МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ

Навчально-науковий інститут кібернетики,
інформаційних технологій та інженерії

Кафедра комп'ютерних наук та прикладної математики

"До захисту допущена"
Зав. кафедри комп'ютерних наук та
прикладної математики
д.т.н., проф. Ю.В. Турбал
«____» _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

Проектування та розробка веб-додатку для пошуку спільних поїздок на основі технології ASP.NET MVC

Виконав: Єгоров Андрій Миколайович
_____
(прізвище, ім'я, по батькові) (підпис)

група ІПЗ-41

Керівник: к.т.н., доцент, доцент Остапчук О. П
_____
(науковий ступінь, вчене звання, посада, прізвище, ініціали) (підпис)

Рівне – 2024

ЗМІСТ

Національний університет водного господарства та природокористування
(повне найменування вищого навчального закладу)

Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії

Кафедра **комп'ютерних наук та прикладної математики**
Освітньо-кваліфікаційний рівень **бакалавр**
Галузь знань **12 Інформаційні технології**
(шифр і назва)
Спеціальність **121 Інженерія програмного забезпечення**
(шифр і назва)
Спеціалізація –

«ЗАТВЕРДЖУЮ»

Завідувач кафедри
д.т н., проф. Ю.В. Турбал

"1" жовтня 2023 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Єгорову Андрію Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *"Проектування та розробка веб-додатку для пошуку спільних поїздок на основі технології ASP.NET MVC"*

керівник роботи *Остапчук Оксана Петрівна, к.т.н., доцент, доцент кафедри комп'ютерних наук та прикладної математики.*

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада)

затверджені наказом по університету від *"22" квітня 2024 року С №-525.*

2. Термін подання роботи студентом *"1" червня 2024 року.*

3. Вихідні дані до роботи: *технології, які необхідні для розробки веб-додатку для пошуку спільних поїздок, список основних завдань і вимог щодо базових функціональних можливостей веб-додатку.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати предметну область, спроєктувати веб-додаток та описати технології розробки, розробити та протестувати веб-додаток.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) *Мультимедійна презентація*

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ 1	*доцент Остапчук О.П.*	*01.11.23*	*12.01.24*
Розділ 2	*доцент Остапчук О.П.*	*01.02.24*	*28.03.24*
Розділ 3	*доцент Остапчук О.П.*	*03.04.24*	*28.05.24*

7. Дата видачі завдання "*1" жовтня 2023 року.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	*Вивчення літератури за обраною тематикою*	*01.11.23 – 30.11.23*	*виконав*
2	*Проєктування веб-додатку*	*01.12.23 – 27.12.23*	*виконав*
3	*Розробка функціональних можливостей веб-додатку*	*01.02.24 – 29.02.24*	*виконав*
4	*Розробка дизайну веб-додатку*	*01.03.24 – 14.03.24*	*виконав*
5	*Проведення тестування веб-додатку*	*14.03.24 – 29.03.24*	*виконав*
6	*Проведення оптимізації веб-додатку*	*01.04.24 – 18.04.24*	*виконав*
8	*Загальні висновки до роботи*	*18.04.24 – 30.04.24*	*виконав*
9	*Підготовка звіту кваліфікаційної роботи*	*01.05.24 – 28.05.24*	*виконав*
10	*Підготовка мультимедійної презентації*	*01.06.24 – 05.06.24*	*виконав*
11	*Підготовка до виступу*	*05.06.24 – 10.06.24*	*виконав*

Студент _____ **Єгоров А.М.**
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Остапчук О.П.**
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Кваліфікаційна робота: 43 с., 33 рисунка, 6 таблиць, 12 джерел, 1 додаток.

Мета дослідження – проєктування та розробка веб-додатку, який допоможе користувачам знаходити пасажирів та водіїв для проведення спільних поїздок.

Об'єкт дослідження – розробка веб-додатку для пошуку спільних поїздок.

Предмет дослідження – процеси, технології та методи, що використовуються при розробці веб-додатків, зокрема зосереджені на ASP.NET MVC.

Методи вивчення – технологія ASP.NET MVC, платформа розробки .NET.

Розглянуто предметну область та описано поняття спільної поїздки. Проаналізовано подібні додатки, наведено їх позитивні та негативні сторони. Спроєктовано архітектуру веб-додатку та таблиці до його бази даних. Описано технолігії, що використовувались під час розробки. Розроблено веб-додаток для пошуку спільних поїздок на основі технології ASP.NET MVC.

Ключові слова: ВЕБ-ДОДАТОК, СПІЛЬНІ ПОЇЗДКИ, ASP.NET MVC, .NET, ФРЕЙМВОРК, БАЗА ДАНИХ, ПАТЕРН MVC.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних.

СКБД – система керування базою даних.

CLR – Common Language Runtime.

JIT – Just-In-Time compilation.

JS – JavaScript.

MVC – Model-View-Controller.

SQL – Structured Query Language.

AJAX – Asynchronous JavaScript and XML.

HTML – HyperText Markup Language.

CSS – Cascading Style Sheets.

EF Core – Entity Framework Core.

ВСТУП

Упродовж останніх кількох років спільні поїздки стали хорошою альтернативою маршрутним таксі. Для пасажира такі поїздки є більш гнучкими, мають вільний графік та маршрут, сам транспорт є комфортнішим, чим у громадських перевезеннях. Водії в свою чергу можуть повернути частину коштів або взагалі покрити витрати на пальне.

За дослідженням проведеним blablacar в квітні 2017 року, на питання про те, чи впливає присутність попутників на організацію спільної поїздки, більшість респондентів відповіли ствердно. Близько 72% відзначили, що з попутниками їхня поїздка стає цікавішою. Для 75% водіїв присутність попутників впливає на організованість поїздки, а саме – на правильну підготовку автомобіля до тривалої поїздки, виїзд і приїзд за графіком та інші важливі чинники. Також спільна подорож стимулює водіїв концентруватися на дорозі та дотримуватися правил дорожнього руху, відмовлятися від шкідливих звичок [1]. Тому пошук спільних поїздок стає все актуальнішим завданням. У цьому контексті, веб-додаток, має великий потенціал у сприянні зручності організації таких поїздок, потрібно лиш відкрити необхідну сторінку в браузері, щоб всі можливі варіанти подорожей представились перед користувачем.

Такі сервіси вже існують, але в них є недоліки: високі сервісні збори, велика кількість шахраїв, проблеми із поверненням грошей в разі відміни поїздки. Тому хорошою ідеєю є створення додатку, який буде позбавлений цих недоліків.

Мета кваліфікаційної роботи полягала у проєктуванні та розробці такого веб-додатку, який допоможе користувачам знаходити пасарижів та водіїв для проведення спільних поїздок, забезпечуючи зручний і надійний інструмент для планування подорожей.

Для досягнення мети були визначені наступні завдання:

1. Визначити вимоги до веб-додатку на основі потреб користувачів та аналізу аналогів на ринку.

2. Спроєктувати систему, вибрати технології та інструменти для реалізації поставлених вимог, розробити архітектуру веб-додатку.

3. Реалізувати функціональні можливості веб-додатку на основі розробленої архітектури та провести тестування для забезпечення надійності та ефективності.

Для створення веб-додатку було використано передову технологію ASP.NET з архітектурним патерном MVC (Model-View-Controller). Цей патерн дозволяє розділити логіку додатку на три взаємодіючі компоненти, що робить розробку більш гнучкою та масштабованою. Мовою програмування обрано C#. Для налаштування дизайну сторінок вибрано CSS фреймворк Bootstrap [3]. Для роботи з базою даних було використано Entity Framework, а в якості системи керування базою даних – MySQL.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Опис предметної області

Спільні поїздки – це спільне використання приватного автомобіля декількома людьми, зазвичай для економії коштів. Це дозволяє використовувати вільні місця в автомобілі для перевезення попутників, що знижує витрати на проїзд для пасажирів та витрати на пальне для водіїв. Також це сприяє меншій кількості автомобілів на дорогах, оскільки не кожному потрібно мати авто, щоб дістатися до місця. В свою чергу, це дозволяє зменшити затори та забруднення повітря. Дане явище набуло популярності завдяки сучасним технологіям, що дозволяють легко знаходити попутників через спеціальні мобільні та веб-додатки.

Дані сервіси включають в себе наступні можливості: знайти поїздки за заданим маршрутом, зареєструватися на поїздку, створювати свої поїздки за наявності власного автомобіля та водійського посвідчення.

1.2. Аналіз існуючих рішень для пошуку спільних поїздок

Одним із найпопулярніших сервісів є BlaBlaCar (рис. 1.1). BlaBlaCar – це платформа для спільних поїздок, яка з'єднує мільйони водіїв і пасажирів, що подорожують в одному напрямку. BlaBlaCar був заснований у 2006 році у Франції Фредеріком Маззеллою. Спочатку сервіс називався Covoiturage.fr, але у 2013 році отримав назву BlaBlaCar, що відображає рівень балакучості пасажирів (Bla, BlaBla, BlaBlaBla).

Можливості BlaBlaCar:

1. Реєстрація та створення профілю. Користувачі можуть створювати профілі, де вказують особисту інформацію, фотографії, рівень балакучості та інші деталі. Профілі також включають рейтинги та відгуки від інших користувачів, що підвищує рівень довіри.

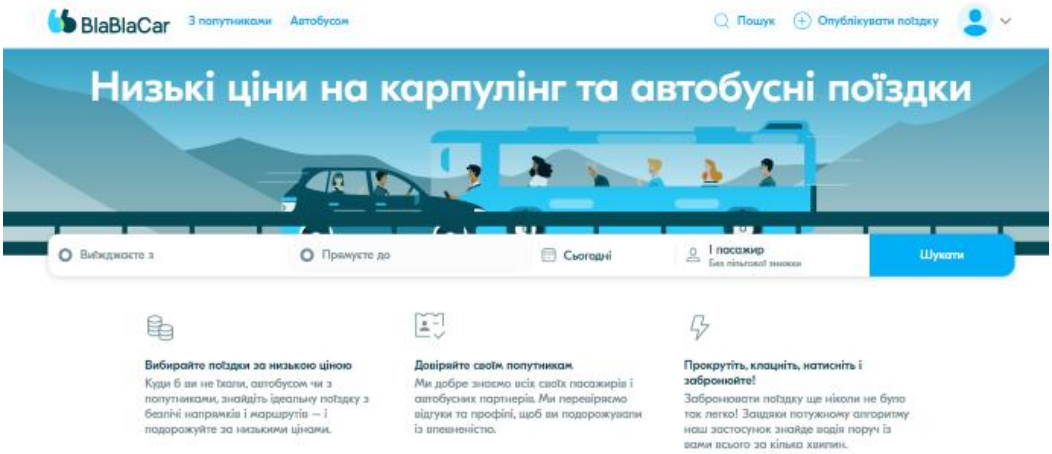

Рис. 1.1. Вигляд головної сторінки BlaBlaCar

2. Пошук та бронювання поїздок. Користувачі можуть шукати поїздки за різними параметрами: маршрут, дата і час відправлення, вартість, кількість вільних місць. Після знаходження відповідної поїздки пасажир може забронювати місце і зв'язатися з водієм.

3. Оплата. BlaBlaCar пропонує різні методи оплати, включаючи онлайн-платежі через платформу. Це дозволяє знизити ризики, пов'язані з обміном готівки під час поїздки.

4. Система рейтингів та відгуків. Користувачі можуть залишати відгуки та оцінки після кожної поїздки, що допомагає іншим користувачам обрати кращого водія або пасажира.

Переваги BlaBlaCar:

– велика база користувачів;

– перевірений часом;

– система відгуків;

– зручний у використанні;

– має додаткові послуги (автобусні сполучення).

Недоліки BlaBlaCar:

– високі сервісні збори;

– проблеми з поверненням коштів у разі відміни поїздки.

Ще одним сервісом для пошуку спільних поїздок є Monocar (рис. 1.2). Це відносно новий мобільний додаток, випущений в 2021 році, який ще знаходиться на стадії бета тестування.

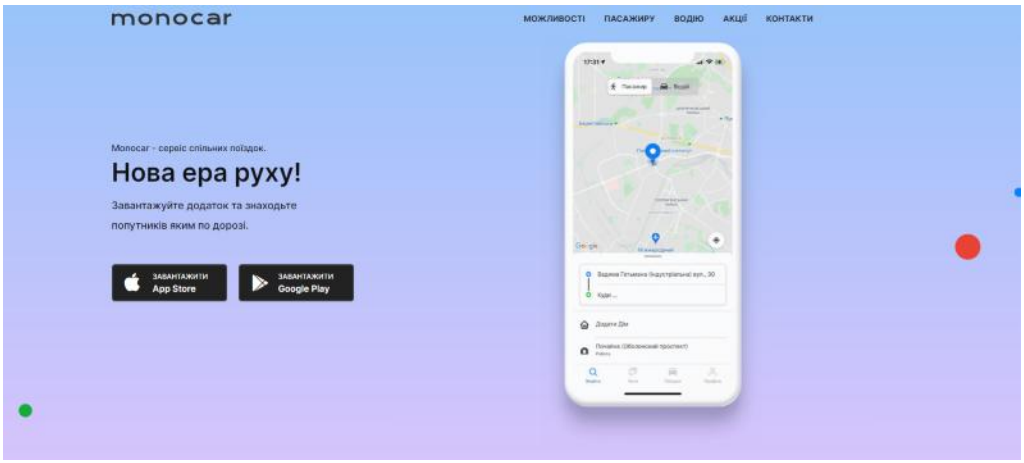

Рис. 1.2. Вигляд головної сторінки Monocar

Переваги Monocar:

– орієнтування на український ринок;

– відсутність сервісних зборів.

Недоліки Monocar:

– мала кількість користувачів;

– доступ лише через мобільний додаток;

– додаток все ще в розробці.

1.3. Постановка вимог

Після детального аналізу предметної області та подібних додатків можна визначити основну та додаткові вимоги до проєкту.

Основною вимогою до проєкту є створення повноцінного веб-додатку з простим, сучасним та зрозумілим інтерфейсом для пошуку спільних поїздок. Якщо пошук здійснює звичайний користувач, то йому необхідно лише ввести необхідні дані для пошуку та вибрати найкращий варіант. Якщо користувач хоче знайти собі попутників, то йому потрібно пройти так звану верифікацію на роль водія і після цього на спеціальній сторінці створити та оформити свою поїдку.

Також веб-додаток повинен надавати наступні можливості:

1. Створювати новий та входити у вже створений акаунт.

2. Надавати дозвіл ставати водієм після заповнення деяких додаткових персональних даних.

3. Залишати відгуки та скарги.

4. Переглядати минулі та заплановані поїздки.

5. Додавати нові міста у базу, переглядати заявки водіїв та скарги користувачів. (для адміністраторів)

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА ОПИС ЗАСОБІВ РЕАЛІЗАЦІЇ ВЕБ-ДОДАТКУ

2.1. Мова програмування C#

В цей час мова програмування C# одна з найпотужніших і найзатребуваніших мов в IT-галузі. На даний момент за допомогою цієї мови розробляються різні програми: від невеликих десктопних програм до великих веб-сервісів, які щодня обслуговують мільйони користувачів.

На даний момент є найпопулярнішою мовою для платформи .NET. C# – це строго типізована об'єктно-орієнтована мова програмування, такий підхід дозволяє вирішити завдання з побудови великих, але в той же час гнучких, масштабованих додатків, що розширюються. Мова C#, так як і вся платформа .NET, вже пройшла великий шлях. Перша версія мови вийшла разом із релізом Microsoft Visual Studio .NET у лютому 2002 року. Поточною версією мови є версія 12, яка вийшла 14 листопада 2023 разом із релізом .NET 8. C# є мовою із C-подібним синтаксисом і близька у цьому відношенні до C++ та Java. Тому якщо ви знайомі з однією з цих мов, то оволодіти C# буде легше. C# продовжує активно розвиватися і з кожною новою версією з'являється все більше цікавих функціональностей [2-4].

2.2. Платформа .NET

Часто коли говорять C#, мають на увазі технології платформи .NET. І навпаки, коли говорять .NET, нерідко мають на увазі C#. Однак, хоча ці поняття і пов'язані, ототожнювати їх не потрібно. Мова C# була створена спеціально для роботи з .NET, проте саме поняття .NET дещо ширше. .NET – це безкоштовна кросплатформна платформа розробника з відкритим вихідним кодом для створення різноманітних програм.

Можна виділити такі основні переваги платформи .NET:

1. Підтримка кількох мов. Основою платформи є загальномовне середовище виконання Common Language Runtime (CLR). Завдяки цьому, при

компіляції код будь-якою з мов .NET компілюється у проміжну мову CIL (Common Intermediate Language) JIT (Just-in-Time) компілятором. Тому за певних умов ми можемо зробити окремі модулі однієї програми окремими мовами.

2. Кросплатформеність. .NET є платформою, що підтримується на більшості сучасних ОС Windows, MacOS, Linux, Andoid, iOS.

3. Велика бібліотека класів. .NET представляє широкий спектр бібліотек класів. При розробці будь-якого продукту на C# – текстового редактору, чату або складного веб-сайту, так чи інакше буде використана бібліотека класів .NET.

4. Різноманітність технологій. Загальномовне середовище виконання CLR та базова бібліотека класів є основою цілого стеку технологій, які розробники можуть задіяти при побудові тих чи інших додатків. Наприклад, для роботи з базами даних призначено технологію ADO.NET та Entity Framework Core. Для побудови графічних додатків з багатим насиченим інтерфейсом – технологія WP, для створення більш простих графічних додатків – Windows Forms. Для розробки кросплатформених мобільних та десктопних програм – Xamarin/MAUI. Для створення веб-сайтів та веб-додатків – ASP.NET і т.д.

5. Продуктивність. Програми на .NET відрізняються високою продуктивністю, згідно з низкою тестів веб-програми на .NET у ряді категорій сильно випереджають програми, побудовані за допомогою інших технологій (рис. 2.1).

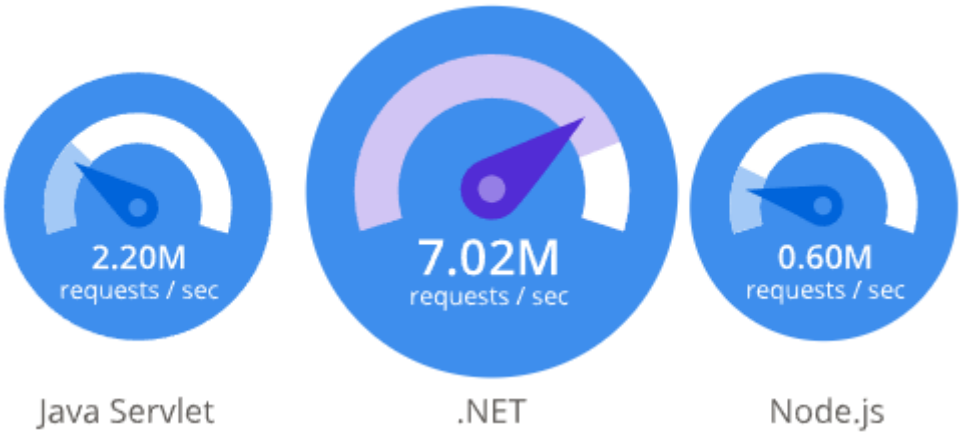

Рис. 2.1. Продуктивність .NET у порівнянні з іншими техногологіями

6. Автоматично керує пам’яттю. Загальномовне середовище виконання саме викликає збирач сміття коли необхідно та очищує пам'ять. А це означає, що розробнику не доведеться, на відміну від мови C++, дбати про звільнення пам'яті [5].

2.3. Технологія ASP.NET Core MVC

ASP.NET Core (Active Server Pages .NET) – це веб-фреймворк, розроблений компанією Microsoft, який дозволяє створювати динамічні веб-сайти, веб-додатки та веб-сервіси. Він являє собою частину платформи .NET і надає потужний набір інструментів для розробки серверної частини веб-додатків.

ASP.NET Core MVC – це частина платформи ASP.NET Core, особливістю якої є застосування патерну MVC (Model-View-Controller). Перевагою використання фрейморка ASP.NET Core MVC у порівнянні з простим ASP.NET Core є те, що він спрощує розробку додатків, особливо коли розроблюється великий додаток.

Компоненти патерну MVC:

– Model. Модель відповідає за дані додатка. Вона містить правила, які керують поведінкою додатку, включаючи валідацію даних, взаємодію з базою даних, обробку даних та їх маніпуляцію. Модель не знає нічого про відображення або контролер. Вона просто надає дані, які контролер може отримати та змінювати.

– View. Відображення відповідає за вивід даних користувачеві. Воно отримує дані від моделі через контролер і форматує їх для виводу на веб-сторінку. Відображення не містить бізнес-логіки, а лише логіку відображення. Воно не змінює дані, а лише відображає їх.

– Controller. Контролер відповідає за обробку запитів від користувача. Він отримує запити, викликає відповідні методи моделі для обробки даних, і потім передає дані для відображення. Контролер виступає посередником між моделлю та відображенням, координуючи їх взаємодію [6].

Приклад роботи такої архітектури відображено на рис. 2.2.

1. Користувач взаємодіє з View, яке в свою чергу відправляє запит до Controller.

2. Controller оброблює запит та направляє його у необхідний метод. Якщо в результаті роботи метода проводиться обробка даних, то здійснюється запит до Model, яка дістає їх з бази даних.

3. Дані повертаються з бази даних, оброблюються і відображаються користувачу через View.

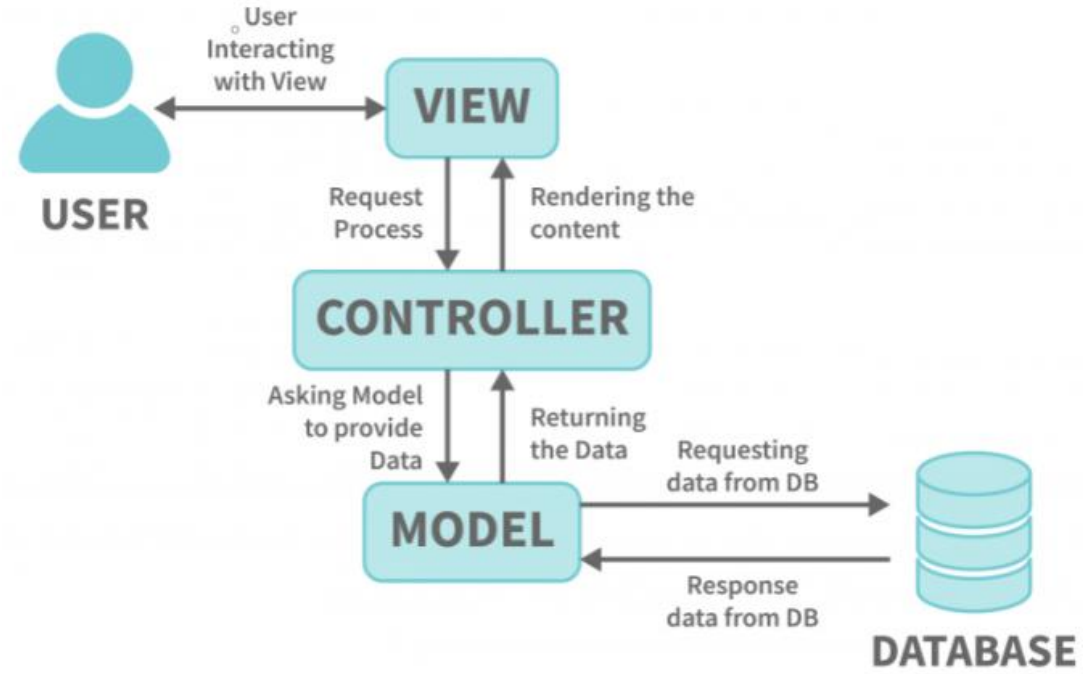

Рис. 2.2. Структура моделі MVC

2.4. Bootstrap

Bootstrap – це популярний фронтенд-фреймворк для створення адаптивних веб-сайтів та веб-додатків (рис. 2.3). Він містить набір інструментів, що включає шаблони дизайну на основі HTML, CSS та JavaScript. Bootstrap розроблений компанією Twitter і випущений у відкритий доступ. Bootstrap значно спрощує процес розробки адаптивного та сучасного інтерфейсу користувача [7].

Рис. 2.3. Логотип Bootstrap

2.5. Entity Framework Core

Entity Framework Core (EF Core) – це кросплатформений інструмент об'єктно-реляційного зіставлення (ORM), за домогою якого можна працювати з базою даних, використовуючи об'єкти мови C#. Це зменшує кількість написаного SQL-коду та знижує ймовірність помилок.

У EF Core доступ до даних здійснюється за допомогою моделі. Модель складається з класів сутностей і об’єкта контексту, який представляє сеанс із базою даних. Об’єкт контексту дозволяє запитувати та зберігати дані. При цьому для запитів до даних використовується спеціальна мова запитів LINQ (Language-Integrated Query). EF Core підтримує різні підходи до розробки взаємодії додатку з БД, як створення моделі на основі існуючої БД, так і створення БД після створення моделі у додатку [8].

2.6. MySQL

MySQL – це система керування базами даних яка використовується для зберігання, обробки та управління даними (рис. 2.4). Вона забезпечує дуже швидкий, багатопоточний, багатокористувацький і надійний сервер бази даних SQL. MySQL є відкритим програмним забезпеченням, але також доступна комерційна ліцензія для тих, хто хоче використовувати її у власних комерційних продуктах [9].

Рис. 2.4. Логотип MySQL

2.7. Проєктування БД

База даних – це організована колекція структурованої інформації, або даних, які зазвичай зберігаються в електронному форматі в комп'ютерній системі. Зазвичай, базою даних керують за допомогою системи керування базами даних (СКБД). Дані, СКБД і програми, пов'язані з ними, разом називають системою баз даних, скорочено просто базою даних (БД).

У найпоширеніших сучасних типах БД дані, зазвичай, представлено як рядки й стовпці в низці таблиць, щоб підвищити ефективність обробки та запитів щодо даних. Після цього до даних можна легко отримати доступ, керувати ними, а також змінювати, оновлювати, контролювати та впорядковувати їх. Більшість баз даних використовують мову структурованих запитів (SQL) для запису й запитів щодо даних [10-12].

Як було згадано вище, в якості СКБД даного проєкту було використано MySQL. Всього для зберігання в БД виділено 6 сутностей, які описані в наступних таблицях (табл. 2.1-2.6).

Таблиця 2.1

Сутність користувача (user)

Назва поля	Тип даних	Короткий опис
Id	int	Унікальний ідентифікатор
Email	varchar	Електронна пошта
Login	varchar	Логін

Продовження таблиці 2.1

Password	varchar	Пароль
Name	varchar	Ім’я
Surname	varchar	Прізвище
PhoneNumber	varchar	Номер телефону
Role	varchar	Роль
DriverLicense	longblob	Зображення водійського посвідчення
AmountOfBadReports	int	Кількість скарг
AmountOfReports	int	Кількість відгуків
Rating	float	Рейтинг
isBanned	tinyint	Чи є акаунт заблокованим

Таблиця 2.2

Сутність міста (city)

Назва поля	Тип даних	Короткий опис
Name	varchar	Назва міста

Таблиця 2.3

Сутність поїздки (trip)

Назва поля	Тип даних	Короткий опис
Id	int	Унікальний ідентифікатор
DepartureTime	datetime	Час відправлення
DeparturePoint	varchar	Місце відправлення (Зовнішній ключ до поля Name сутності міста)

Продовження таблиці 2.3

ArrivalPoint	varchar	Місце прибуття (Зовнішній ключ до поля Name сутності міста)
AmountofFreeSeats	int	Кількість вільних місць
AmountOfSeats	int	Загальна кількість місць
Price	int	Ціна
UserId	int	Ідентифікатор створившого (Зовнішній ключ до поля Id сутності користувача)

Таблиця 2.4

Сутність реєстрації на поїздку (trip registration)

Назва поля	Тип даних	Короткий опис
Id	int	Унікальний ідентифікатор
UserId	int	Ідентифікатор користувача, що зареєструвався (Зовнішній ключ до поля Id сутності користувача)
TripId	int	Ідентифікатор поїздки(Зовнішній ключ до поля Id сутності поїздки)

Таблиця 2.5

Сутність заявки на роль водія (request)

Назва поля	Тип даних	Короткий опис
Id	int	Унікальний ідентифікатор
Name	varchar	Імя користувача
Surname	varchar	Прізвище користувача
DriverLicense	longblob	Зображення водійського посвідчення
PhoneNumber	varchar	Номер телефону
Status	varchar	Статус заявки (прийнята, відхилена, розглядається)
UserId	int	Ідентифікатор користувача (Зовнішній ключ до поля Id сутності користувача)

Таблиця 2.6

Сутність відгуку (скарги) на водія (report)

Назва поля	Тип даних	Короткий опис
Id	int	Унікальний ідентифікатор
Text	longtext	Текст відгуку(скарги)
Rating	int	Оцінка водія
CreatorId	int	Ідентифікатор створившого (Зовнішній ключ до поля Id сутності користувача)

Продовження таблиці 2.6

ReportedId	int	Ідентифікатор водія на якого пишеться відгук(скарга) (Зовнішній ключ до поля Id сутності користувача)
TripId	int	Ідентифікатор поїздки(Зовнішній ключ до поля Id сутності поїздки)

На рис. 2.5 зображено діаграму зв'язку між сутностями (ER Diagram).

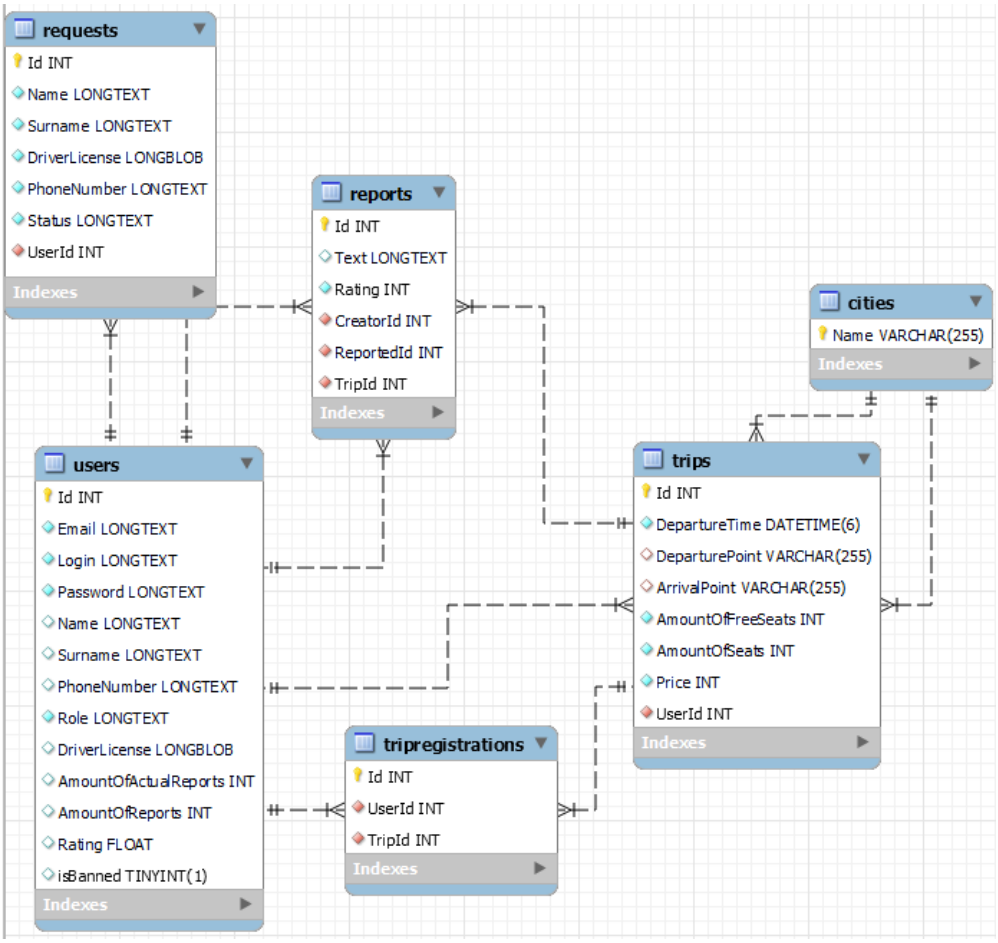

Рис. 2.5. ER Diagram

2.8. Проєктування архітектури веб-додатку

Оскільки веб-додаток проєктується з використанням патерну MVC, він був розділений на три частини: моделі, відображення та контролери.

Реалізовано наступні моделі:

– City;

– DriverRequest;

– Report;

– Trip;

– TripRegistration;

– User.

Для посередництва між даними та відображеннями створені такі контролери:

– UserController – для обробки запитів, що пов'язані з користувачем, його профілем, аутентифікацією;

– TripController – для запитів безпосередньо по'вязаних із поїдками.

Для висвітлення даних використано ряд відображень.

Відображення TripController:

– Create – для створення поїздок;

– Index – головна сторінка, сторінка пошуку;

– MyTrips – сторінка поїздок конкретного користувача;

– Report – сторінка для залишення відгуків або скарг.

Відображення UserController:

– Index – сторінка входу в акаунт та реєстрації нового;

– AdminPanel – панель адміністратора;

– Profile – профіль користувача.

РОЗДІЛ 3
РОЗРОБКА ТА ОПИС РОБОТИ ВЕБ-ДОДАТКУ

3.1. Розробка аутентифікації

Для реалізації аутентифікації було створено: модель користувача, модель відображення користувача, відображення логіну та реєстрації. Також додано чотири методи до контролеру UserController.

Модель користувача містить поля, що описані у розділі проєктування.

Модель відображення – це особлива модель, що використовується для відокремлення моделі від відображення та для передачі даних у відображення. Модель відображення користувача містить тільки ті поля, що безпосередньо використовуються у процесі входу та реєстрації акаунта. Також, вона використовується для валідації введених даних.

Сторінки входу та реєстрації зображені на рис. 3.1 та рис. 3.2 відповідно. На них розміщенні поля для введеня даних та кнопки для підтвердження дії.

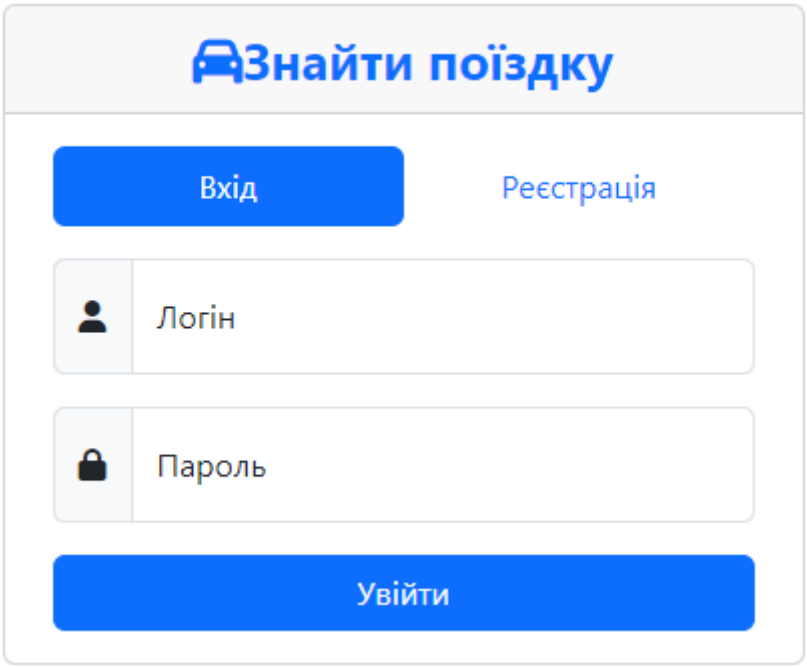

Рис. 3.1. Вигляд сторінки входу

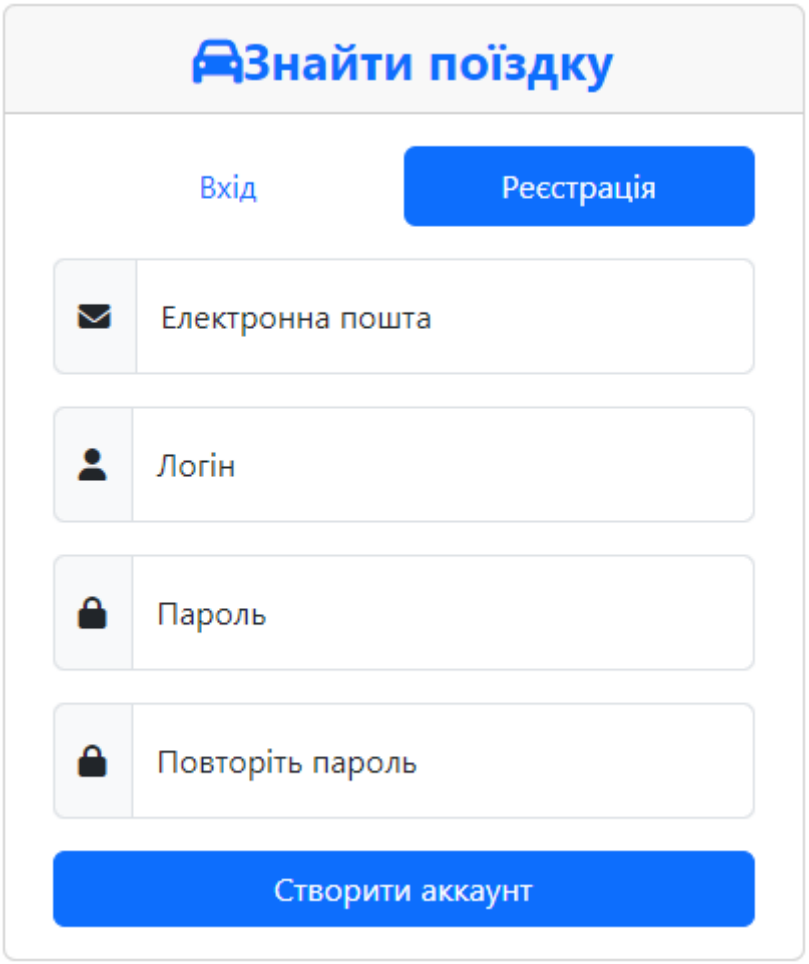

Рис. 3.2. Вигляд сторінки реєстрації

Введені дані після натиснення кнопки передаються у контролер, де вони валідуються. Після цього, для входу перевіряються наступні умови: чи існує такий акаунт, чи введений пароль підходить. Для реєстрації перевіряється чи є логін та електронна пошта вільними. Якщо умови виконуються, то користувача аутентифікує, визначається його роль для доступу до інших функцій. Далі кнопка входу замінюється меню користувача.

3.2. Розробка профілю користувача

У кожного аутентифікованого користувача є профіль, де користувач може переглянути свої персональні дані, змінити пароль, подати заявку на роль водія або переглянути її статус (рис. 3.3).

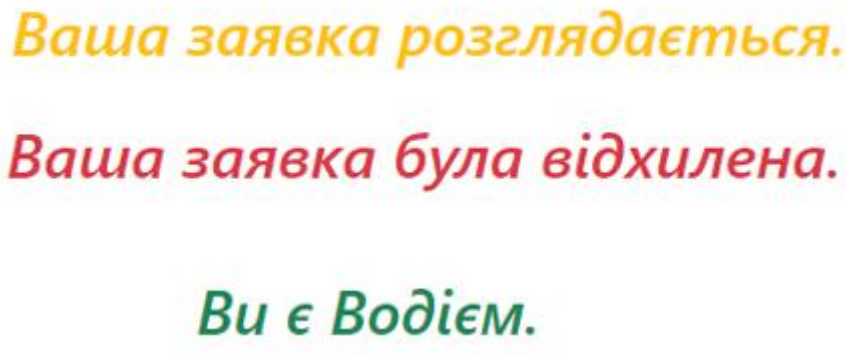

Рис. 3.3. Статуси заявки користувача

На рис. 3.4 можна побачити, що для відправлення заявки на водія необхідно прикріпити фото водійського посвідчення, ввести ім’я та прізвище, номер телефону. Для передачі цих даних у контролер створено модель відображення водія. Для роботи функціональності профілю було створено методи для зміни паролю, створення заявки. Також для виводу даних використовується модель відображення користувача.

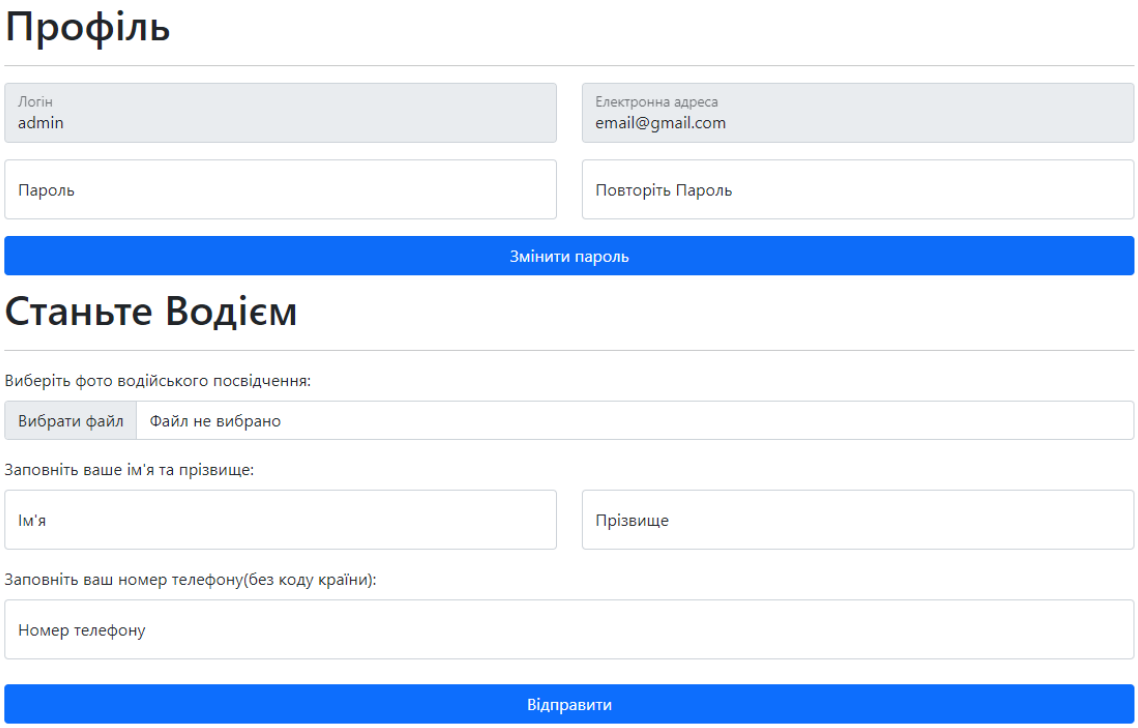

Рис. 3.4. Вигляд сторінки профілю

3.3. Розробка створення поїздок

Якщо користувач пройшов верифікацію, він отримує доступ до сторінки створення (рис. 3.5). На цій сторінці розміщено форму, яку потрібно заповнити для створення поїздки.

Рис. 3.5. Вигляд сторінки створення поїздки

Для полегшення процесу заповнення форми використовується “автозаповнення” (рис. 3.6). Воно реалізоване через JS та використовує AJAX-запити.

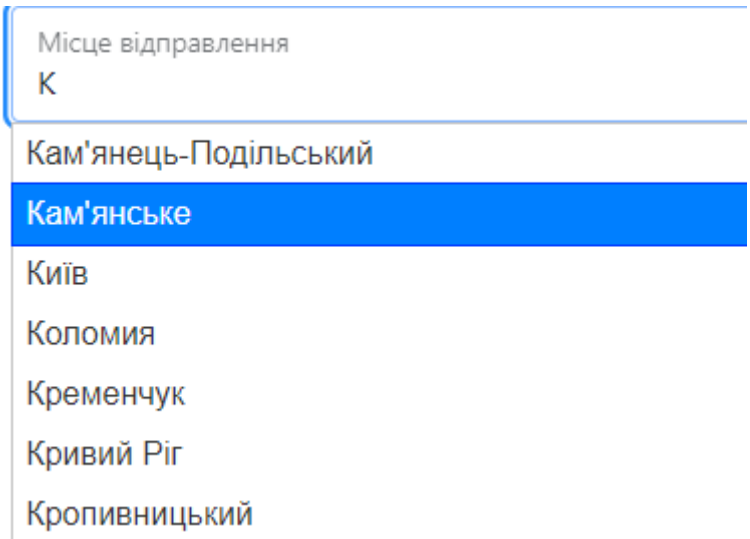

Рис. 3.6. Приклад автозаповнення

Також був створений метод для створення поїдки, який її валідує та додає у базу даних.

3.4. Розрозробка пошуку поїздок

Сторінка пошуку поїздок є головною, нею може користуватися навіть не аутентифікований користувач (рис. 3.7). У верхній частині сторінки знаходиться форма пошуку, після здійснення якого, результати будуть зображені в нижній частині сторінки у вигляді таблиці. При натисненні кнопки дані форми передаються у контролер поїздок (TripController). Він передає їх у метод, що звертається до бази даних та повертає назад поїздки, які відповідають заданим критеріям (рис. 3.8).

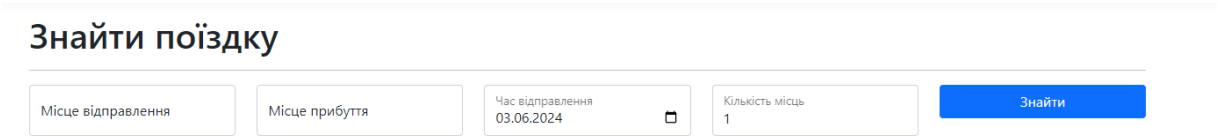

Рис. 3.7. Сторінка пошуку поїздок

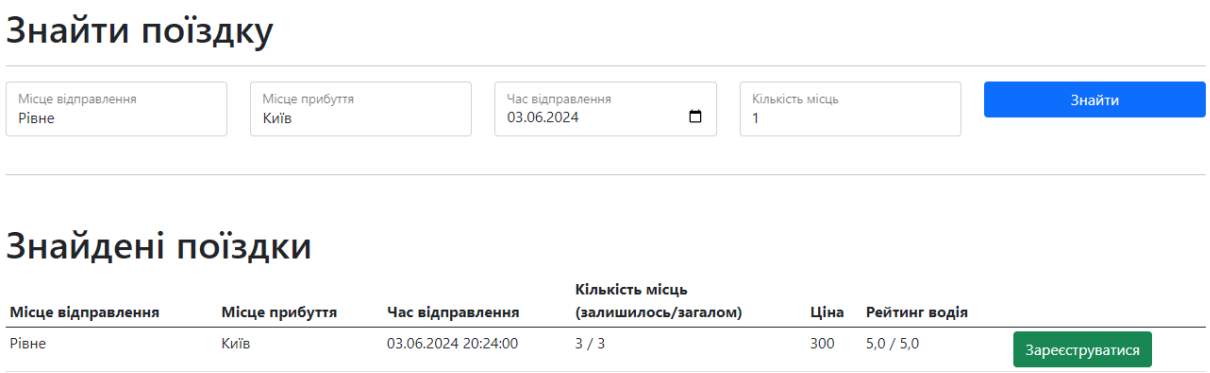

Рис. 3.8. Результат пошуку поїздки

3.5. Розробка перегляду поїздок користувача

Сторінка перегляду поїздок служить місцем, де користувач може переглянути усі свої завершені та заплановані поїздки, залишити відгук, відмінити реєстрацію на поїздку. Сторінка складається з двох частих: заплановані (рис. 3.9) та завершені поїздки (рис. 3.10). Кожна з цих частин містить таблицю з даними поїздок користувача.

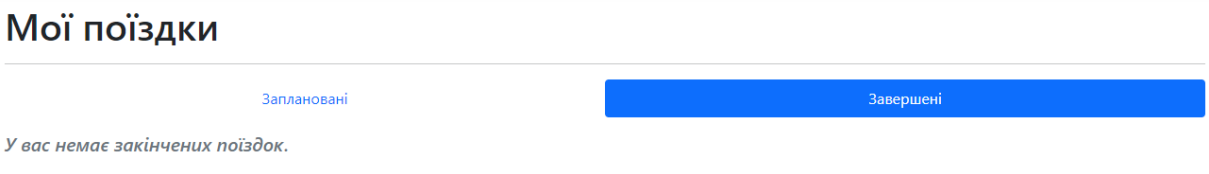

Рис. 3.9. Сторінка перегляду завершених поїздок

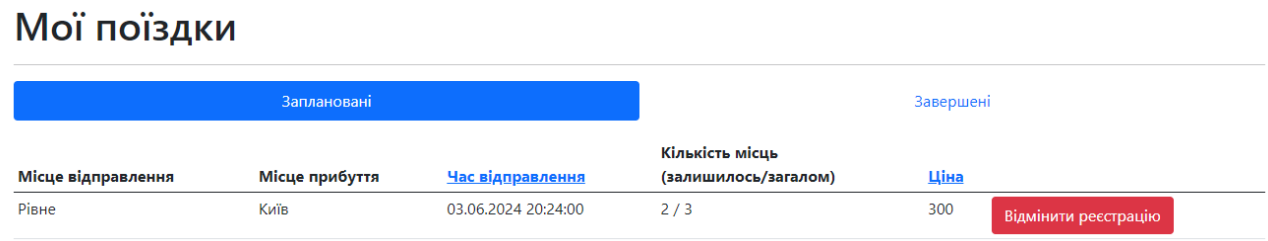

Рис. 3.10. Сторінка перегляду запланованих поїздок

3.6. Розробка відгуків (скарг)

Для залишення відгуку або скарги користувач повинен зайти на сторінку своїх поїздок та написнути кнопку “Залишити відгук” (рис. 3.11). Користувач має право залишити відгук лише протягом 24 годин від початку поїздки.

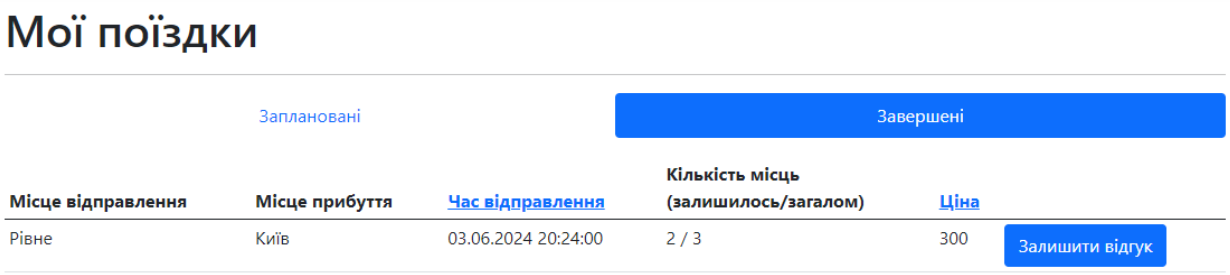

Рис. 3.11. Кнопка для залишення відгуку

На сторінці відгуку користувачу пропонується написати короткий коментар, та поставити оцінку від 1 до 5 зірок, при цьому відгук з 1 зіркою буде вважатися скаргою (рис. 3.12).

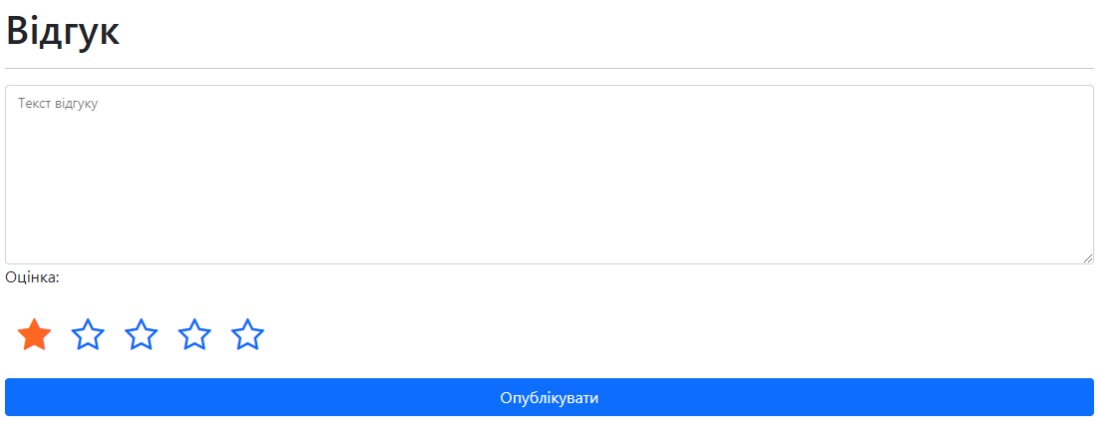

Рис. 3.12. Сторінка залишення відгуку

3.7. Розробка панелі адміністратора

Панель адміністратора – це сторінка, до якої має доступ лише адміністратор. На ній можна: додавати міста у БД, переглядати заявки на роль водія, розблоковувати водіїв та переглядати скарги на них (рис. 3.13-рис. 3.15).

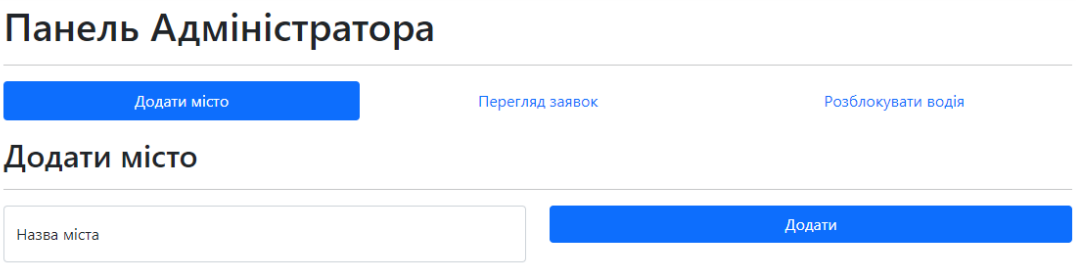

Рис. 3.13. Панель адміністратора (додавання міста)

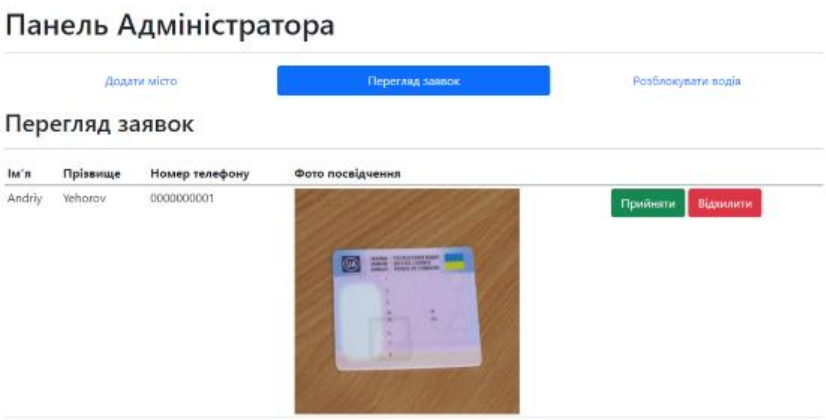

Рис. 3.14. Панель адміністратора (перегляд заявок)

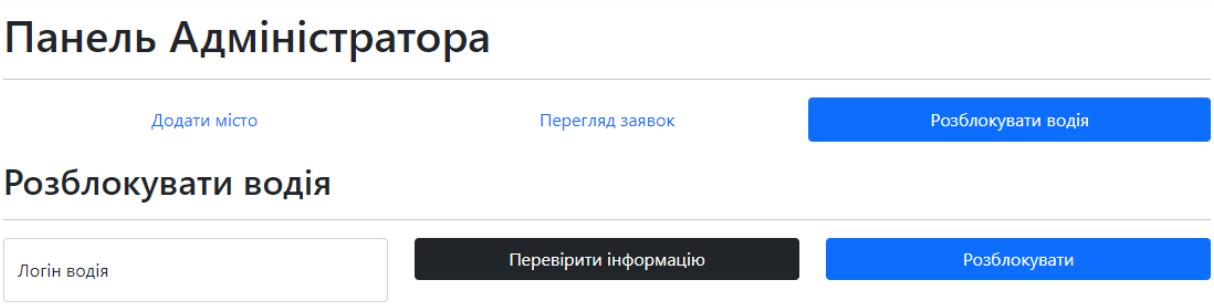

Рис. 3.15. Панель адміністратора (розблокування водія)

3.8. Опис роботи веб-додатку

Для початку роботи потрібно перейти за посиланням, при цьому з’явиться головна сторінка. На ній можна знайти поїздку, або увійти в акаунт (рис. 3.16).

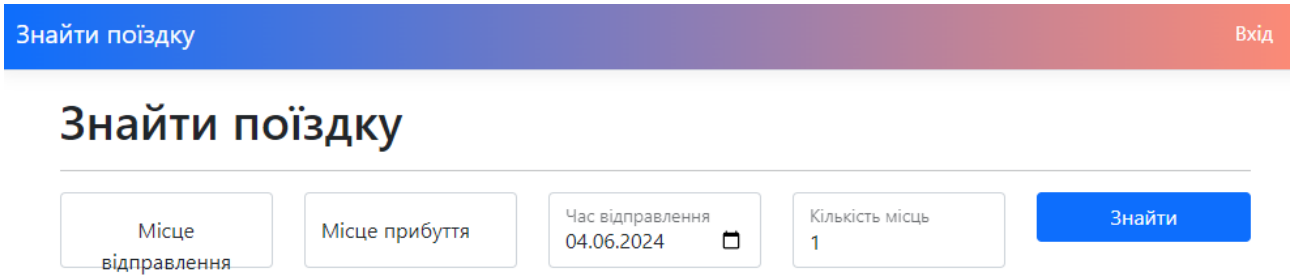

Рис. 3.16. Головна сторінка

Робота валідації при реєстрації зображена на рис. 3.17.

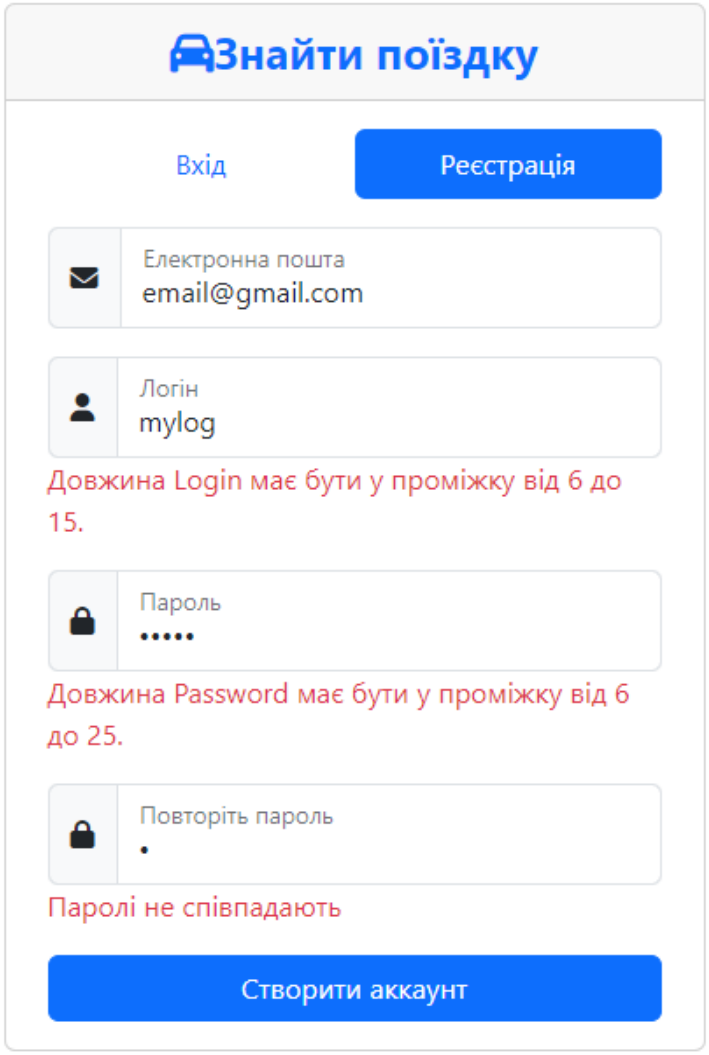

Рис. 3.17. Помилки валідації

Після успішної реєстрації кнопка входу замінюється логіном користувача, також з'являється кнопка "Мої поїздки" (рис. 3.18). При натисненні на логін відкривається меню.

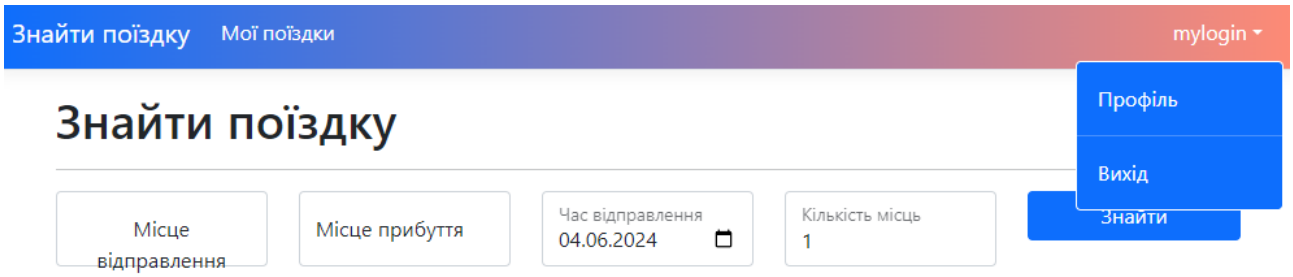

Рис. 3.18. Головна сторінка після входу

Створений запис відповідного користувача у базі даних зображено на рис. 3.19.

Id	Email	Login	Password	Name	Surname	PhoneNumber	Role
4	myemail@gmail.com	mylogin	mypassword	NULL	NULL	NULL	user

Рис. 3.19. Запис користувача у базі даних

Процес зміни паролю та подання заявки на роль водія зображено на рис. 3.20.

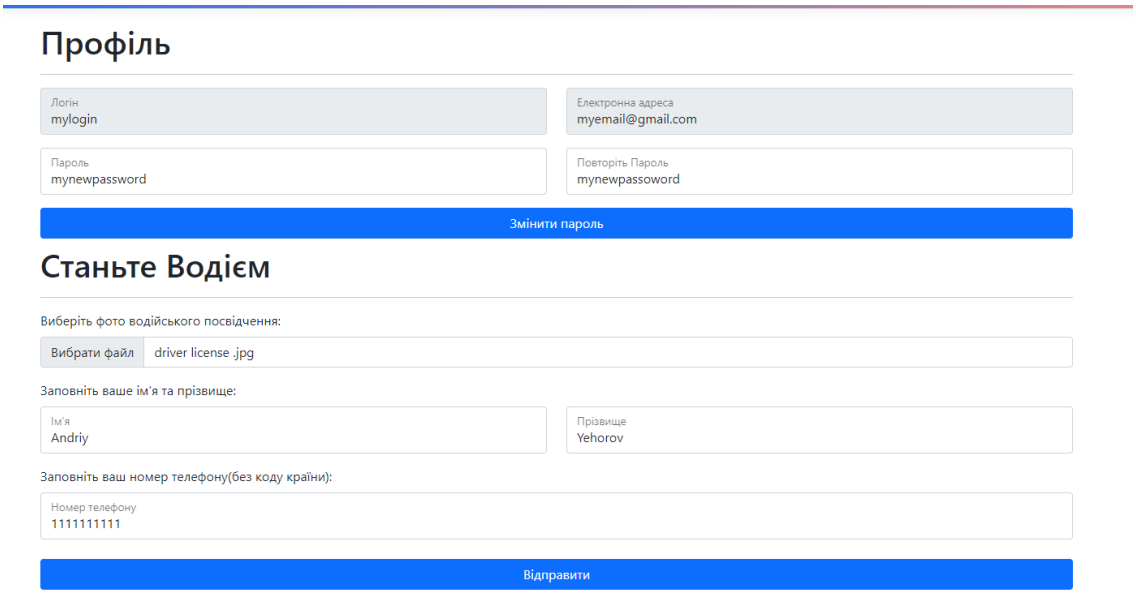

Рис. 3.20. Профіль нового користувача

Після натискання кнопки “Змінити пароль” здійснюється перевірка на співпадіння паролів та проводиться зміна паролю у базі даних (рис. 3.21).

Id	Email	Login	Password
4	myemail@gmail.com	mylogin	mynewpassword

Рис. 3.21. Зміна паролю

Після натискання кнопки “Відправити” створюється нова заявка у базі даних, яку повинен розглянути адміністратор (рис. 3.22).

Id	Name	Surname	DriverLicense	PhoneNumber	Status	UserId
21	Andriy	Yehorov	BLOB	1111111111	on review	4

Рис. 3.22. Запис заявки у БД

Далі адміністратор розглядає заявку та прийняє чи відхиляє її. Якщо заявка прийнята, у користувача змінюється роль з “user” на “driver” (рис. 3.23).

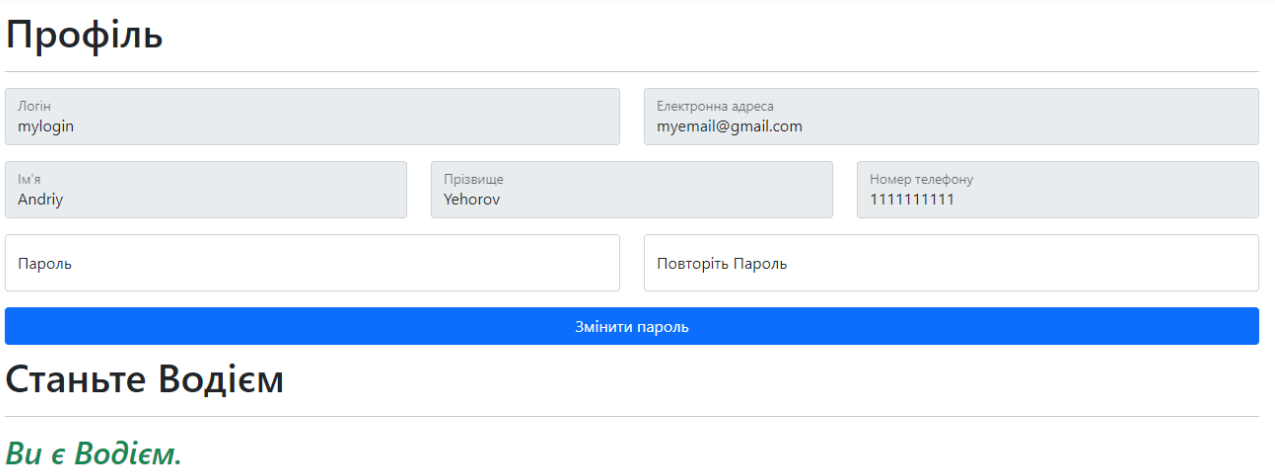

Рис. 3.23. Вигляд профілю після прийняття заявки

Тепер користувач має можливість створювати поїздки (рис. 3.24). Після натиснення кнопки “Опублікувати” запис про дану поїздку з’явиться у БД та буде доступний для пошуку (рис. 3.25, рис. 3.26).

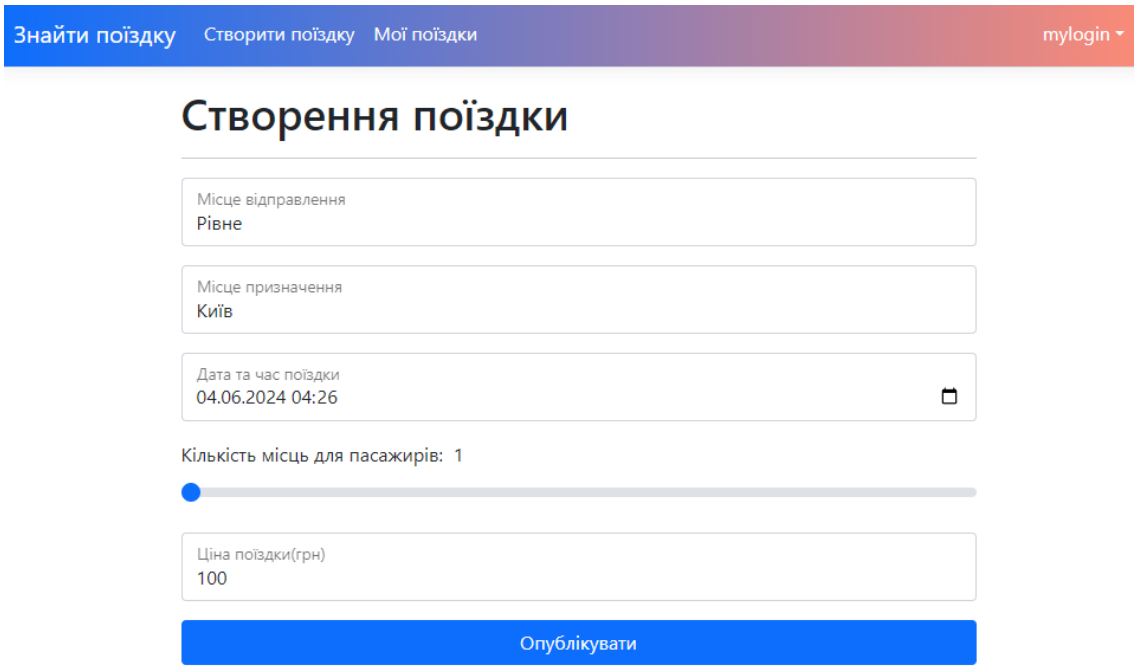

Рис. 3.24. Створення поїздки

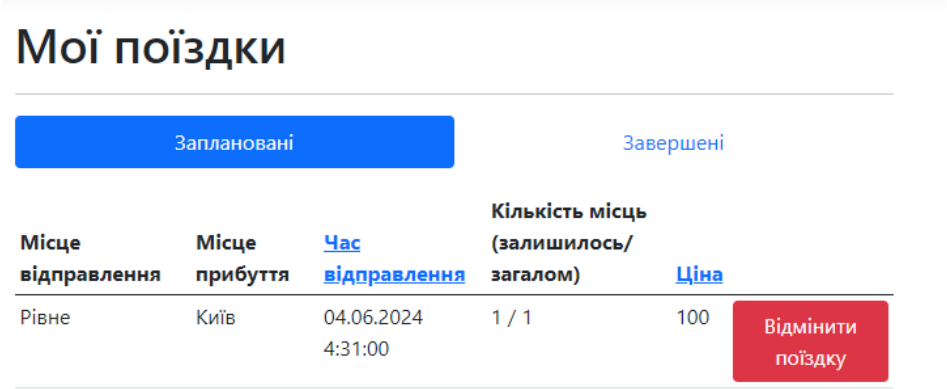

Рис. 3.25. Створена поїздка у розділі “Мої поїздки”

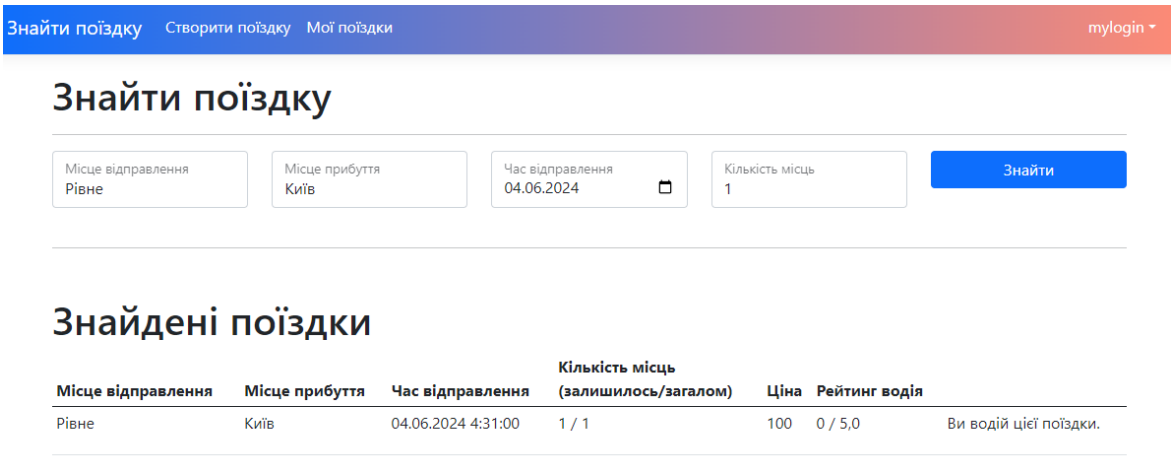

Рис. 3.26. Створена поїздка у пошуку

ВИСНОВКИ

Під час виконання кваліфікаційної роботи, мету, що полягала у проєктуванні та розробці веб-додатку, який допоможе користувачам знаходити пасарижів та водіїв для проведення спільних поїздок було досягнуто. Також були виконані всі поставлені завдання.

В першу чергу розглянуто предметну область та описано поняття спільної поїздки. Також, проаналізовано подібні додатки, наведено їх позитивні та негативні сторони. Поставлено основні та додаткові вимоги до проєкту.

В процесі роботи оглянуто технології, які були використані під час розробки веб-додатку, а саме:

– Мова програмування C#;

– Платформа .NET;

– ASP.NET Core MVC;

– Bootstrap;

– Entity Framework Core;

– MySQL.

Для кожної технології наведено короткий опис та переваги використання. Відповідно до вимог, спроєктовано архітектуру веб-додатку та таблиці для його бази даних.

В останню чергу було описано розробку компонент додатку таких, як: аутентифікація, профіль користувача, створення поїздок, пошук поїздок, перегляд поїздок користувача, залишення відгуків (скарг), панель адміністратора. Також описано роботу додатку на реальному прикладі.

Після аналізу схожих додатків, а саме: blablacar та monocar, можна зробити висновок, що даний додаток має наступні переваги: відсутність сервісних зборів, наявність відгуків (скарг), розроблений за допомогою сучасних технологій.

Даний веб-додаток може бути використаний як повноцінна платформа для пошуку спільних поїздок.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дослідження blablacar.
URL: https://blog.blablacar.com.ua/newsroom/novini/poputniki-spriiaiut-dorozhnii-distciplini-rezultati-doslidzhennia-ta-infografika (дата звернення: 15.01.2024).

2. Information about C#. URL: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/overview (дата звернення: 17.02.2024).

3. Andrew Troelsen, Philip Japikse. Pro C# 7: With .NET and .NET Core. Apress, 2018. 1328 p.

4. Jeffrey Richter. CLR via C#. Microsoft Press, 2013. 894 p.

5. Introduction to .NET. URL: https://learn.microsoft.com/en-us/dotnet/core/introduction (дата звернення: 17.02.2024).

6. MVC Architecture – Detailed Explanation.
URL: https://www.interviewbit.com/blog/mvc-architecture/ (дата звернення: 20.03.2024).

7. Bootstrap Documentation. URL: https://getbootstrap.com/docs/5.3/getting-started/introduction/ (дата звернення: 22.03.2024).

8. About Entity Framework Core. URL: https://learn.microsoft.com/en-us/ef/core/ (дата звернення: 27.03.2024).

9. MySQL Documentation. URL: https://dev.mysql.com/doc/ (дата звернення: 27.03.2024).

10. Інформація про бази данних.
URL: https://www.oracle.com/ua/database/what-is-database/ (дата звернення: 27.03.2024).

11. Ben Forta. SQL in 10 Minutes a day. Sams Publishing, 2020. 256 p.

12. Anthony Debarros. Practical SQL. No Starch Press, 2018. 392 p.

ДОДАТКИ

Додаток А

Код UserController

```
namespace WebApplication1.Controllers
{
[Authorize]
public class UserController : Controller
   {
      private readonly ApplicationContext _context;

      public UserController(ApplicationContext context)
      {
         _context = context;
      }

         [AllowAnonymous]
         public IActionResult Index(string returnUrl)
      {
         ViewData["RetUrl"] = returnUrl;
         return View();
      }

         [AllowAnonymous]
         public async Task<IActionResult> LoginAsync([Bind(Prefix = "l")] LoginViewModel
model,string retUrl) {
         if (!ModelState.IsValid)
         {
            return View("Index", new UserViewModel
            {
               LoginViewModel = model
            });
         }

         var user = await _context.Users.FirstOrDefaultAsync(u => u.Login == model.Login &&
u.Password == model.Password);

         if (user == null) {

            ViewBag.Error = "Неправильний логін і(або) пароль";
            return View("Index", new UserViewModel
            {
               LoginViewModel = model
            });
         }

         await AuthenticateAsync(user);

         if (retUrl!="empty")
```

```
        {
            return Redirect(retUrl);
        }

        return RedirectToAction("Index","Trip");
    }

        [AllowAnonymous]
        private async Task AuthenticateAsync(User user)
    {
        string role;

        if (!string.IsNullOrEmpty(user.Role))
            role = user.Role;
        else
            role = "user";

        var claims = new List<Claim>
        {
            new Claim(ClaimTypes.NameIdentifier, user.Id.ToString()),
            new Claim(ClaimsIdentity.DefaultNameClaimType, user.Login),
            new Claim(ClaimsIdentity.DefaultRoleClaimType, role)
        };

        var id = new ClaimsIdentity(claims, "ApplicationCookie",
ClaimsIdentity.DefaultNameClaimType, ClaimsIdentity.DefaultRoleClaimType);
        await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new
ClaimsPrincipal(id));
    }

                [AllowAnonymous]
                public async Task<IActionResult> RegisterAsync([Bind(Prefix = "r")]
RegisterViewModel model, string retUrl)
    {
        if (!ModelState.IsValid)
            return View("Index", new UserViewModel
            {
                RegisterViewModel = model
            });

        var user = await _context.Users.FirstOrDefaultAsync(u => u.Login == model.Login);

        if (user != null)
        {
            ViewBag.RegisterError = "Користувач з таким логіном уже існує!";
            return View("Index", new UserViewModel
            {
                RegisterViewModel = model
            });
        }
```

```
                        var userEmail = await _context.Users.FirstOrDefaultAsync(u => u.Email ==
model.Email);

                        if (userEmail != null)
                        {
                                ViewBag.RegisterError = "Користувач з такою електронною
поштою уже існує!";
                                return View("Index", new UserViewModel
                                {
                                        RegisterViewModel = model
                                });
                        }

                        user = new User(model.Login, model.Password);
            user.Email = model.Email;
            user.Role = "user";
            user.AmountOfReports = 0;
            user.AmountOfBadReports = 0;

            await _context.Users.AddAsync(user);
            await _context.SaveChangesAsync();

            await AuthenticateAsync(user);

            if (retUrl != "empty")
            {
               return Redirect(retUrl);
            }

            return RedirectToAction("Index", "Trip");
        }

        public async Task<IActionResult> LogoutAsync()
        {
            await HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);
            return RedirectToAction("Index", "Trip");
        }

        [Authorize(Roles = "admin")]
        public async Task<IActionResult> AdminPanel(int showTab = 0, string errorMsg = "")
        {
                        var requests = await _context.Requests.Where(r => r.Status == "On
review").ToListAsync();

                        ViewData["Requests"] = requests;

            ViewData["ShowTab"] = showTab;

            if (errorMsg != "")
            {
               ViewBag.Error = errorMsg;
            }
```

```
                return View();
            }

        public async Task<IActionResult> Profile()
        {
            var user = await _context.Users.FirstOrDefaultAsync(u => u.Id ==
int.Parse(User.FindFirstValue(ClaimTypes.NameIdentifier)));

            var model = new ProfileViewModel(user.Login,user.Email);

            var statuses = await _context.Requests.Where(r => r.UserId == user.Id).Select(r =>
r.Status).ToListAsync();

            string userStatus = "";

            if (user.Role == "user")
            {
                if (statuses.Contains("on review"))
                    userStatus = "on review";
                else if (statuses.Contains("declined"))
                    userStatus = "declined";
            }
            else
                userStatus = "accepted";

            ViewData["DriverModel"] = new
DriverViewModel(user.Name,user.Surname,user.PhoneNumber);

            ViewData["UserStatus"] = userStatus;

            return View("Profile", model);
        }

        public async Task<IActionResult> ChangePassword(ProfileViewModel model)
        {
                var user = await _context.Users.FirstOrDefaultAsync(u => u.Id ==
int.Parse(User.FindFirstValue(ClaimTypes.NameIdentifier)));

                if (!ModelState.IsValid)
            {
                    var statuses = await _context.Requests.Where(r => r.UserId ==
user.Id).Select(r => r.Status).ToListAsync();

                    string userStatus = "";

                    if (user.Role == "user")
                    {
                        if (statuses.Contains("on review"))
                            userStatus = "on review";
                        else if (statuses.Contains("declined"))
                            userStatus = "declined";
```

```
                    }
                    else
                        userStatus = "accepted";

                    ViewData["DriverModel"] = new DriverViewModel(user.Name,
user.Surname, user.PhoneNumber);

                    ViewData["UserStatus"] = userStatus;

                    return View("Profile",model);
        }

        user.Password = model.Password;

        _context.Users.Update(user);
        await _context.SaveChangesAsync();

        return RedirectToAction("Profile");
    }

    public async Task<IActionResult> CreateDriverRequest(DriverViewModel model)
    {
        var userId = int.Parse(User.FindFirstValue(ClaimTypes.NameIdentifier));

        if (!ModelState.IsValid)
        {
            var user = await _context.Users.FirstOrDefaultAsync(u => u.Id == userId);
            var profModel = new ProfileViewModel(user.Login,user.Email);


                    return View("Profile",profModel);
        }

        var userPhone = await _context.Users.FirstOrDefaultAsync(u => u.PhoneNumber ==
model.PhoneNumber);

                if (userPhone != null)
                {
                    var user = await _context.Users.FirstOrDefaultAsync(u => u.Id ==
userId);
                    var profModel = new ProfileViewModel(user.Login, user.Email);
                    ViewBag.Errord = "Такий номер телефону вже зареєстровано";

                    return View("Profile", profModel);
                }

                var fileVal = FileValidation(model.DriverLicense);

        if (fileVal != "")
        {
            var user = await _context.Users.FirstOrDefaultAsync(u => u.Id == userId);
            var profModel = new ProfileViewModel(user.Login, user.Email);
```

```
            ViewBag.Errord = fileVal;

            return View("Profile", profModel);
        }

        var request = new DriverRequest();

        request.Name = model.Name;
        request.Surname = model.Surname;
        request.PhoneNumber = model.PhoneNumber;

        using (MemoryStream ms = new MemoryStream())
        {
            await model.DriverLicense.CopyToAsync(ms);
            request.DriverLicense = ms.ToArray();
        }

        request.UserId = userId;
        request.Status = "on review";

        await _context.Requests.AddAsync(request);
        await _context.SaveChangesAsync();

        return RedirectToAction("Profile");
    }

    private string FileValidation(IFormFile file)
    {
        int maxSize = 5000000; //в байтах (5Мб)
        string[] permittedExtensions = { ".png", ".jpg", ".gif", ".jpeg" };

        var ext = Path.GetExtension(file.FileName).ToLowerInvariant();

        if (string.IsNullOrEmpty(ext) || !permittedExtensions.Contains(ext))
        {
            var extensionStr = "";

            foreach (var str in permittedExtensions)
                extensionStr += str + " ";

            return "Файл повинен мати одне з наступних розширень: " + extensionStr;
        }

        else if(file.Length> maxSize)
        {
            return "Розмір файлу не повинен перевищувати 5Мб:";
        }

        return "";
    }

    [Authorize(Roles = "admin")]
```

```
        public async Task<IActionResult> DeclineRequest(int id)
    {
        var req = await _context.Requests.FirstOrDefaultAsync(r => r.Id == id);

        if (req != null)
        {
            req.Status = "declined";

            _context.Requests.Update(req);
            await _context.SaveChangesAsync();
        }

        return RedirectToAction("AdminPanel", new { showTab = 1 });
    }

        [Authorize(Roles = "admin")]
        public async Task<IActionResult> AcceptRequest(int id)
        {
            var req = await _context.Requests.FirstOrDefaultAsync(r => r.Id == id);
        var user = await _context.Users.FirstOrDefaultAsync(u => u.Id == req.UserId);

        if (req != null)
        {
            req.Status = "accepted";

            user.Role = "driver";
            user.Name = req.Name;
            user.Surname = req.Surname;
            user.DriverLicense = req.DriverLicense;
            user.PhoneNumber = req.PhoneNumber;

            _context.Requests.Update(req);
                        _context.Users.Update(user);
                        await _context.SaveChangesAsync();
        }


        return RedirectToAction("AdminPanel", new { showTab = 1 });
    }

        [Authorize(Roles = "admin")]
    public async Task<IActionResult> AddCity(string newCityName)
    {
        var cities = await _context.Cities.Select(c => c.Name).ToListAsync();

        var regExp = new Regex("^[а-яґєіїА-ЯҐЄІЇ]+$");

        if (newCityName == null || newCityName.Count() < 3 || newCityName.Count() > 25 ||
!regExp.IsMatch(newCityName))
                {
            return RedirectToAction("AdminPanel", new { errorMsg = "Назва міста повинна бути
довжиною від 3 до 25 символів і містити тільки кирилицю" });
```

```
        }

                else if (cities.Contains(newCityName))
        {
            return RedirectToAction("AdminPanel", new { errorMsg = "Назва цього міста вже є у
базі" });
        }

                var city = new City();
        city.Name = newCityName;

                await _context.Cities.AddAsync(city);
                await _context.SaveChangesAsync();

                return RedirectToAction("AdminPanel");
            }

    [Authorize(Roles = "admin")]
    public async Task<IActionResult> UnbanDriver(string login)
    {
        var user = await _context.Users.FirstOrDefaultAsync(u => u.Login == login);

        if (user != null && user.isBanned == true)
        {
            user.AmountOfBadReports = 0;
            user.isBanned = false;

            _context.Users.Update(user);
            await _context.SaveChangesAsync();
        }

        else
        {
            return RedirectToAction("AdminPanel", new { showTab = 2 , errorMsg = "Такого
користувача не існує або він не заблокований" });
        }

        return RedirectToAction("AdminPanel", new { showTab = 2 });
    }

            [Authorize(Roles = "admin")]
            public async Task<IActionResult> GetDriverReports(string login)
            {
                var user = await _context.Users.FirstOrDefaultAsync(u => u.Login ==
login);

                if (user != null){

                    var reports = await _context.Reports.Where(r => r.ReportedId ==
user.Id).ToListAsync();
                    ViewData["Reports"] = reports;
        ViewData["ShowTab"] = 2;
```

```
                        return View("AdminPanel");
                    }

                    else
                    {
            return RedirectToAction("AdminPanel", new { showTab = 2, errorMsg = "Такого користувача не існує" });
        }
                }
        }
}
```