції з системами контролю версій.

У цьому проекті ми використовуємо Visual Studio Code (VS Code) як основне середовище розробки - безкоштовний, легкий, але потужний редактор коду, розроблений компанією Microsoft. VS Code підтримує велику кількість мов програмування і має відкриту архітектуру плагінів, яка може бути адаптована до будь-яких потреб розробника.

Середовище Visual Studio Code відіграло ключову роль у реалізації програмного забезпечення для управління автобусними перевезеннями, забезпечивши легкість розробки серверної логіки, побудови користувацьких інтерфейсів, налаштування маршрутизації, валідації форм та взаємодії з базою даних MongoDB на Node.js.

Як результат, використання сучасних середовищ розробки значно підвищує ефективність процесу розробки та забезпечує відповідність коду сучасним стандартам якості.

2.2 Мова програмування

Для реалізації програмного забезпечення для управління рухом автобусів ми використовували JavaScript, одну з найпопулярніших і універсальних мов у

сучасній веб-розробці.

JavaScript - це динамічна мова високого рівня, яка спочатку використовувалася для створення інтерактивних елементів на веб-сторінках. Однак з часом, з появою середовищ виконання, таких як Node.js, JavaScript перетворився на повноцінну мову програмування, яка використовується як на фронтенді, так і на бекенді.

На фронтенді JavaScript використовується для створення динамічних інтерфейсів, обробки подій, валідації форм та взаємодії з серверами через API, а бібліотеки та фреймворки, такі як React, Vue.js або Angular, значно розширили можливості мови JavaScript і спростили створення сучасних односторінкових додатків (SPA).

На бекенді, за допомогою середовища Node.js, JavaScript дозволяє створювати повноцінні сервери, обробляти HTTP-запити, працювати з базами даних, забезпечувати авторизацію та інші логічні процеси. Для організації серверної частини часто використовують фреймворки, такі як Express.js.

Завдяки своїй гнучкості, великій спільноті та багатій екосистемі JavaScript залишається основною мовою для веб-розробки, що дозволяє створювати повнофункціональні додатки - від невеликих веб-сайтів до великих платформ. У цьому проекті одна мова використовується як для клієнта, так і для сервера, що значно спрощує процес розробки, тестування та підтримки системи.

2.3 Веб-фреймворки

Web Framework - це набір готових інструментів, бібліотек і фреймворків, які полегшують розробку веб-застосунків. Він надає розробникам шаблони, стандарти та готові фреймворки, які допомагають їм створювати веб-сайти та веб-сервіси швидше та якісніше.

Основна мета використання веб-фреймворків - спростити і прискорити процес розробки, зменшити кількість рутинного коду і забезпечити логічну організацію проекту. В результаті програмісти можуть зосередитися на бізнес-

логіці програми, а не на технічних деталях.

Існують фреймворки для фронтенду (тобто інтерфейсу, який взаємодіє з користувачем) - наприклад, React.js, Vue.js, Angular - і фреймворки для бекенду (тобто серверної частини, яка обробляє запити, працює з базами даних тощо).

Веб-фреймворки дозволяють:

– Створювати структуровані проекти з чіткими лініями відповідальності;

– Використовувати готові компоненти для авторизації, управління базами даних, маршрутизації тощо

– Використовувати вбудовані механізми для забезпечення захисту та безпеки веб-застосунків;

– Реалізовувати швидку інтеграцію з іншими сервісами або API;

– Використання перевірених рішень для оптимізації продуктивності та зменшення кількості помилок.

2.3.1 Express.js

Express.js - один з найпопулярніших фреймворків для створення серверних додатків на основі Node.js. Він надає мінімальну, але потужну інфраструктуру для створення веб-серверів, REST API та обробки HTTP-запитів.

Ключові можливості Express.js:

– Маршрутизація: Просте визначення маршрутів (GET, POST, PUT, DELETE) та обробка параметрів запитів.

– Проміжне програмне забезпечення: Підтримка проміжного програмного забезпечення дозволяє обробляти запити до того, як вони досягнуть основної логіки. Це корисно для авторизації, логування та обробки помилок.

– Гнучкість: Express не нав'язує жорстких структур, дозволяючи створювати як прості API, так і великі модульні додатки.

– Широка екосистема: Багато готових рішень і бібліотек (CORS, Helmet, експрес-валідатор, мультиплікатор тощо).

– Ідеальна інтеграція з MongoDB (через Mongoose) або іншими базами

даних.

В рамках цього проекту Express.js використовується для реалізації маршрутизації, обробки попередньо визначеної логіки, взаємодії з базами даних, перевірки авторизації користувачів та генерації відповідей API.

2.3.2 React.js

React.js - це бібліотека JavaScript, розроблена компанією Meta (Facebook) для створення користувацьких інтерфейсів. Хоча формально React вважається бібліотекою, у практиці веб-розробки його часто називають фреймворком через розвинену екосистему та організацію інтерфейсу користувача.

Особливості React.js:

– Компонентна архітектура: інтерфейс розділений на окремі компоненти (наприклад, форма входу, список рейсів, квиткова картка), які можна легко повторно використовувати та тестувати.

– Віртуальний DOM: React не використовує DOM браузера напряму, натомість створює віртуальну копію та оновлює лише ті частини інтерфейсу, які змінилися, що може значно пришвидшити роботу додатку.

– Стан та події: React дозволяє ефективно керувати станом вашого застосунку та швидко реагувати на події користувача.

– Підтримка SPA (Single Page Application): React дозволяє створювати односторінкові додатки, в яких можна переходити між сторінками без перезавантаження браузера.

– Сумісність з REST API: Ідеально підходить для створення фронтендів, які взаємодіють з бекендом через API.

У цьому проекті React.js використовувався для побудови користувацького інтерфейсу: пошуку рейсів, бронювання, історії замовлень, сторінок реєстрації та входу. Завдяки реактивності та компонентній моделі користувацький досвід є швидким і простим.

2.4 Технології зберігання та керування даними

В рамках проекту для зберігання, обробки та управління даними використовується MongoDB - сучасна файлово-орієнтована система управління базами даних (СКБД) класу NoSQL. На відміну від традиційних реляційних баз даних, MongoDB зберігає інформацію у вигляді JSON (BSON) файлів, що забезпечує гнучкість структури даних і високу продуктивність при виконанні CRUD операцій (Create, Read, Update, Delete).

MongoDB ідеально підходить для веб-застосунка з частою зміною даних або складними вкладеними структурами. Завдяки горизонтальній масштабованості, підтримці реплікації, гнучкості запитів та легкій інтеграції з Node.js, ця СУБД широко використовується в сучасній розробці.

Для полегшення розгортання баз даних у хмарі ми використовуємо MongoDB Atlas - офіційний хмарний сервіс MongoDB, який забезпечує автоматичне резервне копіювання, безпеку доступу, моніторинг продуктивності та зручне управління через веб-інтерфейс. MongoDB Atlas спрощує процес створення баз даних та надає можливість підключатися до них з будь-якої точки світу, що особливо важливо при роботі над проектами командної розробки або хостингу. MongoDB Atlas спрощує процес створення баз даних і надає можливість підключатися до них з будь-якої точки світу, що особливо корисно при роботі над проектами командної розробки або хостингу.

Для локального перегляду структур баз даних, колекцій та документів ми використовуємо MongoDB Compass - офіційний графічний клієнт, який дозволяє легко переглядати дані, будувати фільтри, виконувати запити, перевіряти індекси та швидко переміщатися структурами баз даних.

Таким чином, поєднання MongoDB, MongoDB Atlas і Compass забезпечило надійне, масштабоване та зручне середовище для роботи з даними, що значно спростило реалізацію серверної логіки додатку.

2.5 Система контролю версій

Програмне забезпечення було розроблено з використанням системи контролю версій Git у поєднанні з хостинговим сервісом GitHub. Ці інструменти є стандартом для сучасної розробки програмного забезпечення, особливо у відкритих або командних проектах.

Git - це децентралізована система контролю версій, яка дозволяє відстежувати всі зміни у вихідному коді проекту. Git дозволяє розробникам працювати незалежно один від одного, а потім об'єднувати свої напрацювання в загальну кодову базу.

GitHub - популярна платформа для розміщення Git-репозиторіїв у хмарі. Вона надає зручний веб-інтерфейс для перегляду коду, управління комітами, створення запитів на витягування, управління проблемами, документування проектів (README, вікі), а також базові інструменти CI/CD. GitHub дозволяє співпрацювати над проектами, коментувати зміни коду, автоматизувати контрольні суми та релізи в команді.

У цьому проекті Git використовується для контролю локальних змін, створення гілок для певних функцій і фіксації версій, а GitHub - для зберігання репозиторію в хмарі, його резервного копіювання і забезпечення віддаленої роботи. Завдяки цим інструментам було забезпечено зручне управління кодовою базою, контроль якості, швидкий доступ до історії змін та безпеку зберігання проєкту.

2.6 Додаткові інструменти та бібліотеки

Інші інструменти та бібліотеки також використовуються в цьому проекті, щоб допомогти в ефективній розробці, покращити якість коду та зробити API та інтерфейси простішими у використанні.

1) Axios - це бібліотека для створення HTTP-запитів. Вона використовується на фронтенді (React) для надсилання запитів до серверних API, і Axios дозволяє легко обробляти відповіді, виправляти помилки та працювати з

асинхронними запитами.

2) React Router - це бібліотека для організації маршрутів у SPA-додатках. Вона дозволяє створювати логічні сторінки (наприклад, /login, /bookings) без перезавантаження, забезпечуючи зручну навігацію між компонентами.

3) dotenv - бібліотека для зберігання конфіденційної інформації у файлах .env для безпечного зберігання параметрів з'єднання з базами даних, портів сервера та інших змінних оточення.

4) Postman - зовнішній інструмент для тестування запитів до API на етапі розробки. З його допомогою можна перевірити коректність відповідей на стороні сервера без необхідності запускати фронтенд.

5) Mongoose - це обгортка для використання MongoDB в Node.js. Вона спрощує побудову моделей, перевірку даних, встановлення зв'язків між колекціями та інші базові операції, пов'язані з базами даних.

6) Helmet і CORS - бібліотеки безпеки, що використовуються в Express.js. Helmet додає заголовки безпеки до відповідей сервера, а CORS дозволяє встановлювати політики доступу до API з боку клієнта.

7) MUI (Material UI) - бібліотека готових UI-компонентів для React, створена на основі принципів Material Design від Google. Була використана для побудови адаптивного та привабливого інтерфейсу користувача.

3 ПРОЄКТУВАННЯ

3.1 Проєктування архітектури застосунку

Архітектура програми базується на класичній моделі клієнт-сервер, де інтерфейсна (клієнтська) частина чітко відокремлена від серверної логіки. Такий підхід забезпечує більшу гнучкість, масштабованість і простоту обслуговування програмного забезпечення.

Клієнт реалізований за допомогою бібліотеки React.js. Вона відповідає за відображення користувацького інтерфейсу, взаємодію з користувачами, надсилання запитів до API та обробку відповідей. Всі дані завантажуються динамічно через REST-запити, тому є можливість створювати односторінкові додатки (SPA) без необхідності повністю перезавантажувати сторінки.

Бекенд розроблений з використанням фреймворку Express.js в середовищі Node.js. Логіка сервера обробляє вхідні HTTP-запити, виконує перевірку даних, взаємодіє з базою даних, виконує авторизацію, створює бронювання, переглядає маршрути і т.д. Всі запити клієнтів проходять через централізований API для забезпечення надійної та безпечної взаємодії. Всі запити клієнтів проходять через централізований API для забезпечення надійної та безпечної взаємодії.

База даних - зберігання інформації реалізовано через MongoDB. Всі дані (користувачі, маршрути, маршрути, бронювання) зберігаються у вигляді файлів. Для полегшення взаємодії з базою даних було використано бібліотеку Mongoose, яка дозволяє використовувати об'єкти, моделі та схеми даних на високому рівні абстракції.

Зв'язок між фронт-ендом і бек-ендом здійснюється за протоколом HTTP з використанням REST API. Всі дані передаються у форматі JSON. Аутентифікація виконується на основі розмітки (JWT).

Рис. 3.1 Діаграма розгортання клієнт-серверної моделі

3.2 Архітектура клієнтської частини

Клієнтська частина додатку реалізована за допомогою бібліотеки React.js, що забезпечує сучасний, реактивний та динамічний користувацький інтерфейс. Завдяки компонентному підходу інтерфейс розділений на окремі логічні частини (компоненти), які можна легко розширювати, повторно використовувати та тестувати.

Загальна структура клієнта складається з двох окремих фронт-ендів:

– Веб-застосунок для користувачів (пасажирів) - дозволяє їм шукати рейси, переглядати інформацію про подорожі, реєструватися, входити в систему та бронювати квитки.

– Адміністративна панель менеджера компанії - надає можливість керувати маршрутами, маршрутами, переглядати історію бронювання та взаємодіяти з базою даних через API.

Зв'язок між клієнтом і сервером реалізований через REST API, а HTTP-запити доставляються за допомогою бібліотеки Axios. Всі запити та відповіді обробляються у форматі JSON.

Обидва клієнтські інтерфейси маршрутизуються за допомогою реалізації React Router, що дозволяє створювати односторінкові додатки (SPA), які можуть швидко перемикатися між сторінками без перезавантаження.

У структурі додатку є кілька основних каталогів:

components/ - багаторазові UI-компоненти (наприклад, форми пошуку, списки рейсів, картки з цінами на авіаквитки);

pages/ - сторінки додатку, кожна сторінка відповідає окремому маршруту (наприклад, /home, /login, /bookings);

assets/ - каталог для зберігання статичних ресурсів: зображень, іконок, шрифтів, файлів стилів та інших візуальних матеріалів. Це дозволяє централізовано керувати всіма медіа-елементами інтерфейсу та спростити їх використання в компонентах.

utils/ - містить допоміжні функції та утиліти, які не є частиною логіки окремих компонентів, але використовуються в різних частинах програми. Наприклад, це можуть бути функції для форматування дат, обробки рядків, перевірок валідації або перетворення даних.

Інтерфейс користувача реалізовано за допомогою MUI (Material UI), популярної бібліотеки компонентів, заснованої на принципах Material Design від Google. Це забезпечує єдиний стиль, адаптивність до різних екранів та швидку розробку інтерфейсу.

Клієнт забезпечує повний користувацький досвід і є ключовим компонентом UX (User Experience), оскільки саме тут здійснюються пошук, бронювання авіаквитків та інші ключові дії.

Рис. 3.2 Структурна діаграма клієнтського вебзастосунку (website)

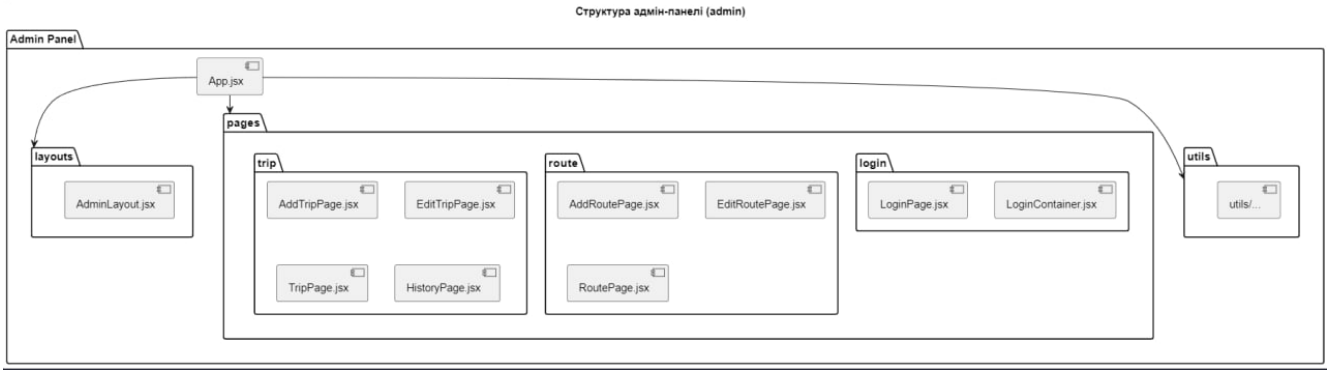


Рис. 3.3 Структурна діаграма інтерфейсу адміністратора (admin panel)

3.3 Архітектура серверної частини

Серверна частина програми реалізована за допомогою середовища виконання Node.js та веб-фреймворку Express.js, що дозволяє швидко будувати надійні REST API.Архітектура серверної частини побудована за модульним принципом, забезпечуючи розбивку логіки на окремі блоки - маршрутизацію, контролери, сервіси, моделі та проміжне програмне забезпечення.

Основні компоненти серверної частини:

– Маршрути - визначають кінцеві точки REST API, запитувані клієнтом. Кожен маршрут відповідає за певну частину функціоналу (наприклад, /auth, /trips, /bookings) і перенаправляє запити до відповідного контролера;

– Контролери - містять основну бізнес-логіку. Вони обробляють дані

користувача, викликають відповідні сервіси, обробляють моделі та генерують відповіді у форматі JSON;

– Сервіси - окремі модулі, які інкапсулюють повторювану логіку (наприклад, створення бронювання, перевірка наявності місць, розрахунок вартості, генерація токенів тощо). Вони викликаються контролером, що робить бізнес-логіку чистою і простою в обслуговуванні. Такий підхід - патерн рівня послуг - збільшує розподіл обов'язків і спрощує тестування;

– Моделі - Використовують бібліотеку Mongoose для опису структури файлів у базі даних MongoDB. Наприклад, існують моделі для User, Trip, Booking, які визначають шаблони полів, типи даних, зв'язки та обмеження;

– Middleware - Окремі функції проміжного ПЗ, які виконуються між отриманням запиту та його обробкою. Наприклад, проміжне програмне забезпечення для авторизації (автентифікація за допомогою токенів JWT), обробки помилок, ведення журналів або CORS;

– Файл конфігурації сервера - в index.js або app.js ініціалізуйте Express, підключіться до проміжного програмного забезпечення, запустіть маршрутизацію, підключіться до MongoDB через Mongoose і запустіть сервер на вказаному порту.

Зв'язок між клієнтом і сервером здійснюється за допомогою HTTP-запитів у форматі JSON, що відповідає принципу REST. Сервер приймає запит, перевіряє авторизацію, обробляє дані, отримує доступ до бази даних і надсилає відповідь клієнту.

Структура сервера є гнучкою та розширюваною завдяки Express.js. Можно легко додавати маршрути або функції, не порушуючи логіку роботи вже запущених частин програми. Крім того, незалежна обробка помилок і повторне використання проміжного програмного забезпечення допомагають підтримувати високу якість коду і стабільність системи.

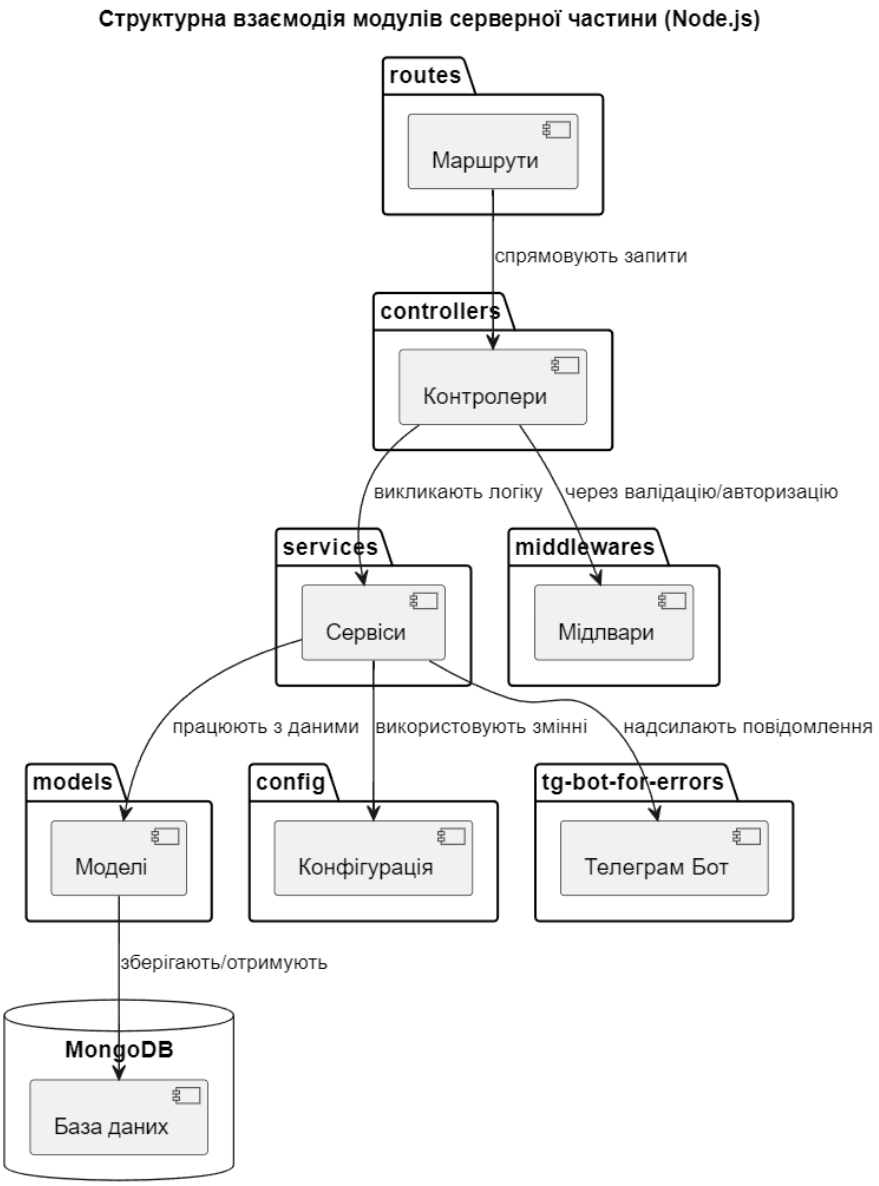


Рис. 3.4 Діаграма взаємозв’язків між модулями серверної частини веб-застосунку

3.4 Модель даних і база даних

Цей проект використовує MongoDB, файлову систему управління базами даних категорії NoSQL, для зберігання та обробки даних. Вона зберігає інформацію у вигляді гнучких BSON-файлів (подібних до JSON), що дозволяє ефективно обробляти динамічно змінювані структури даних і вкладені об'єкти.

Для взаємодії з MongoDB на стороні сервера необхідно використовувати бібліотеку Mongoose, яка забезпечує об'єктно-документне моделювання (ODM). Це дозволяє описувати і моделювати структури документів у вигляді схем і реалізовувати зв'язки між сутностями, валідацію, індексацію та інші функції на високому рівні абстракції.

Основні моделі, реалізовані в базі даних:

– User - зберігає інформацію про зареєстрованого користувача: ім'я, email, пароль (зашифрований), роль в системі (користувач, адміністратор тощо), історію бронювання. Також може містити бейдж або статус облікового запису;

– Admin - окрема модель для адміністраторів або менеджерів компанії. Вона може містити інформацію про компанію, контактну інформацію, рівні доступу, а також бути пов'язана з керованими поїздками або маршрутами;

– Route - містить інформацію про напрямок подорожі (наприклад, «Київ - Львів»). Зазвичай включає початковий і кінцевий пункти, час і можливі зупинки;

– Trip - представляє конкретну реалізацію маршруту, включаючи дату та час відправлення, кількість вільних місць, ціну та автобус. Містить посилання на Маршрут за ObjectId;

– Ticket - зберігає інформацію про квиток, який було заброньовано або придбано. Містить посилання на Trip, User, Seat, Status (Booked/Paid) і Ticket Date.

Зв'язки між моделями реалізуються за допомогою ObjectId (посилання на інші документи). Наприклад, модель Ticket містить поля userId і tripId, які вказують на відповідні документи в інших колекціях, що дозволяє ефективно організувати дані відповідно до логіки реального застосування.

Перевагами використання MongoDB в цьому проекті є гнучкість структури даних, висока продуктивність при обробці запитів, підтримка вкладених об'єктів для зберігання складних структур, таких як квитки або маршрути, а також легка інтеграція з Node.js за допомогою бібліотеки Mongoose.

Уся база даних була розгорнута в хмарі за допомогою MongoDB Atlas, що надало можливість легкого підключення, масштабування, автоматичного

резервного копіювання та безпечного доступу через whitelisted IP.

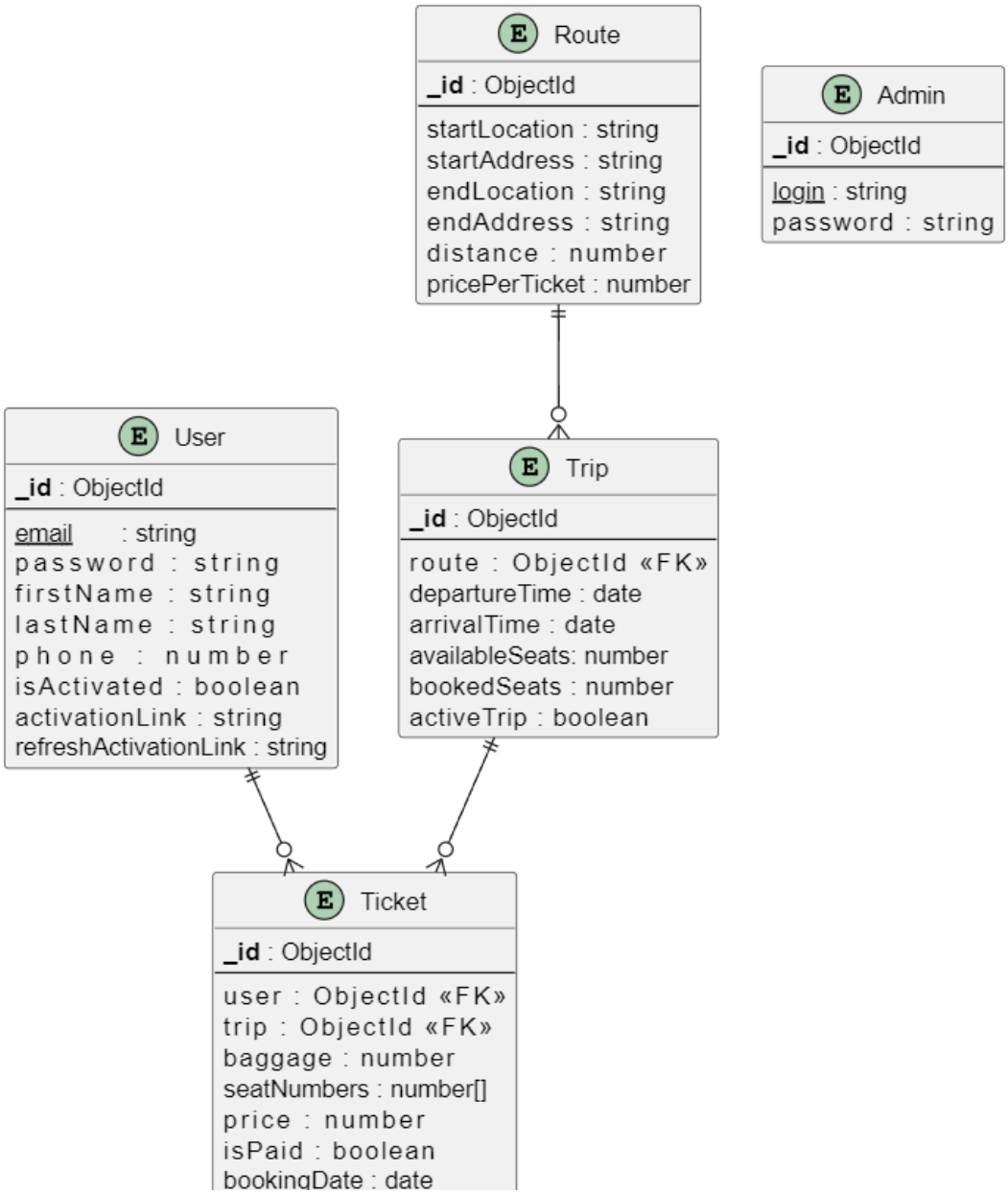


Рис. 3.5 Діаграма класів бази даних

Для візуалізації взаємодії основних учасників із системою було побудовано діаграму варіантів використання (див. рис. 3.6). На ній представлено два актори: Користувач, що взаємодіє з клієнтською частиною застосунку, та Адміністратор (менеджер компанії), який працює через адміністративну панель.

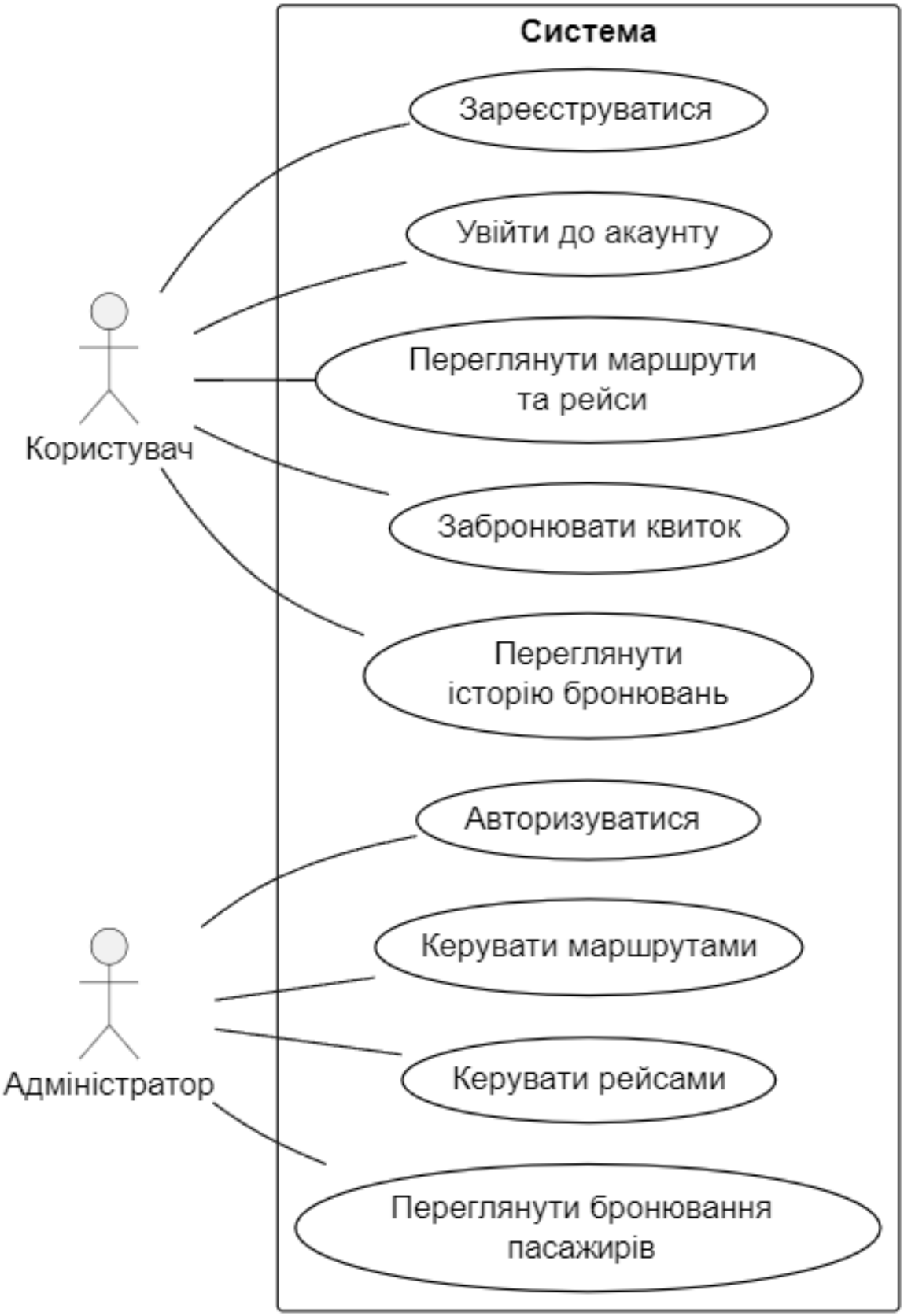


Рис. 3.6 Діаграма варіантів використання

4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

4.1 Реалізація користувацького інтерфейсу

Клієнтська частина веб-застосунку реалізована за допомогою бібліотеки React.js, яка дозволяє будувати швидкі, динамічні та масштабовані інтерфейси на основі компонентного підходу. Всі взаємодії з користувачем реалізовані у форматі SPA (Single Page Application), що забезпечує оновлення контенту в режимі реального часу без необхідності повного перезавантаження сторінки.

Для старту проєкту було використано інструмент Vite, який забезпечує швидке створення React-проєкту з мінімальними налаштуваннями.

Процес ініціалізації відбувався за допомогою наступної команди в терміналі:

npm create vite@latest

Після цього було обрано шаблон React, встановлено залежності командою:

npm install

та запущено проєкт локально:

npm run dev

Результатом ініціалізації є формування базової структури проекту, яка в подальшому розширюється відповідно до потреб програми (див. рис. 4.1). Основний каталог клієнта:

– /src - це кореневий каталог, що містить вихідний код. Він містить всі компоненти, сторінки, стилі, утиліти та налаштування.

– /components - Багаторазові елементи інтерфейсу, такі як заголовки, колонтитули, компоненти списків, блоки пошуку тощо.

– /pages - Окремі сторінки в додатку, які відповідають маршрутам (наприклад, головна сторінка, сторінки бронювання, профілі користувачів).

– /layouts - Макети сторінок, що поєднують спільні елементи інтерфейсу (наприклад, навігація + контент).

– /utils - Підтримка функцій та утиліт, таких як налаштування аксіом для HTTP-запитів, обробка помилок тощо.

– /assets - каталог для зберігання зображень, іконок та інших статичних ресурсів.

– /public - каталог для загальнодоступних файлів, таких як index.html.

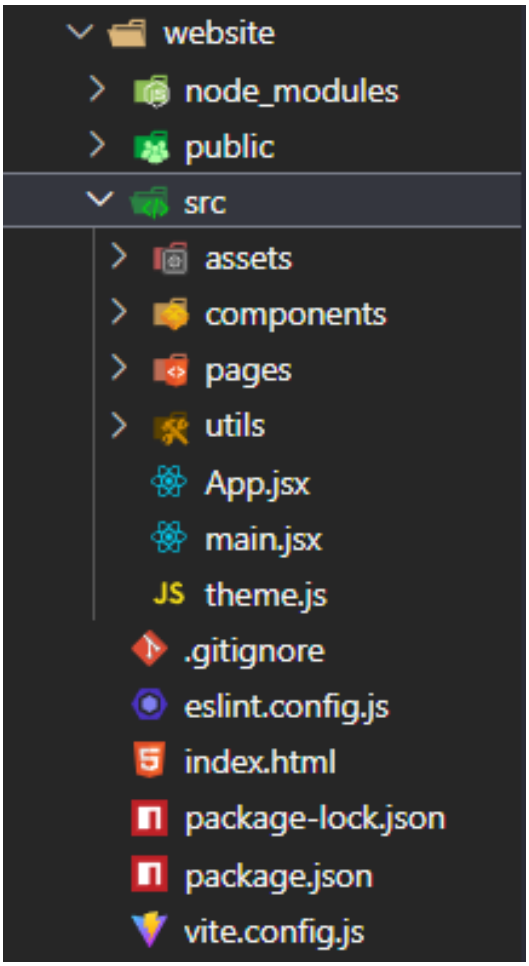


Рис. 4.1 Приклад структури React проєкту після ініціалізації

Для побудови інтерфейсу ми використали Material UI (MUI), популярну бібліотеку компонентів React, яка спрощує створення сучасних, адаптивних та візуально привабливих користувацьких інтерфейсів. Компоненти MUI дозволяють реалізувати навігацію, форми, таблиці, діалоги, кнопки, сповіщення та інші інтерактивні елементи.

Інтерфейс веб-застосунку розділений на дві основні частини:

– Інтерфейс користувача (клієнтський фронт-енд) - орієнтований на

пасажирів, які можуть переглядати маршрути, бронювати квитки та переглядати історію поїздок;

– Адміністративна частина - призначена для співробітників компанії, які можуть керувати маршрутами, рейсами та переглядати деталі забронюваних поїздок.

У наступних розділах детально описані функції кожної частини інтерфейсу.

4.1.1 Інтерфейс користувача

Інтерфейс користувача призначений для пасажирів, які взаємодіють із системою бронювання. Інтерфейс включає в себе ряд функціональних сторінок, які дозволяють здійснювати пошук маршрутів, переглядати доступні поїздки, бронювати квитки, керувати своїм обліковим записом та переглядати історію бронювання.

Під час розробки інтерфейс користувача запускався локально на порту 5000 за адресою *http://localhost:5000* за допомогою інструменту Vite.

Після запуску додатку на сайті http://localhost:5000 користувач потрапляє на головну сторінку (див. рис. 4.2), де знаходиться форма пошуку рейсів за містом відправлення, пунктом призначення, датою та кількістю пасажирів. Нижче знаходиться розділ з переліком найпопулярніших маршрутів для швидкого доступу.

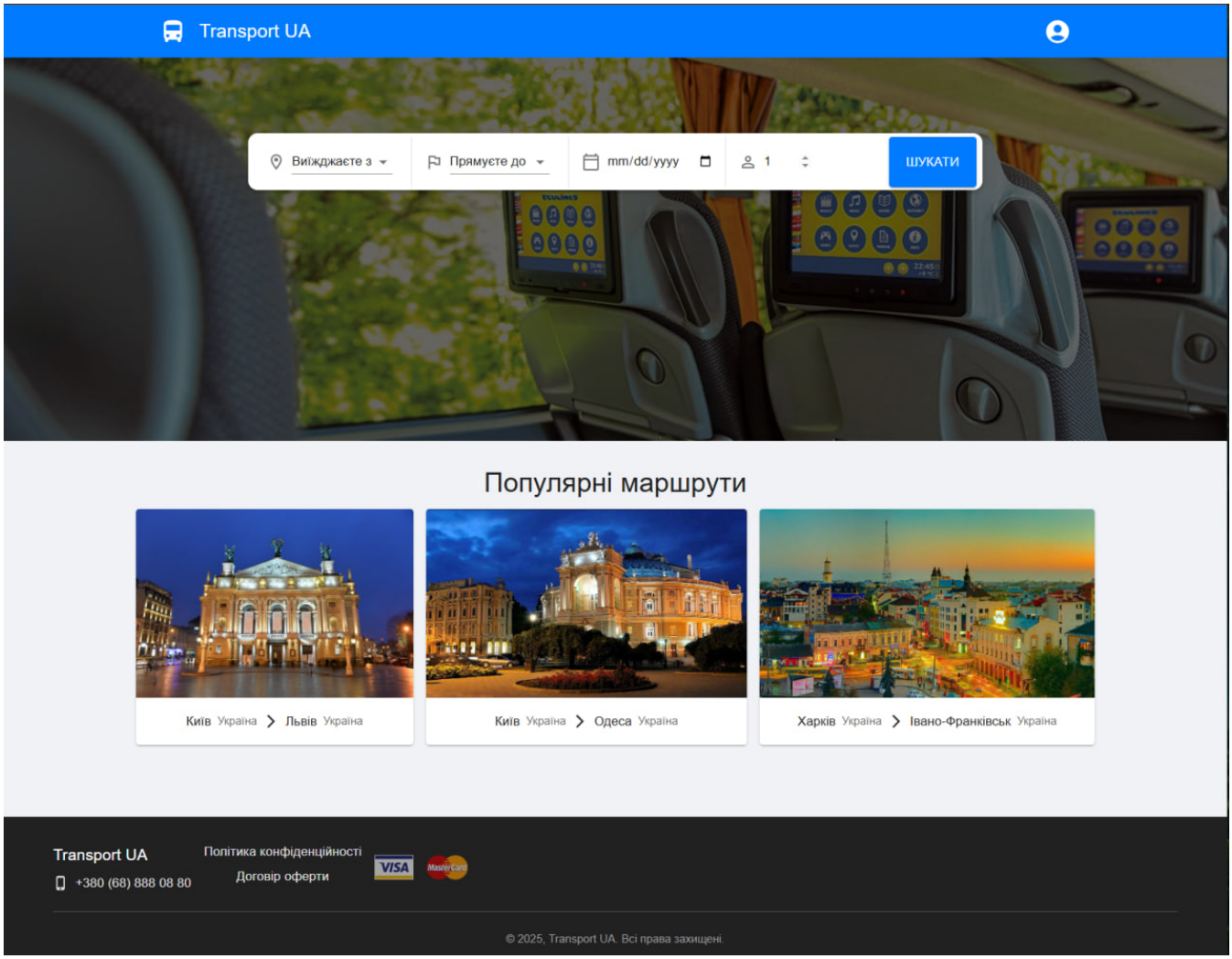


Рис. 4.2 Головна сторінка з пошуком і популярними маршрутами

При натисканні на іконку профілю відкривається діалогова форма входу або реєстрації. Форма реєстрації (див. рис. 4.3) вимагає введення email, паролю, підтвердження паролю, імені, прізвища та телефону. Після успішного заповнення користувачу надсилається лист активації.

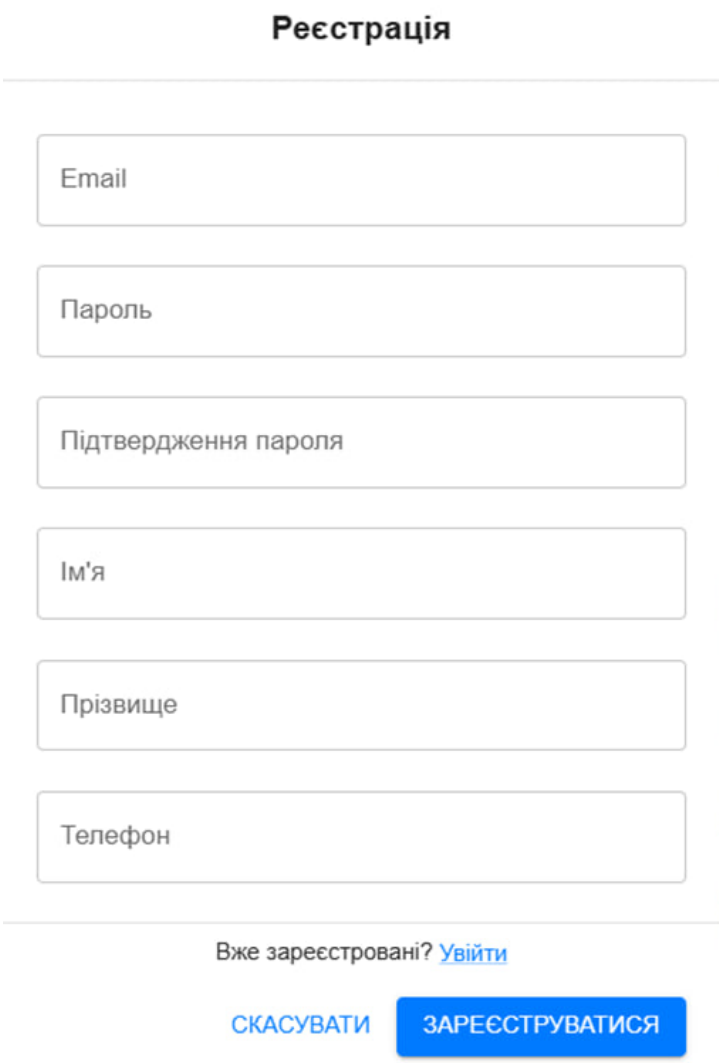


Рис. 4.3 Форма реєстрації користувача

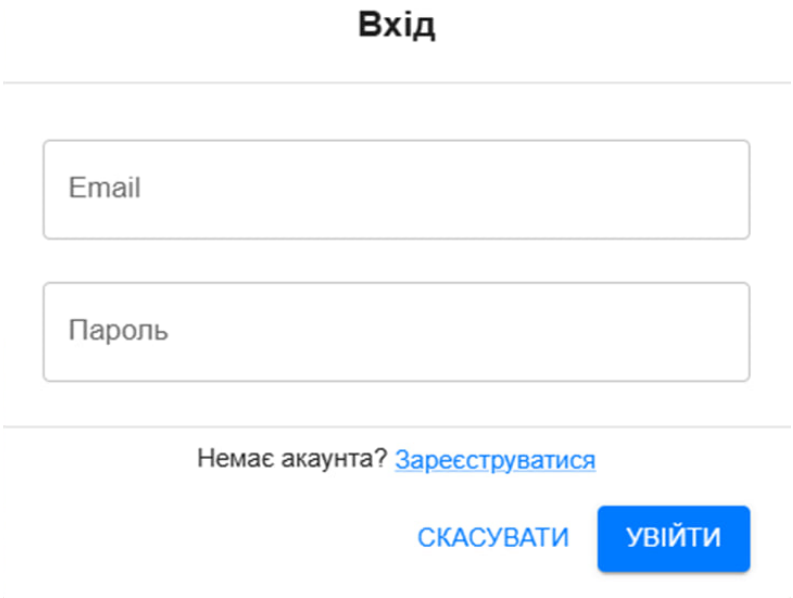


Рис. 4.4 Форма входу в акаунт

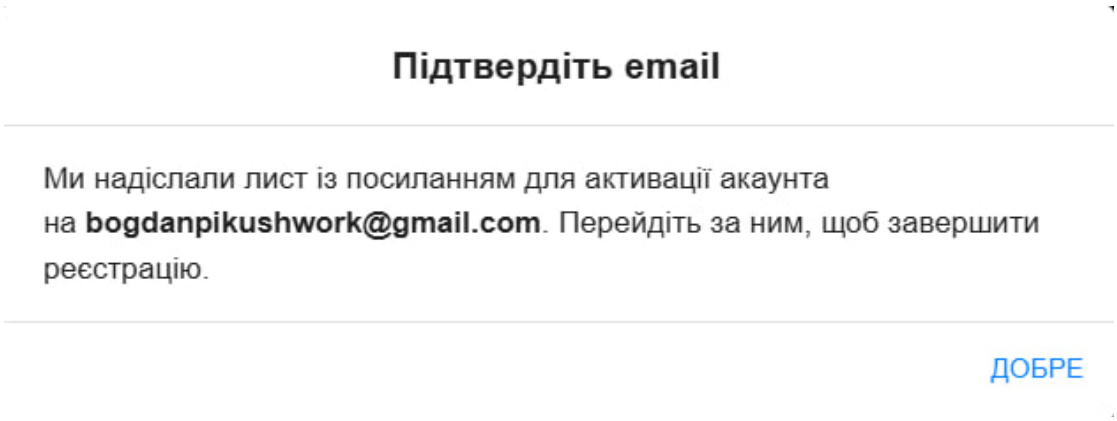


Рис. 4.5 Повідомлення про підтвердження email

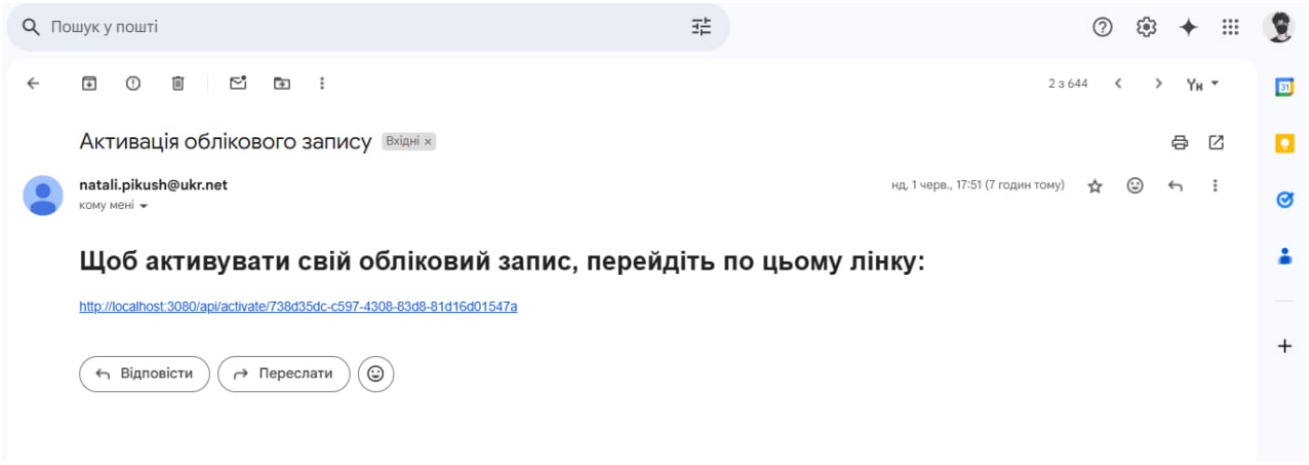


Рис. 4.6 Лист активації облікового запису на електронну пошту

На сторінці результатів пошуку (див. рисисунок 4.7) відображається список доступних поїздок з інформацією про час відправлення та прибуття, ціну, місце відправлення та залишок вільних місць. Для переходу до бронювання достатньо натиснути кнопку «Забронювати».

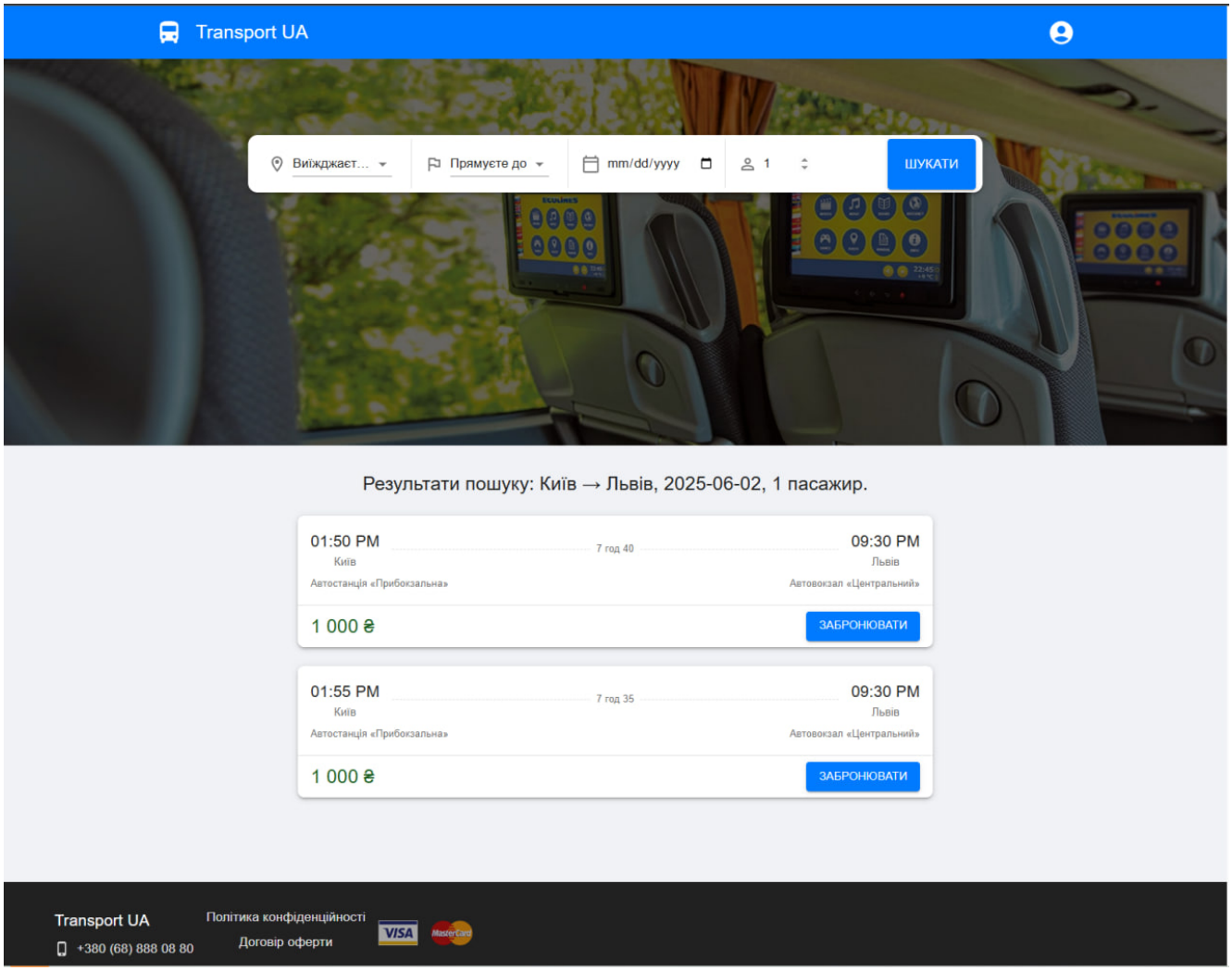


Рис. 4.7 Сторінка результатів пошуку

На сторінці бронювання (див. рисунок 4.8) користувач заповнює дані про пасажира, вибирає кількість багажу і підтверджує бронювання. Після успішного бронювання з'явиться повідомлення (див. рисунок 4.9) з номером квитка.

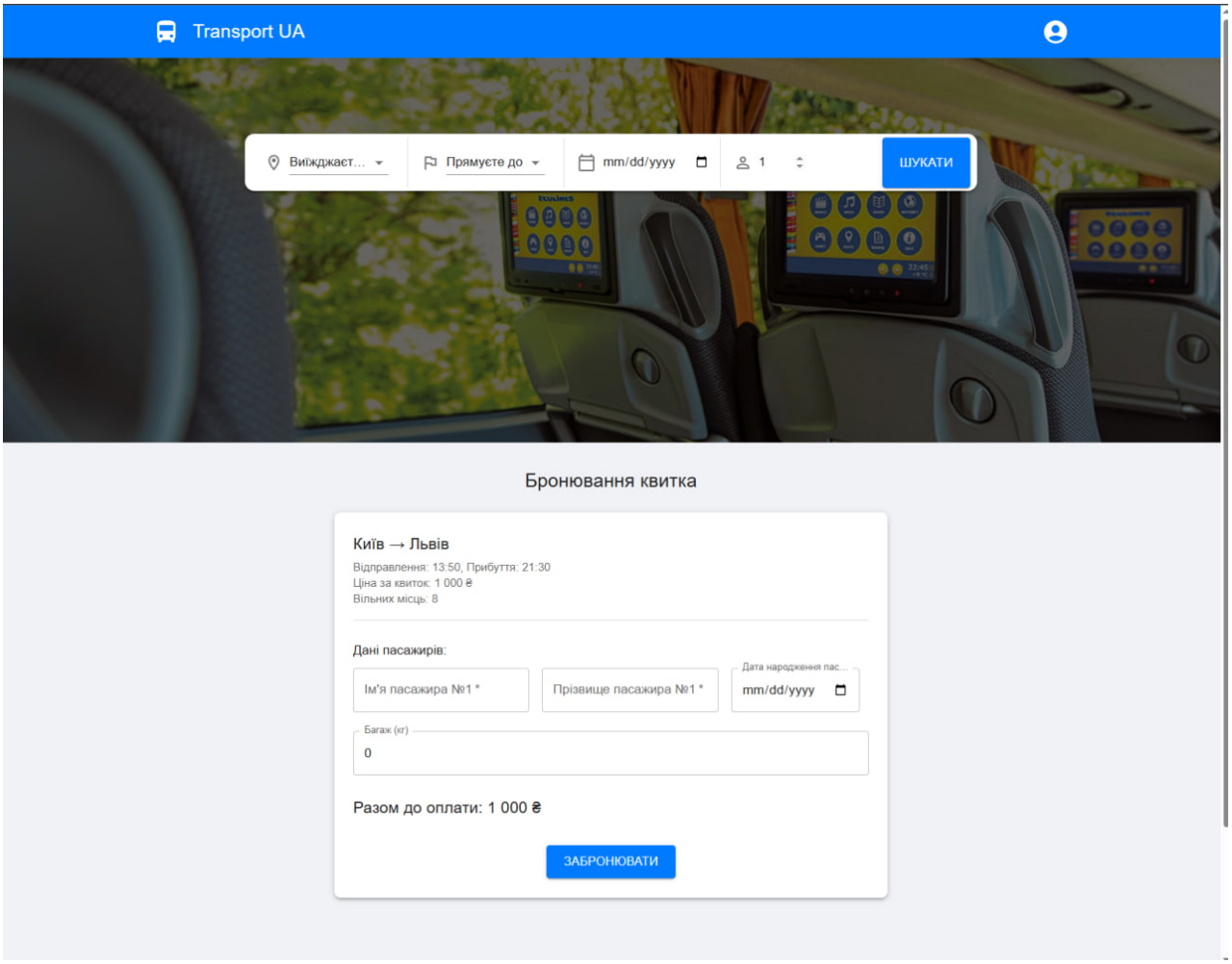


Рис. 4.8 Сторінка бронювання квитка

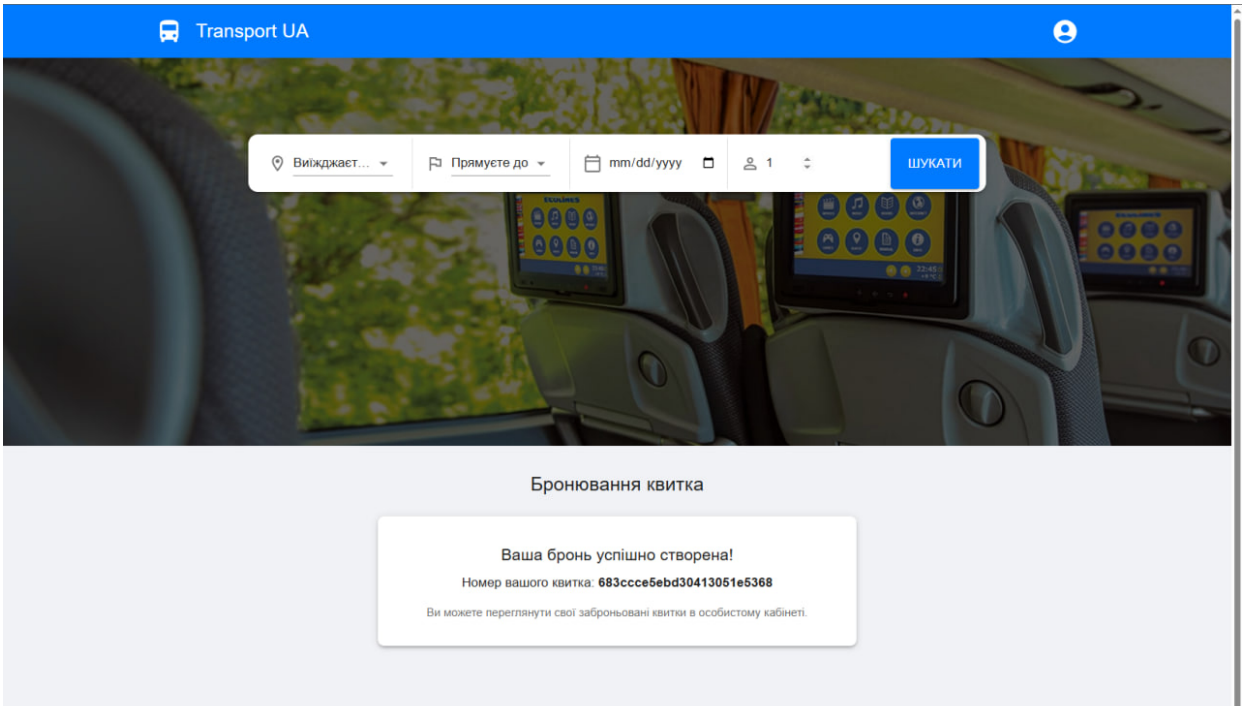


Рис. 4.9 Повідомлення про успішну бронь

У розділі "Мій профіль" (див. рисунок 4.10) користувач може переглянути особисту інформацію (ПІБ, email, телефон), а також список усіх заброньованих квитків, включаючи їх статус оплати, маршрут, дату та кількість пасажирів.

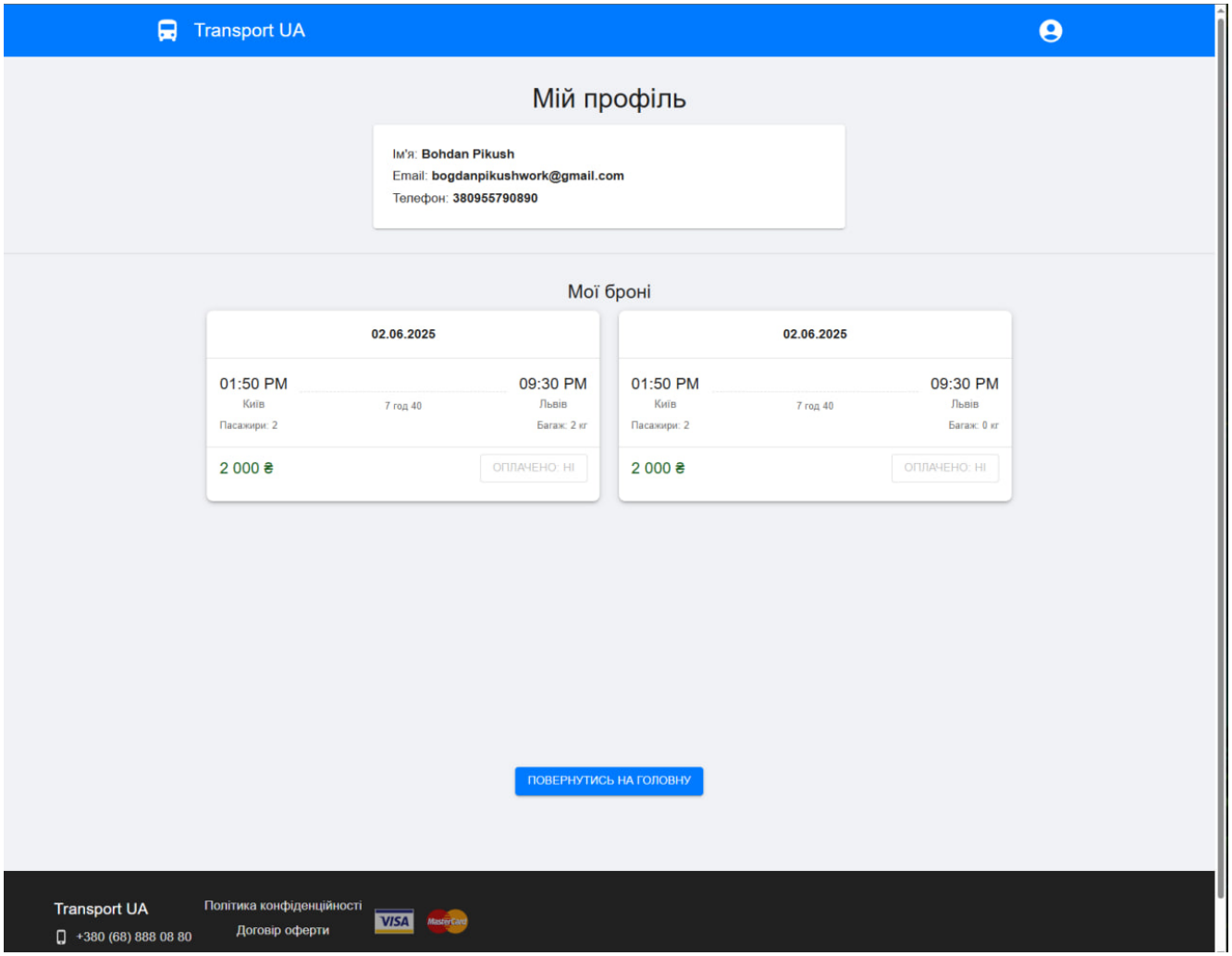


Рис. 4.10 Особистий кабінет користувача з історією бронювань

Інтерфейс користувача забезпечує зручну та зрозумілу взаємодію з системою бронювання. Всі необхідні дії - від пошуку рейсів до бронювання квитків та перегляду історії подорожей - виконуються послідовно та інтуїтивно зрозуміло. Це дозволяє користувачам швидко знаходити потрібну інформацію та здійснювати бронювання без зайвих кроків.

4.1.2 Інтерфейс адміністратора

Адміністративна частина веб-застосунку призначена для співробітників компанії, які керують рейсами, маршрутами та контролюють процес бронювання. Інтерфейс реалізований як окрема фронтенд-частина з власним набором сторінок та доступом через інший маршрут *http://localhost:5001*

Всі компоненти реалізовано з урахуванням потреб адміністратора: швидке введення даних, редагування записів, перегляд списків бронювань.

Адміністративна панель системи надає можливості для керування маршрутами, поїздками, а також перегляду історії бронювань. Вона доступна тільки після авторизації (див. рисунок 4.12).

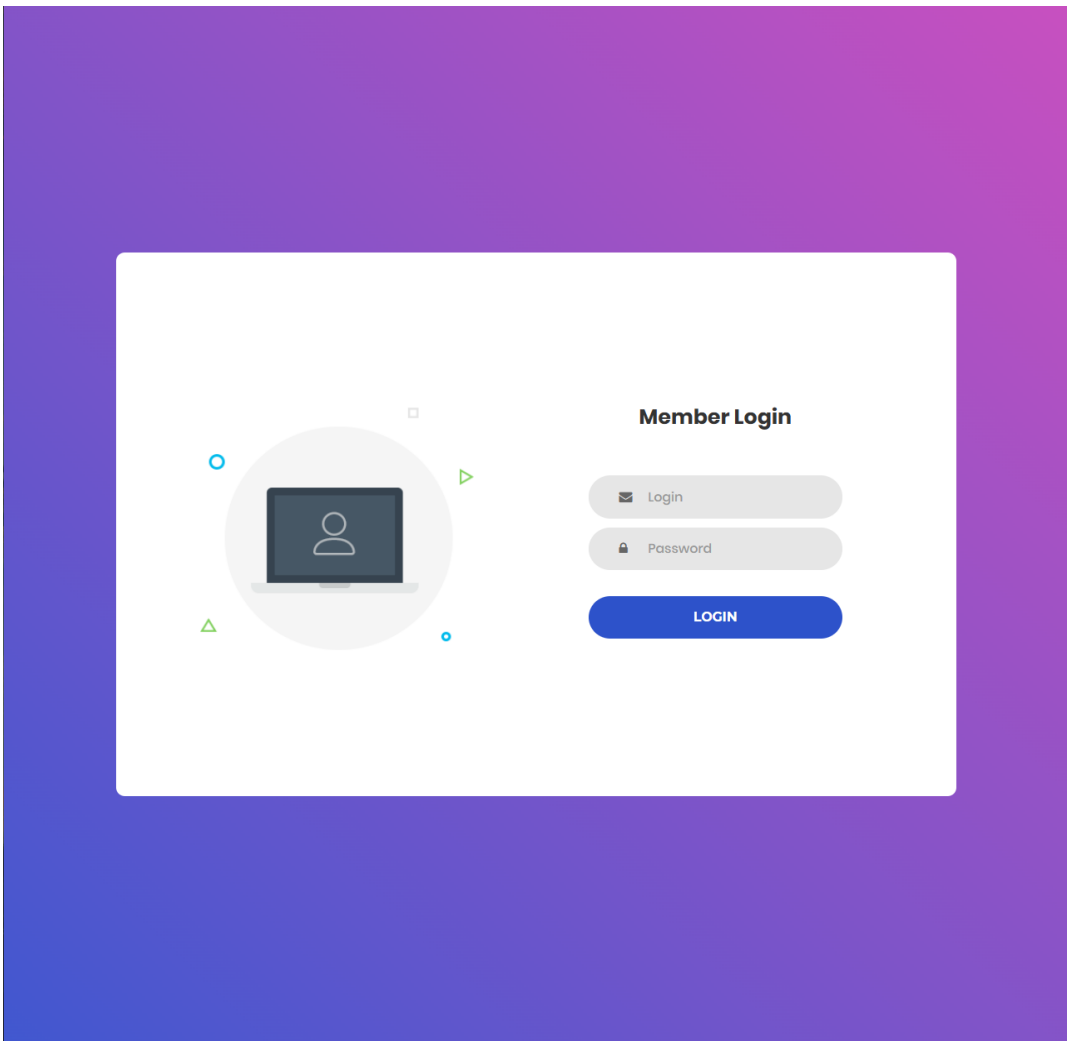


Рис. 4.12 Форма входу адміністратора

На головній сторінці після авторизації адміністратор бачить список створених маршрутів (див. рисунок 4.13) з можливістю їх редагування або додавання нових.

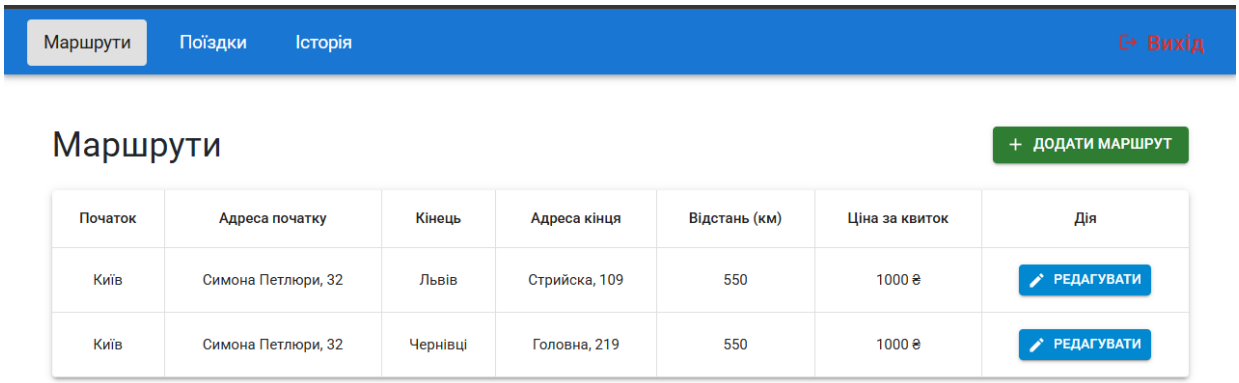


Рис. 4.13 Список маршрутів

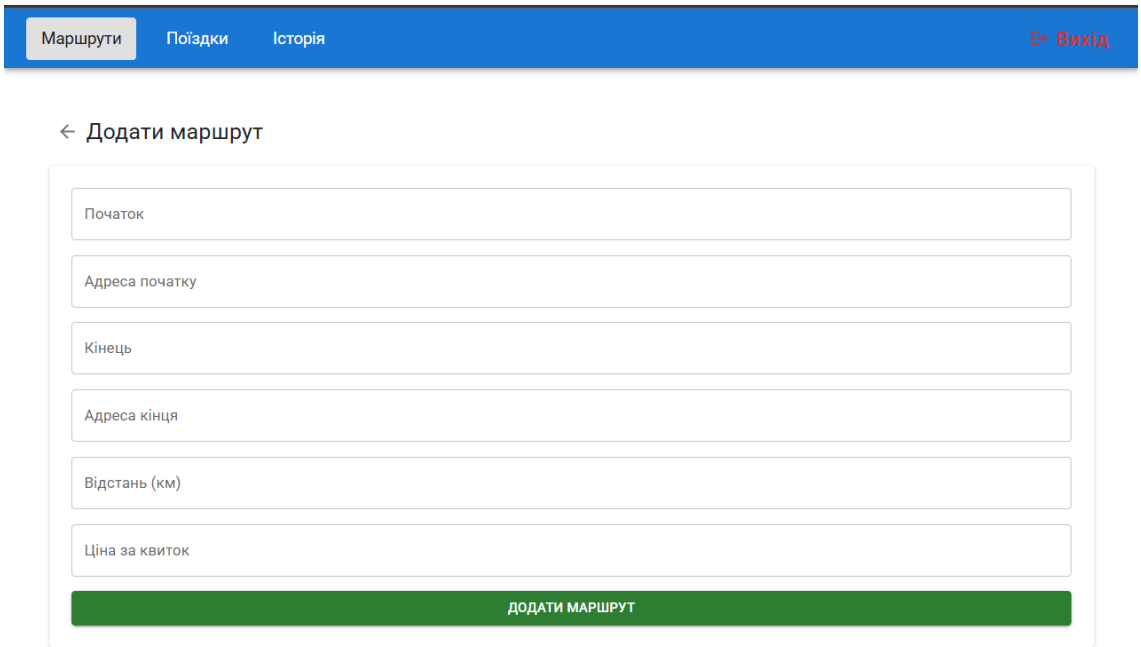


Рис. 4.14 Форма додавання маршруту

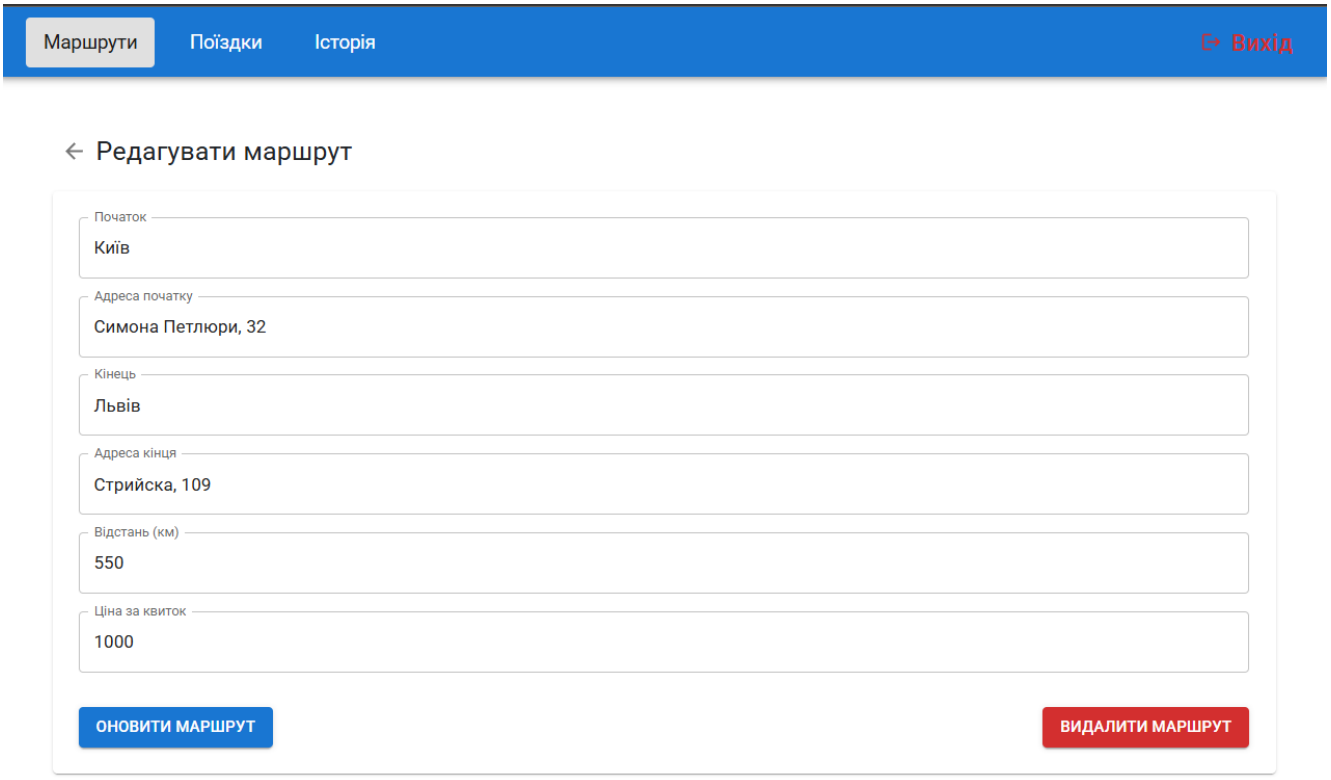


Рис. 4.15 Форма редагування маршруту

У розділі “Поїздки” (див. рисунок 4.16) адміністратор може створювати нові рейси за наявними маршрутами, вказуючи дату та час, кількість місць, а також активувати чи деактивувати поїздку.

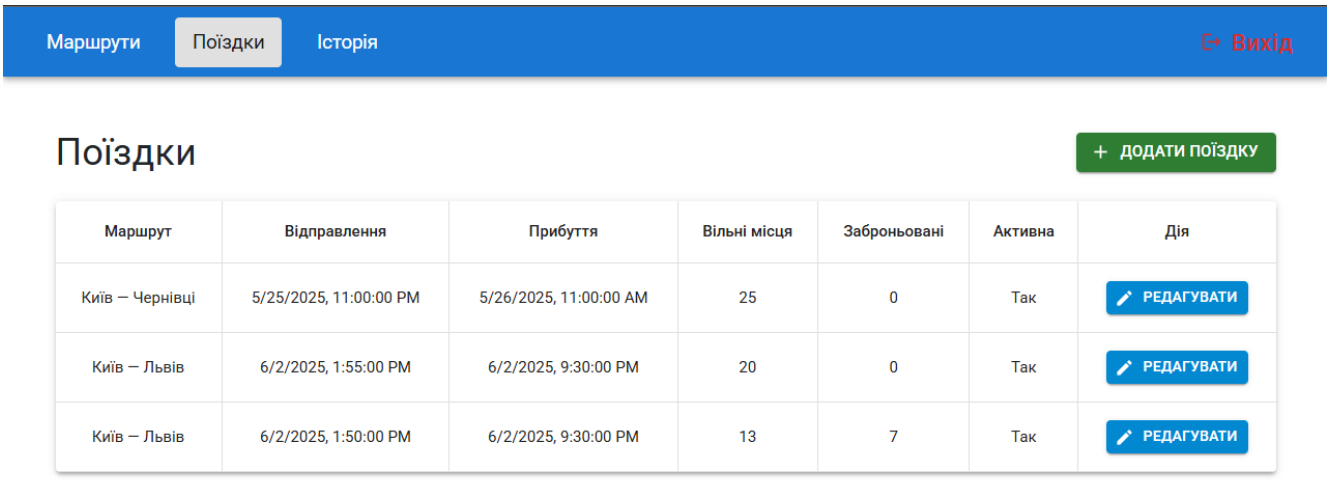


Рис. 4.16 Список поїздок

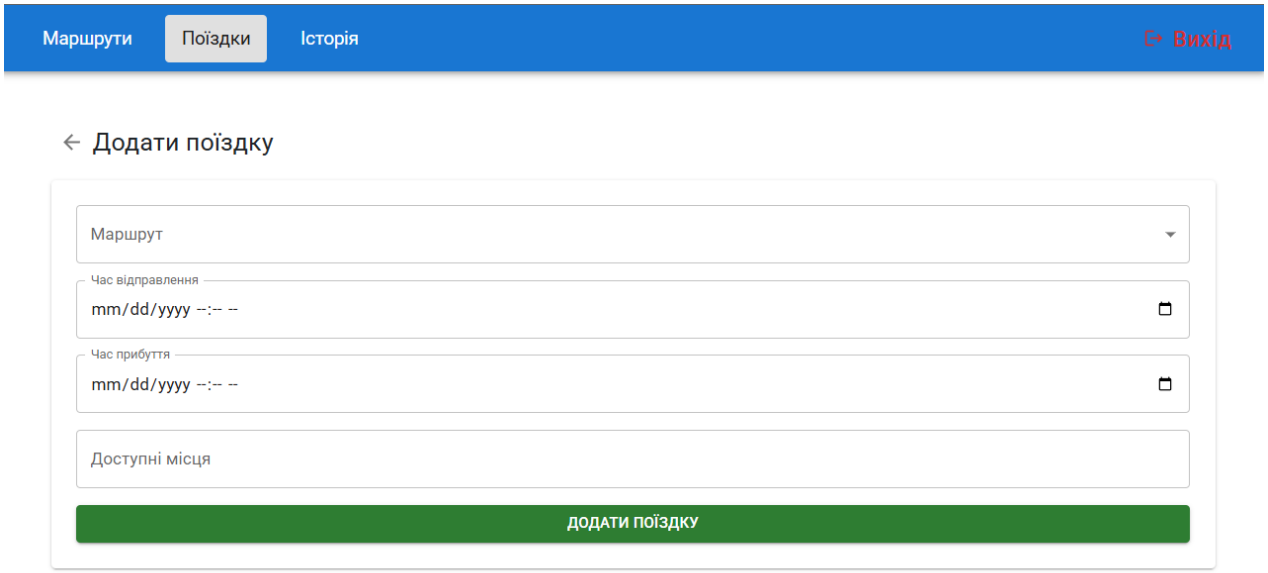


Рис. 4.17 Форма додавання поїздки

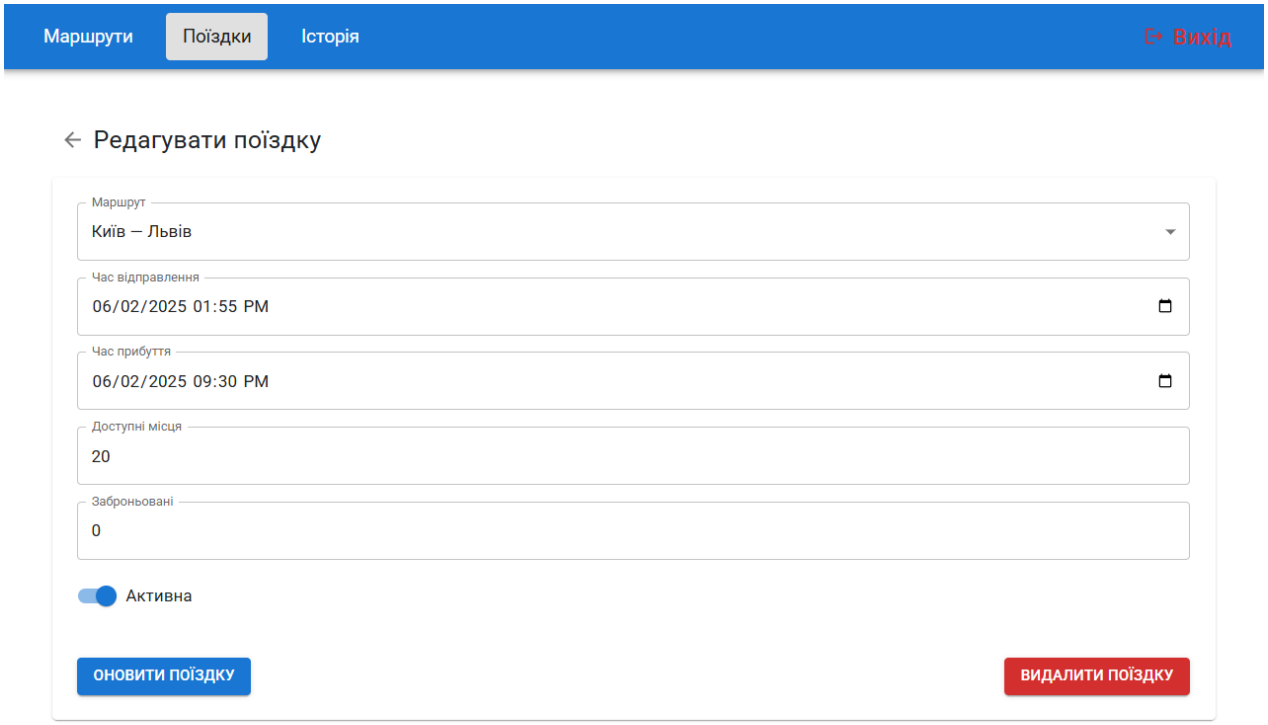


Рис. 4.18 Форма редагування поїздки

Вкладка “Історія” (див. рисунок 4.19) відображає усі заброньовані поїздки. Для кожного рейсу адміністратор може подивитися, хто саме забронював квитки, скільки їх було та які контактні дані користувача.

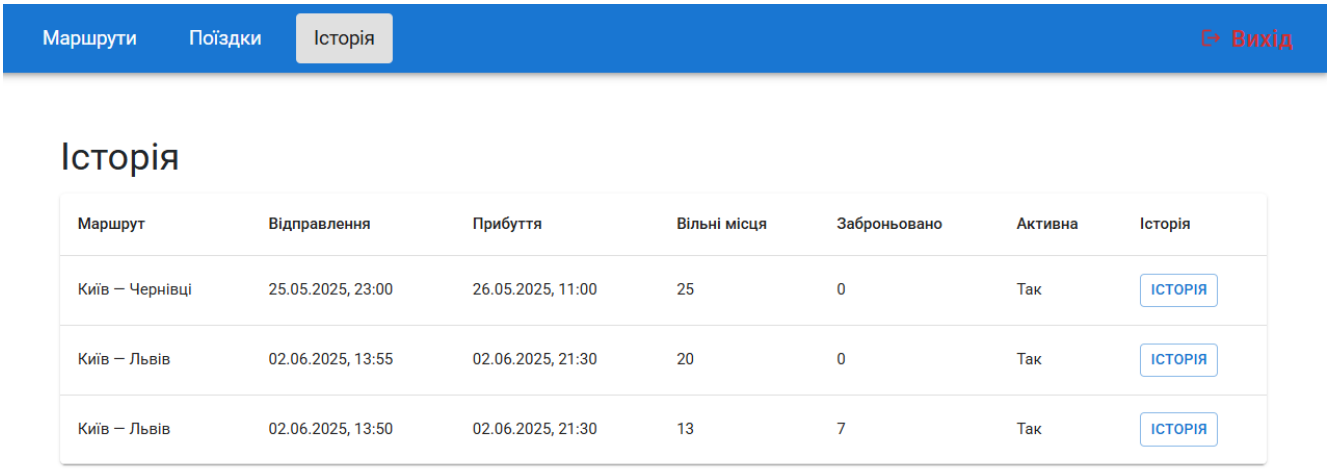


Рис. 4.19 Перелік поїздок з історією

Рис. 4.20 Деталі бронювань поїздки

Інтерфейс адміністратора дозволяє легко керувати всіма ключовими елементами системи бронювання - маршрутами, поїздками та історією замовлень. Завдяки простому та інтуїтивно зрозумілому дизайну адміністратори можуть швидко створювати або редагувати маршрути, контролювати навантаження та швидко реагувати на зміни.

Такий підхід дозволяє ефективно організувати робочий процес компанії та зменшити кількість помилок, пов'язаних з ручним введенням даних.

4.2 Реалізація серверної частини

4.2.1 Ініціалізація та налаштування проєкту

Для створення серверної частини веб-застосунку використовувалася платформа Node.js у поєднанні з фреймворком Express.js, який дозволяє швидко створювати REST API для обробки HTTP-запитів.

Ініціалізація проєкту виконувалась за допомогою наступних команд у терміналі:

npm init -y

npm install express mongoose dotenv cors jsonwebtoken bcrypt

npm install nodemon --save-dev

У результаті була створена структура серверной частини проєкту(див. рис. 4.21), яка включає такі основні каталоги:

– routes/ — визначення маршрутів API;

– controllers/ — логіка обробки запитів;

– services/ — бізнес-логіка;

– models/ — Mongoose-моделі для MongoDB;

– middlewares/ — проміжні обробники (валидація, авторизація);

– config/ — налаштування підключення до бази даних та змін середовища;

– tg-bot-for-errors/ — надсилання повідомлень про помилки у Telegram.

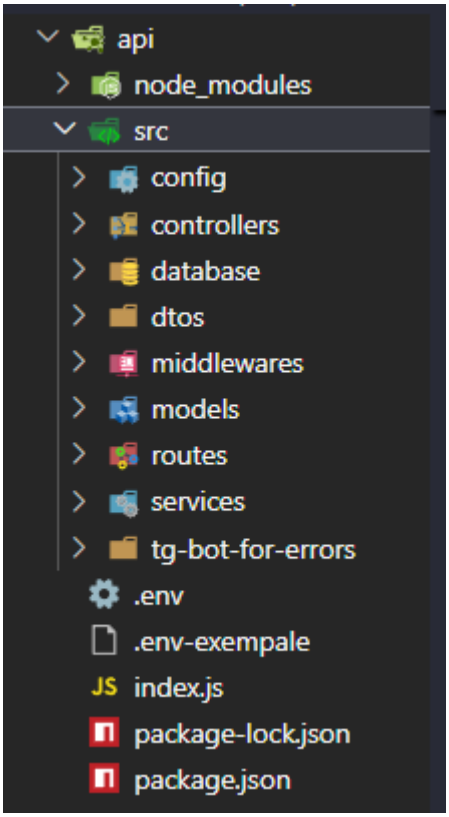


Рис. 4.21 Структура серверной частини проєкту

4.2.2 Безпека та запуск серверу

Серверний застосунок запускається з файлу index.js, який є точкою входу в проєкт. У цьому файлі налаштовано основні параметри безпеки та підключено необхідні middleware для захисту API:

– CORS – дозволяє контролювати доступ до API з інших доменів;

– Helmet – встановлює захисні HTTP-заголовки для запобігання поширеним вразливостям;

– Rate Limiter – обмежує кількість запитів від одного клієнта за певний проміжок часу, захищаючи сервер від перевантаження та DDoS-атак.

Запуск здійснювався за допомогою команди:

npm run dev

Сервер запускається локально на порту 3080 і забезпечує взаємодію з базою даних MongoDB, яка розміщена в хмарному середовищі MongoDB Atlas. Завдяки Atlas база працює безперервно, має високу доступність і дозволяє здійснювати

підключення з будь-якої точки з доступом до інтернету, що особливо зручно під час розробки та тестування застосунку.

4.2.3 Організація маршрутів та обробки запитів

Експрес-маршрутизатор використовується для маршрутизації запитів від сервера додатків. Всі маршрути розбиті на різні файли відповідно до їх призначення, що дозволяє структурувати логіку і полегшити обслуговування коду.

Основні маршрути знаходяться в каталозі routes/:

– auth-router.js - обробляє запити, пов'язані з реєстрацією, авторизацією та активацією облікового запису;

– adminAuth-router.js - окремий маршрутизатор для аутентифікації співробітників компанії

– user-router.js - дозволяє отримати особисту інформацію про користувачів;

– route-router.js - відповідає за створення, оновлення та перегляд маршрутів;

– trip-route.js - для управління маршрутизацією рейсів (маршрутів), включаючи час вильоту та прибуття, кількість місць;

– booking-route.js - дозволяє користувачам бронювати квитки.

Кожен маршрут імпортує відповідний контролер, який містить логіку обробки запиту. Наприклад, при запиті на /api/trips, викликається метод контролера, який звертається до сервісного рівня та бази даних.

4.2.4 Авторизація та захист доступу

Веб-застосунок використовує JWT-токени для авторизації.

Після успішного входу користувач отримує токен, який зберігається на фронтенді та надсилається у заголовках наступних запитів.

Захищені маршрути мають middleware для перевірки токена (authMiddleware.js).

4.2.5 Шифрування паролів та обробка помилок

При реєстрації паролі шифруються за допомогою bcrypt для забезпечення безпечного зберігання.

Крім того, система має глобальний обробник помилок і Telegram-бот, який повідомляє адміністратора про критичні помилки або винятки (tg-bot-for-errors/).

4.3 Завантаження проєкту на GitHub

Для резервного копіювання, спільної роботи та публічної презентації проекту було завантажено весь вихідний код веб-застосунку на платформу GitHub, яка є загальновживаною системою для зберігання та управління Git-репозиторіями.

Перед завантаженням створіть новий репозиторій на GitHub під назвою Diploma_2025 і виконайте наступні кроки в терміналі:

git init

git add .

git commit -m "Initial commit"

git branch -M main

git remote add origin https://github.com/BogdanPikush/Diploma_2025.git

git push -u origin main

Після виконання цих команд проект буде доступний у хмарному репозиторії GitHub, що спрощує використання проекту на будь-якому пристрої та забезпечує легку спільну роботу. Крім того, при використанні .gitignore до сховища не будуть включені сервіси або конфіденційні файли, такі як .env або node_modules.

4.4 Тестування застосунку

Після того, як клієнтська та серверна частини реалізації були завершені, було використано інструмент Postman для тестування запитів API. Мета тесту - перевірити коректність обробки HTTP-запиту, відповідність формату відповіді, валідність вхідних даних та коректність взаємодії з базою даних.

Тестовані ендпоінти:

1) Аутентифікація:

– POST /api/auth/registration - Реєстрація користувача.

– GET /api/auth/activate/:link - Активація облікового запису за допомогою електронної пошти.

– POST /api/auth/login - Логін користувача.

2) Маршрутизація:

– POST /api/routes/createRoute - Створення нового маршруту.

– GET /api/routes/allRoutes - Отримати всі маршрути.

– GET /api/routes/route/:id - Отримати маршрут за ідентифікатором.

– PUT /api/routes/editRoute/:id - Відредагувати маршрут.

– DELETE /api/routes/deleteRoute/:id - Видалити маршрут.

3) Рейси:

– POST /api/trips/createTrip - Створити рейс.

– GET /api/trips/allTrips - Переглянути всі поїздки.

– GET /api/trips/trip/:id - Переглянути конкретну поїздку.

– PUT /api/trips/editTrip/:id - Редагувати подорож.

– DELETE /api/trips/deleteTrip/:id - Видалити подорож.

– GET /api/trips/search - Пошук доступних рейсів.

Методи перевірки:

– Позитивні тести — перевірка правильного сценарію роботи (коректні дані → успішна відповідь);

– Негативні тести — введення некоректних даних (пусті поля, неправильний формат email, неіснуючі ID).

Результат - усі основні маршрути обробляються коректно, відповіді API мають очікуваний формат JSON. Після реєстрації надсилається лист активації на пошту. Помилки (наприклад, неавторизований доступ або відсутній ресурс) обробляються через глобальний обробник помилок та надсилаються до Telegram-бота.

ВИСНОВКИ

В ході дипломного проекту було реалізовано повнофункціональний веб-додаток для організації автобусних перевезень в Україні. Основною метою було створення системи, яка б дозволила користувачам бронювати квитки онлайн, а адміністраторам легко керувати маршрутами та рейсами.

Було виконано наступні задачі:

1) Проаналізовано предметну область - організація автобусних перевезень по Україні, визначення основних функціональних вимог до системи.

2) Сформульовано технічні вимоги до веб-застосунку, включно з ролями користувачів (пасажир, адміністратор), функціоналом пошуку, бронювання, реєстрації, керування маршрутами та поїздками.

3) Розроблено архітектуру застосунку, включно з:

– клієнтською частиною (React.js, MUI),

– серверною частиною (Node.js, Express.js),

– базою даних (MongoDB Atlas).

4) Реалізовано користувацький інтерфейс, який дозволяє пасажирам:

– шукати маршрути,

– реєструватися й авторизовуватися,

– бронювати квитки,

– переглядати історію поїздок.

5) Розроблено адміністративний інтерфейс для керування маршрутами, поїздками та перегляду історії бронювань.

6) Створено REST API, що забезпечує обробку запитів до системи, з поділом за відповідальностями (routes, controllers, services, middlewares).

7) Забезпечено захист API за допомогою:

– JWT-авторизації,

– CORS,

– Helmet,

– Rate-limiter.

8) Інтегровано MongoDB як основну базу даних для зберігання інформації про користувачів, маршрути, поїздки та бронювання.

9) Налаштовано систему сповіщень про помилки через Telegram-бота для швидкого реагування.

10) Виконано тестування функціональності API за допомогою Postman.

11) Завантажено проєкт на GitHub, що дозволяє здійснювати спільну розробку та підтримку в майбутньому.

Розроблений застосунок відповідає сучасним вимогам до веб-систем: він зручний у користуванні, безпечний і легко розширюється. У майбутньому проєкт може бути доповнений модулями для онлайн-оплати, інтеграцією з мобільними додатками, а також багатомовною підтримкою.

Таким чином, поставлені в дипломній роботі завдання були успішно виконані, а мета — досягнута.

Робота пройшла апробацію. За її результатами було опубліковано наступні тези доповідей:

1. Пікуш Б.Ю., Калинюк Ю.С. Інтелектуальні обчислення на краю мережі: тенденції та перспективи використання в Україні. IV Міжнародна науково-технічна конференція «Безпека інформаційних технологій» (IT Sec - 2025), 23 травня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Поданно до друку.

2. Пікуш Б.Ю., Калинюк Ю.С. Сучасні інтелектуальні технології в організації автобусних подорожей. IV Міжнародна науково-технічна конференція «Безпека інформаційних технологій» (IT Sec - 2025), 23 травня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Поданно до друку.

ПЕРЕЛІК ПОСИЛАНЬ

1. Banks A., Porcello E. Learning React: Modern Patterns for Developing React Apps. 3rd ed. Sebastopol, CA: O’Reilly Media, 2023. 350 p.

2. Express.js Guide. Express 5 Documentation URL: https://expressjs.com/en/5x/api.html (дата звернення: 25.05.2025).

3. Flanagan D. JavaScript: The Definitive Guide. 7th ed. Sebastopol, CA: O’Reilly Media, 2020. 706 p.

4. GitHub Docs. Introduction to Git and GitHub. URL: https://docs.github.com/ (дата звернення: 25.05.2025).

5. Material UI Documentation. URL: https://mui.com/ (дата звернення: 25.05.2025).

6. MongoDB Manual. Version 6.0 URL: https://www.mongodb.com/docs/manual (дата звернення: 25.05.2025).

7. Node.js v20 Documentation. Node.js Foundation. URL: https://nodejs.org/en/docs/ (дата звернення: 25.05.2025).

8. React Documentation. Meta Open Source. URL: https://react.dev/ (дата звернення: 25.05.2025).

9. Сидоренко І. М. Програмування на JavaScript: навчальний посібник. Тернопіль: ТНТУ, 2020. 156 с.

10. Чеботарьов А. О. MongoDB у веб-розробці: концепція та використання. Київ: КНУ, 2022. 132 с.

11. Fain Y., Moiseev A. Full Stack Development with Node.js and MongoDB. 2nd ed. Birmingham: Packt Publishing, 2023. 450 p.

12. Vite Documentation. Modern Frontend Tooling. URL: https://vitejs.dev/guide (дата звернення: 25.05.2025).

13. MDN Web Docs. JavaScript Reference and Guides Mozilla Developer Network. URL: https://developer.mozilla.org/ (дата звернення: 25.05.2025).

14. Gospodarek M. JWT Authentication Best Practices. Auth0 Blog. URL: https://auth0.com/blog/jwt-best-practices (дата звернення: 25.05.2025).

15. Tirado J. REST API Design Rulebook. 2nd ed. O’Reilly Media, 2023. 196 p.

16. MongoDB Atlas Documentation URL: https://www.mongodb.com/docs/atlas (дата звернення: 25.05.2025).

17. Яворський С. О. Основи клієнт-серверної розробки: навчальний посібник. Львів: ЛНУ, 2021. 172 с.

18. Білоус О. І. Веб-програмування з JavaScript та Node.js. Київ: КНЕУ, 2020. 144 с.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка програмного забезпечення для організації подорожей автобусом

Виконав студент 4 курсу
групи ПД-43
Пікуш Богдан Юрійович
Керівник роботи
асистент кафедри ІПЗ Калинюк Богдан Сергійович

Київ-2025

МЕТА, ОБ’ЄКТА ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: Розширення функціональності системи організації автобусних поїздок шляхом розробки веб-застосунку з використанням сучасних технологій розробки програмного забезпечення.

Об'єкт дослідження: Процес організації автобусних перевезень.

Предмет дослідження: Методи, інструменти та технології розробки веб-застосунку для бронювання автобусних поїздок.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати предметну область.
2. Сформулювати технічні вимоги до веб-застосунку.
3. Розробити архітектуру застосунку клієнтської частини та серверної.
4. Створити REST API, що забезпечить обробку запитів та забезпечить захист API (Helmet, CORS, rate-limit).
5. Реалізувати автентифікацію через JWT з підтвердженням e-mail та ролями «Пасажир», «Адміністратор».
6. Створити клієнтський SPA-інтерфейс (React + MUI) для публічного сайту та окрему адмін-панель.
7. Виконати тестування функціональності API.

АНАЛІЗ АНАЛОГІВ

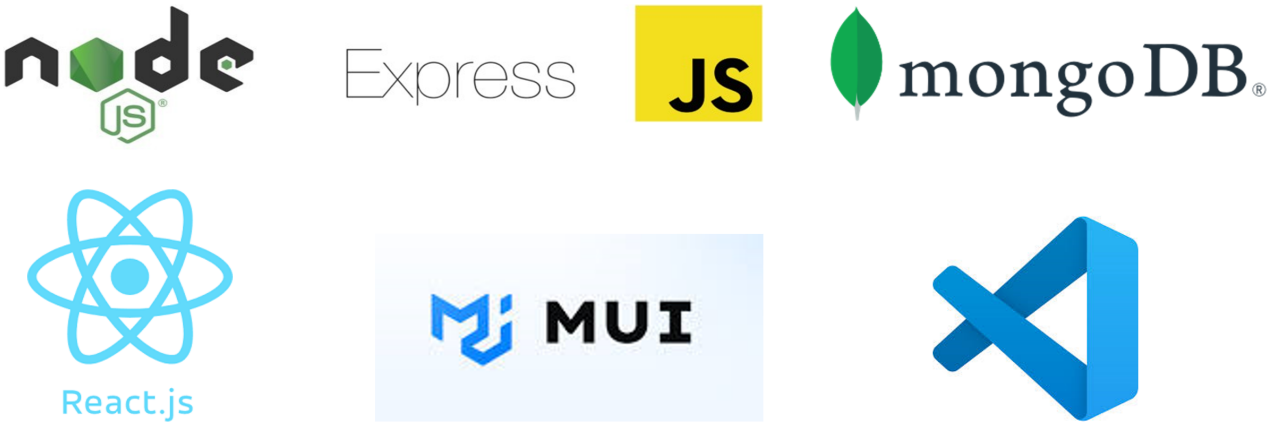
Показник	Tickets.ua	FlixBus	BlaBlaCar Bus
Наявність веб-платформи	+	+	+
Наявність мобільного додатку	+	+	+
Відкритий API	-	Частково (тільки для партнерів)	-
Можливість адміністрування маршрутів	-	-	-
Міжнародне покриття	+	+	+
Простота бронювання	+	+	+
Основна цільова аудиторія	Користувачі з України, які планують подорожі різними видами транспорту	Пасажири, що подорожують на далекі відстані по Європі та США	Молоді мандрівники та користувачі мобільних сервісів

ВИМОГИ ДО ЗАСТОСУНКУ

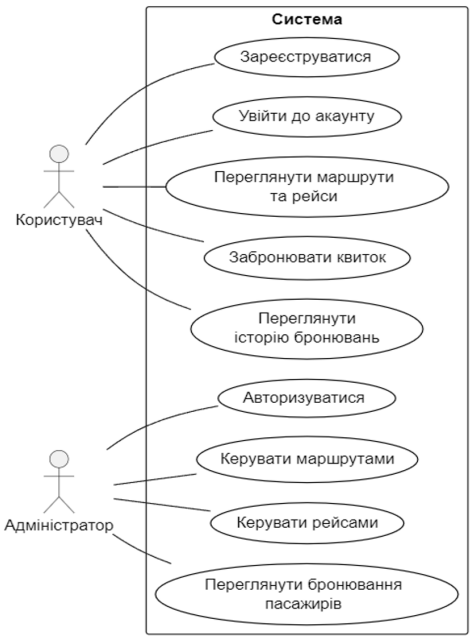
Функціональні вимоги:
- Реєстрація та автентифікація користувачів.
- Перегляд маршрутів і розкладу.
- Бронювання квитків.
- Перегляд історій бронювань.
- Адміністрування маршрутів.

Нефункціональні вимоги:
- Інтуїтивно зрозумілий інтерфейс.
- Безпека (шифрування, надійна автентифікація).
- Масштабованість системи.
- Стабільна робота застосунку.

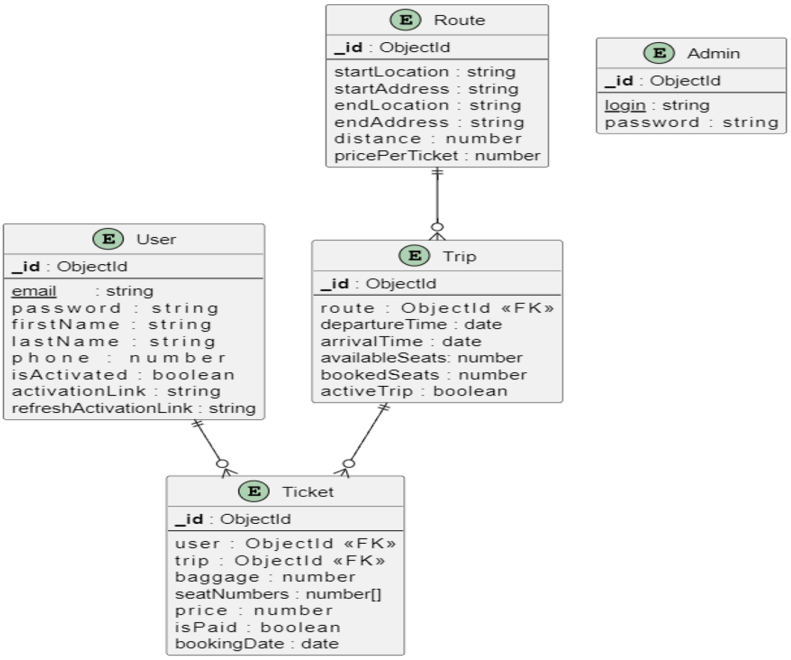
ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ



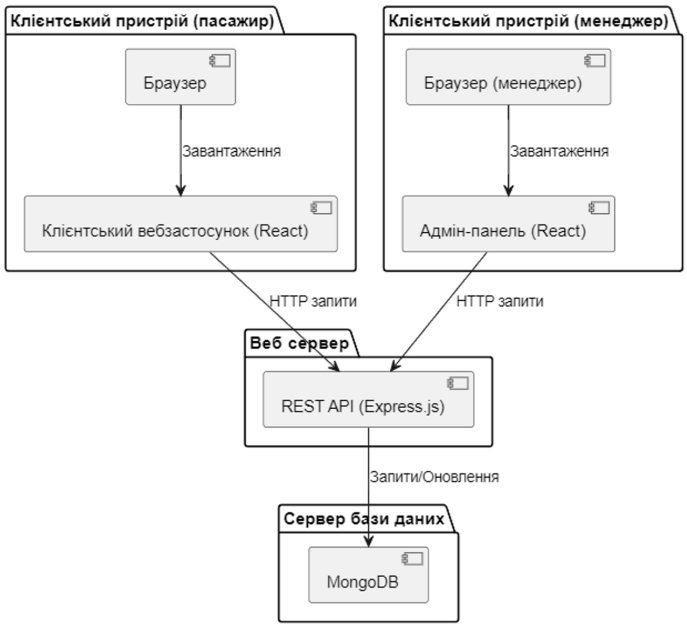
ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



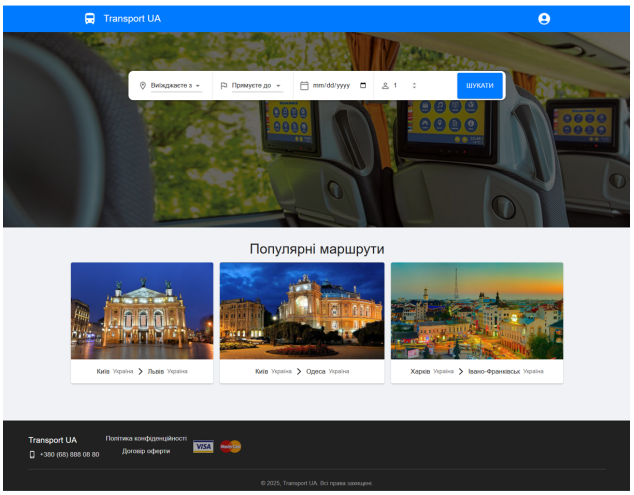
ДІАГРАМА СТРУКТУРИ БАЗИ ДАННИХ



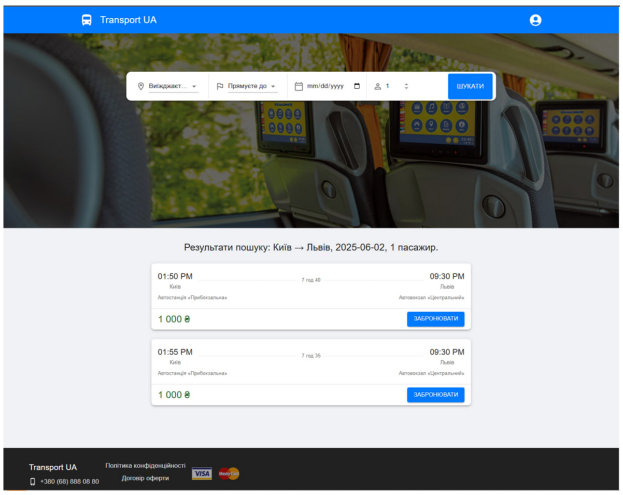
ДІАГРАМА РОЗГОРТАННЯ КЛІЄНТ-СЕРВЕРНОЇ МОДЕЛІ



ЕКРАННІ ФОРМИ

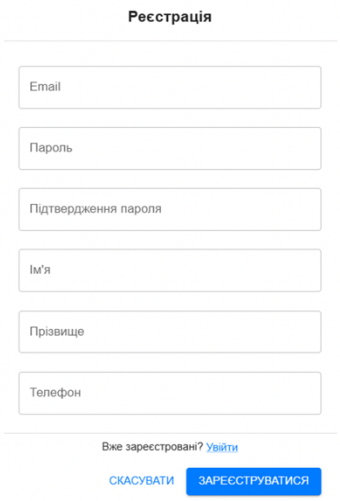


Головна сторінка

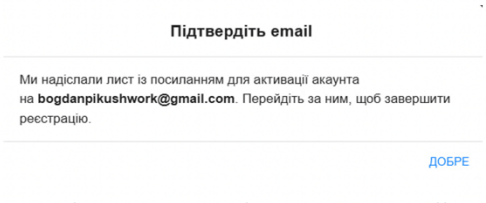


Сторінка результатів пошуку

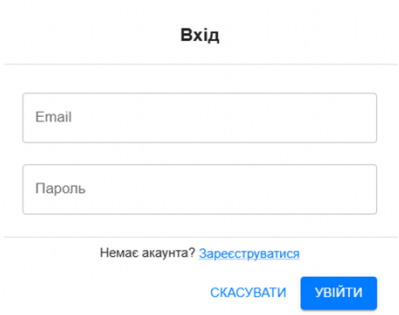
ЕКРАННІ ФОРМИ



Реєстрація

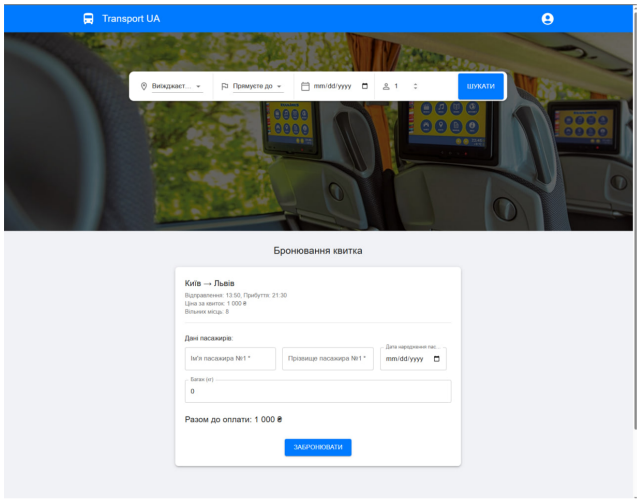


Повідомлення про підтвердження email

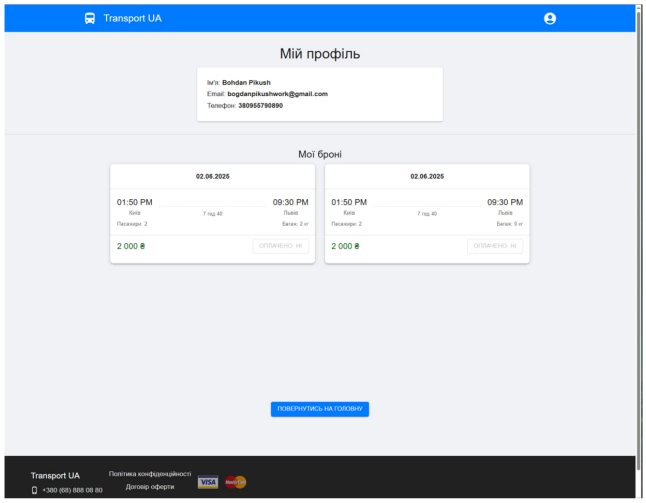


Вхід

ЕКРАННІ ФОРМИ

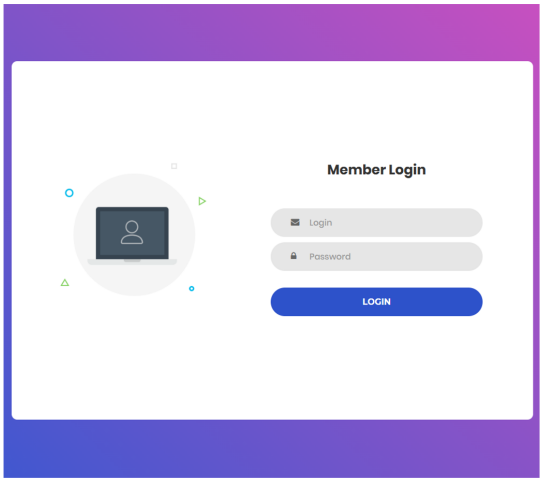


Сторінка бронювання квитка

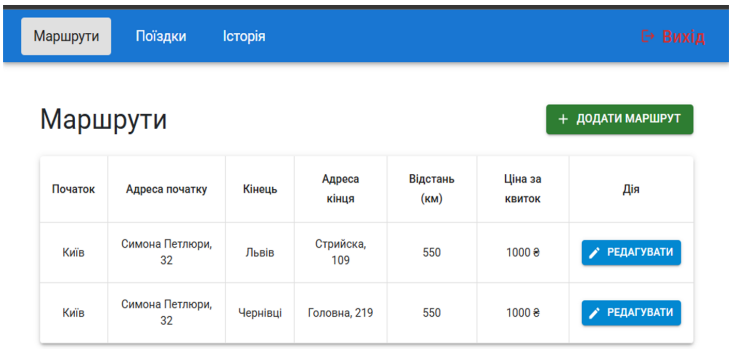


Особистий кабінет користувача

ЕКРАННІ ФОРМИ

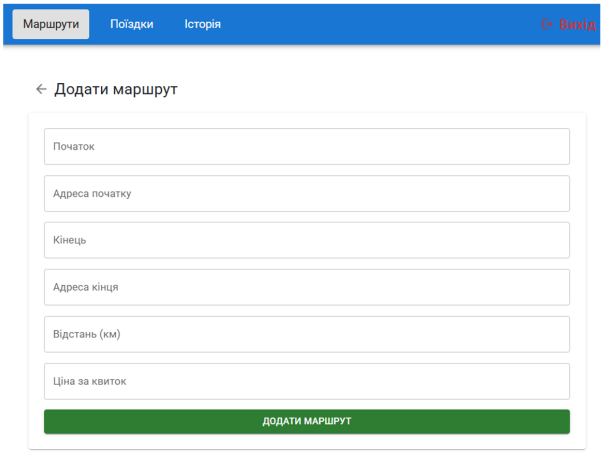


Вхід в систему

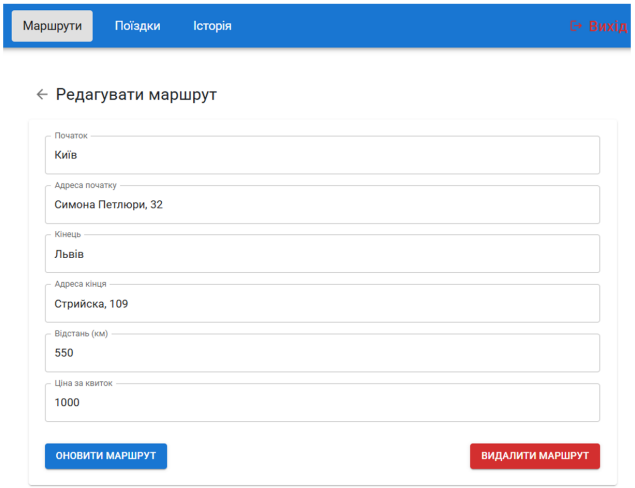


Сторінка Маршрутів

ЕКРАННІ ФОРМИ

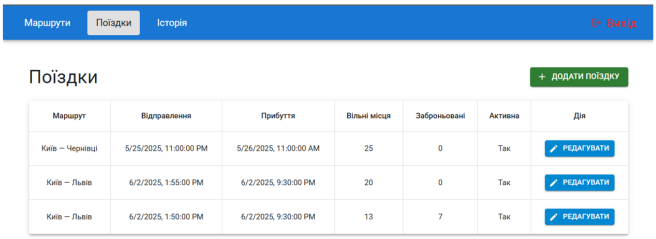


Форма додавання маршруту

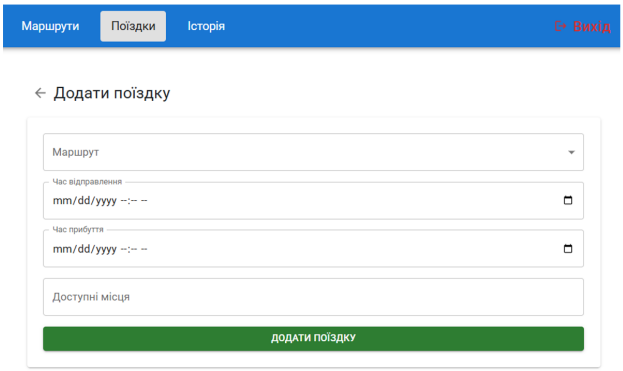


Форма редагування маршруту

ЕКРАННІ ФОРМИ



Список поїздок



Форма додавання поїздки

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Пікуш Б.Ю., Калинюк Ю.С. Інтелектуальні обчислення на краю мережі: тенденції та перспективи використання в Україні. IV Міжнародна науково-технічна конференція «Безпека інформаційних технологій» (IT Sec - 2025), 23 травня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Подано до друку.

2. Пікуш Б.Ю., Калинюк Ю.С. Сучасні інтелектуальні технології в організації автобусних подорожей. IV Міжнародна науково-технічна конференція «Безпека інформаційних технологій» (IT Sec - 2025), 23 травня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Подано до друку.

ВИСНОВКИ

1. Проаналізовано предметну область - організація автобусних перевезень по Україні, визначення основних функціональних вимог до системи.
2. Сформульовано технічні вимоги до веб-застосунку, включно з ролями користувачів (пасажир, адміністратор), функціоналом пошуку, бронювання, реєстрації, керування маршрутами та поїздками.
3. Розроблено архітектуру застосунку, включно з клієнтською частиною та серверною.
4. Розроблено REST-API на Node.js/Express із автентифікацією JWT, підтвердженням e-mail і розподілом ролей «Пасажир / Адміністратор», що гарантує безпечний доступ до сервісів.
5. Створено SPA-клієнт на React + MUI та окрему адмін-панель, які забезпечують інтуїтивно зрозумілий інтерфейс.
6. Реалізовано захист API (Helmet, CORS, rate-limit), що підвищує надійність і стійкість системи.
7. Проведене тестування функціональності API за допомогою Postman, застосунок сформульований вимогам і забезпечив стабільну роботу.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

Вхідний код файлу React.js

Для веб-сайту:

```
import React, { useState, useEffect } from "react";
import { BrowserRouter, Routes, Route, useLocation } from "react-router-dom";
import Header from "./components/Header";
import HomePage from "./pages/HomePage";
import SearchResults from "./pages/SearchResults";
import BookingPage from "./pages/BookingPage";
import ProfilePage from "./pages/ProfilePage";
import AuthDialog from "./pages/AuthDialog";

function AppRoutes() {
 const location = useLocation();
 const [authOpen, setAuthOpen] = useState(false);

 useEffect(() => {
  const token = localStorage.getItem("accessToken");
  if (!token && (location.pathname.startsWith("/profile") || location.pathname.startsWith("/booking"))) {
   setAuthOpen(true);
  }
 }, [location]);

 return (
  <>
   <Header onRequireAuth={() => setAuthOpen(true)} />

   <Routes>
    <Route path="/" element={<HomePage />} />
    <Route path="/search" element={<SearchResults />} />
    <Route path="/booking/:tripId" element={<BookingPage />} />
    <Route path="/profile" element={<ProfilePage />} />
   </Routes>

   <AuthDialog open={authOpen} onClose={() => setAuthOpen(false)} />
  </>
 );
}

export default function App() {
 return (
  <BrowserRouter>
   <AppRoutes />
  </BrowserRouter>
 );
}
```

Для адмін панелі:

```
import React from 'react'
import { Routes, Route, Navigate } from 'react-router-dom'
import AdminLayout from './layouts/AdminLayout'
import LoginContainer from './pages/login/LoginContainer'
// route
import RoutePage from './pages/route/RoutePage'
import AddRoutePage   from './pages/route/AddRoutePage.jsx'
import EditRoutePage from './pages/route/EditRoutePage.jsx'
// trip
import TripPage from './pages/trip/TripPage.jsx'
import AddTripPage from './pages/trip/AddTripPage.jsx'
```

```
import EditTripPage from './pages/trip/EditTripPage.jsx'
//history
import HistoryPage from "./pages/HistoryPage.jsx"

export default function App() {
  return (
    <Routes>
      <Route path="/" element={<Navigate to="/login" replace />} />
      <Route path="/login" element={<LoginContainer />} />

      <Route
        path="/admin/*"
        element={
          <AdminLayout>
            <Routes>
              <Route path="route" element={<RoutePage />} />
              <Route path="route/add"   element={<AddRoutePage/>} />
              <Route path="route/edit/:id" element={<EditRoutePage />} />

              <Route path="trip" element={<TripPage /> } />
              <Route path="trip/add" element={<AddTripPage /> } />
              <Route path="trip/edit/:id" element={<EditTripPage /> } />

              <Route path="tickets" element={<HistoryPage />} />
              <Route path="" element={<Navigate to="route" replace />} />
            </Routes>
          </AdminLayout>
        }
      />

      <Route path="*" element={<Navigate to="/login" replace />} />
    </Routes>
  )
}
```

Вхідний код файлу Node.js

index.js:

```
import { config } from './src/config/config.js';

import express from "express";
import cookieParser from "cookie-parser";

// lib for security
import cors from 'cors';
import rateLimit from 'express-rate-limit';
import helmet from 'helmet';

// routers api
import authRouter from "./src/routes/auth-router.js";
import routeRouter from "./src/routes/route-router.js";
import tripRouter from './src/routes/trip-route.js';

// routers admin
import adminAuthRouter from './src/routes/adminAuth-router.js';

// connect to db
import connectDB from "./src/database/db.js";
import bookingRouter from './src/routes/booking-route.js';
import userRouter from './src/routes/user-router.js';
connectDB();
```

```
const app = express();

app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// Setting CORS
const allowedOrigins = [
  'http://localhost:5000',  // your website
  'http://localhost:5001',  // your admin panel
];

const corsOptions = {
  origin: (incomingOrigin, callback) => {
    // allow requests with no origin (like mobile apps, curl, etc.)
    if (!incomingOrigin) return callback(null, true);

    if (allowedOrigins.includes(incomingOrigin)) {
      // origin is in our whitelist
      callback(null, true);
    } else {
      callback(new Error(`Origin ${incomingOrigin} not allowed by CORS`));
    }
  },
  methods: ['GET','POST','PUT','DELETE','OPTIONS'],
  credentials: true,
  exposedHeaders: ['Authorization']
};

app.use(cors(corsOptions));

app.use(cookieParser());

// api routes
app.use("/api", authRouter);
app.use("/api", routeRouter);
app.use("/api", tripRouter);
app.use("/api", bookingRouter);
app.use("/api", userRouter);

// admin routes
app.use("/admin", adminAuthRouter);

// Setting Rate Limiter
const limiter = rateLimit({
  windowMs: 1 * 60 * 1000, // 1 min
  max: 100, // max 100 requests for 1 IP
  message: 'Too many requests, please try again later.',
});
app.use(limiter);

// Setting Helmet
app.use(helmet());

app.listen(config.PORT, () => {
  console.log(`Server started in port - ${config.PORT}`);
});
```