НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____2023 р.

Дипломний проект
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інформаційне забезпечення робототехнічних систем»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Система пошуку та реєстрації точок прокату автомобілів»

Виконав:
студент IV курсу, групи ІК-92
Куц Геннадій Євгенович _____

Керівник:
доцент, канд. техн. наук, доцент,
Богданова Наталія Володимирівна _____

Рецензент:
Старший викладач кафедри ІПІ
Марченко Олена Іванівна _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційне забезпечення роботехнічних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____2023 р.

ЗАВДАННЯ

на дипломний проект студенту

Куцу Геннадію Євгеновичу

1. Тема проекту «Система пошуку та реєстрації точок прокату автомобілів», керівник проекту Богданова Наталія Володимирівна, кандидат наук, доцент, затверджені наказом по університету від «31» травня 2023 р. № 2101-с.

2. Термін подання студентом проекту: 11.06.2023 р.

3. Вихідні дані до проекту:

Програмне забезпечення на клієнт-серверній архітектурі. Мова програмування JavaScript, середовище розробки Visual studio cod, обрані технології – Node.js, Expressjs, бібліотеки React.js та Mobx, Posgrees orm Sequelize.

4. Зміст пояснювальної записки Огляд предметної області, Опис технологій, Розробка додатку, Висновки

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо)

Діаграма Бази даних, Діаграма адмін можливостей, Діаграма возможностей пользователя Діаграма послідовності

6. Дата видачі завдання «10» вересня 2023 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	Затвердження теми проекту	17.04-21.04	
2	Вивчення та аналіз завдання	24.04-28.04	
3	Розробка архітектури та загальноїструктури системи	01.05-05.05	
4	Розробка структур окремихпідсистем	08.05-12.05	
5	Програмна реалізація системи	15.05-19.05	
6	Оформлення пояснювальної записки	20.05-22.05	
7	Захист програмного продукту	23.05-09.06	

Студент Геннадій КУЦ

Керівник Наталія БОГДАНОВА

АНОТАЦІЯ

Куц Г.Є. Система пошуку та реєстрації точок прокату автомобілів. КПІ ім. Ігоря Сікорського, Київ 2023.

Проєкт містить 96 с. тексту, 41 рисунок, посилання на 13 літературні джерела, додатки та 4 конструкторських документів.

Ключові слова: Faq, HTML, trc-20, DOM, MVC, middleware, Pern JSON, RDBMS, SQL.

Об’єктом розробки є система пошуку та реєстрації точок прокату автомобілів.

Мета проєкту: надати користувачам можливість, не тільки обирати автомобілі для прокату із існуючого списку, але й додавати оголошення про здачу в оренду їх власного автомобіля.

Для досягнення цієї мити було проведено дослідження наявних веб-пропозицій та аналіз обраного стеку PERN. На основі отриманих результатів був розроблений веб-сайт з серверною частиною і клієнтською частиною с використанням архітектурного підходу MVC.

SUMMARY

Куц Г.Є. System for Searching and Registering Car Rental Points. Igor Sikorsky Kyiv Polytechnic Institute, Kyiv, 2023.

The project consists of 62 pages of text, 30 figures, references to 15 literary sources, appendices, and 4 design documents.

Keywords: Faq, HTML, trc-20, DOM, MVC, middleware, Pern JSON, RDBMS, SQL.

The object of development is a system for searching and registering car rental points.

The goal of the project is to provide users with the ability not only to choose cars for rent from an existing list but also to add advertisements for renting their own car.

To achieve this goal, research was conducted on existing web offerings and analysis of the chosen PERN stack. Based on the obtained results, a website was developed with a server-side and client-side using the MVC architectural approach.

Номер рядка	Формат	Позначення	Найменування	Кільк. аркушів	Номер екзем.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІК92.120БАК.005 ПЗ	Пояснювальна записка	62		
6	А3	ІК92.120БАК.005 Д1	Діаграма адмін можливостей	1		
7						
8	А3	ІК92.120БАК.005 Д2	Діаграма можливостей	1		
9			користувача			
10	А3	ІК92.120БАК.005 Д3	Діаграма Бази Даних	1		
11						
12	А3	ІК92.120БАК.005 Д4	Діаграма послідовності	1		
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						
26						
27						
28						

Зм.	Аркуш	№ докум.	Підпис	Дата	ІК92.120БАК.005 ТП		
Розроб.		Куц Г.Є.			Система пошуку та реєстрації точок прокату автомобілів Відомість проекту	Літ. Т	Аркуш 1 / Аркушів 1
Керівн.		Богданова Н.В.				КПІ ім. Ігоря Сікорського Група ІК-92	
Затв.							

Пояснювальна записка
до дипломного проекту
на тему: «Система пошуку та реєстрації точок прокату автомобілів»

Київ – 2023 рік

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

trc-20: блокчейне TRON

Dom Document Object Model «объектная модель документа».

Mvc Model-View-Controller (Модель-Представлення-Контролер).

Middleware: Сполучне програмне забезпечення.

PERN: PostgreSQL, Express.js, React і Node.js,

PostgreSQL - свободная объектно-реляционная система

JSON: (JavaScript Object Notation)

RDBMS: Relational Database Management System (система управління реляційними базами даних

SQL: (Structured Query Language) Мова Структурованих Запитів

CSS: (Cascading Style Sheets Каскадні таблиці стилів

JWT: (JSON Web Token) апаратний токен

HTTP: (Hypertext Transfer Protocol)

ВСТУП

Галузь автомобільної оренди стала для багатьох людей невід'ємною частиною їхнього повсякденного життя. У сучасному світі особливо з огляду на мобільності та швидкий розвиток технологій. Можливість орендувати автомобіль на певний період дозволяє людям вільно переміщатися, особливо в ситуаціях, коли власний автомобіль недоступний або незручний. Будь то ділова поїздка, сімейна відпустка або просто потреба в автомобілі на кілька днів. Все це дає людині гнучкість та свободу.

Однак, зі зростанням попиту в цій галузі виникають нові виклики, пов’язані з пошуком та реєстрацією точок прокату автомобілів, які найкраще відповідають вимогам та перевагам клієнтів. У сучасному інформаційному суспільстві, де користувачі звикли до миттєвого доступу до інформації, існує необхідність ефективної системи пошуку та реєстрації точок прокату автомобілів.

Тому метою даного проекту є розробка веб-сайту з інноваційною системою пошуку та реєстрації точок прокату автомобілів. Така система надає користувачеві зручну та ефективну платформу, що значно спростить процес пошуку точок доступних для прокату автомобілів. Однією з ключових особливостей системи що розробляється стане передова система пошуку. Користувачі будуть мати змогу задавати різноманітні параметри та критерії пошуку, включаючи місце розташування, тип автомобіля, дати та інші вимоги для того щоб обрати найбільш привабливі точки прокату автомобілів. Алгоритми сортування та рекомендації допоможуть користувачеві обрати найбільш оптимальну пропозицію, яка буде відповідати його вимогам та задовольняти його потреби.

Ця розробка стане корисною як для клієнтів, яким потрібна оренда автомобіля, так і для компаній які надають послуги з прокату автомобільного транспорту. Клієнти зможуть заощаджувати час та зусилля, які необхідно витратити на пошук та вибір відповідної точки прокату, а компанії отримають зручну платформу для залучення нових клієнтів та керування власним автомобільним парком.

У подальшому будуть розкриті подробиці проектування та розробки цієї веб-системи, включаючи функціональні вимоги, архітектуру, основні можливості, технології а також план реалізації проекту.

Актуальні проблеми, пов’язані з системою пошуку та реєстрацією точок прокату автомобілів, є важливими аспектами, які необхідно визначити і врахувати під час розробки веб-сайту.

1. Невідповідальність пропозицій. Клієнти можуть зіткнутися з ситуацією, коли інформація про точки прокату на різних веб-сайтах або платформах не відповідає дійсності. Така невідповідність може бути пов’язана із не коректною інформацією про наявність автомобілів, зміною тарифів або умов оренди. Ці помилки викликають скарги, негативні емоції та незадоволеність клієнтів, тому важно мати систему, котра буде регулярно оновлювати інформацію та підтримувати актуальність даних про точки прокату автомобілів.

2. Складнощі при зміні або скасуванні бронювання. Деякі компанії мають жорсткі правила щодо зміни або скасуванні бронювання, це викликає незручності для клієнтів, яким потрібно внести зміни до своїх планів (скоригувати маршрут, час оренди та інше).

3. Несумісність систем та інтерфейсів. Різноманітні компанії з прокату автомобілів можуть використовувати різні системи та інтерфейси для реєстрації і бронювання автомобілів. Ця система створює складності для користувачів, особливо якщо вони повинні використовувати різні платформи та проходить процедури авторизації окремо для кожної компанії. Інтеграція всіх систем та створення уніфікованого інтерфейсу дозволить вирішити цю проблему та зробити процес реєстрації та бронювання більш зручним для користувачів.

Вирішення цих актуальних проблем є ключовим аспектом розробки веб-сайту системи пошуку та реєстрації точок прокату автомобілів. Враховуючи всі ці фактори, ми прагнемо створити систему, яка буде забезпечувати зручність, прозорість та надійність для клієнтів. Система повинна задовольнити їх потреби в оренді автомобілів дозволяючи легко знаходити та реєструватися в точках прокату, які відповідають їх вимогам.

1 ДОСЛІДЖЕННЯ ІСНУЮЧИХ ВЕБ-СИСТЕМ

1.1 Аналіз існуючих систем

Перед початком розробки веб-сайту були проведені дослідження схожих існуючих веб-додатків. Були визначені основні технології, використані для їх розробки, а також визначені переваги та недоліки цих систем.

Були розглянуті наступні веб-системи: Car.renta.ua та Nars.car

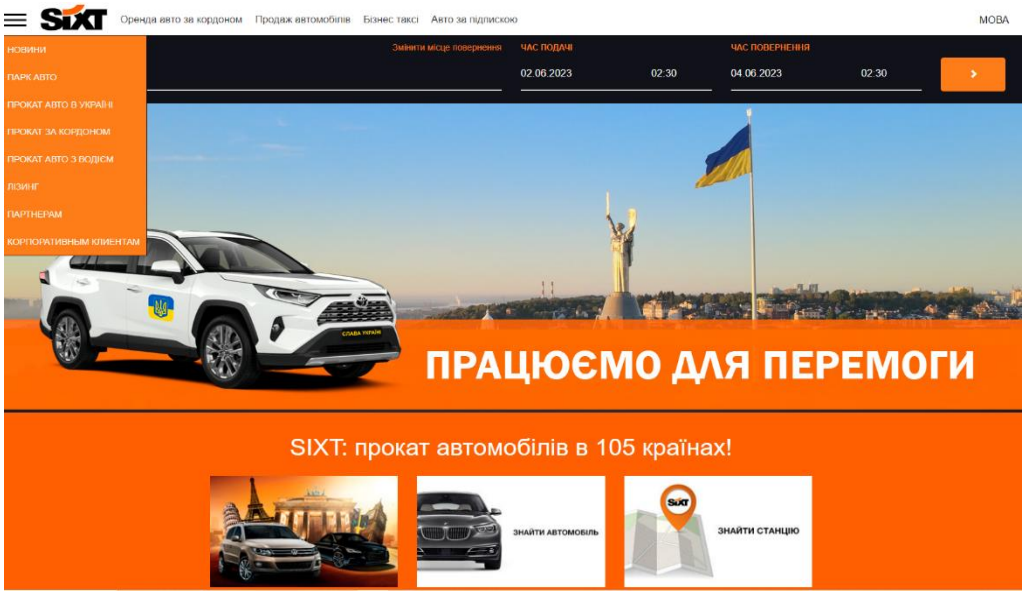

Рисунок 1.1 — Головна сторінка сайту

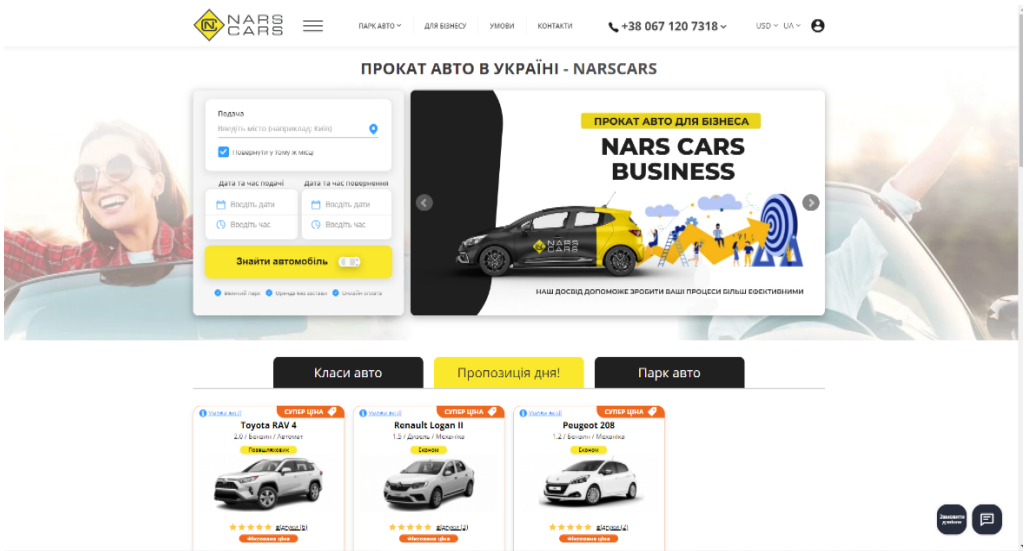

Рисунок 1.2 — Головна сторінка сайту

1.1.1 Переваги та недоліки веб-сайту Car.renta.ua

Розглянемо переваги веб-сайту Car.renta.ua

- Дизайн. Привабливий дизайн Веб-сайту. Гармонійне поєднання палітри кольорів, шрифтів, макета створює професійний вигляд та приємне візуальне враження.
- Навігація. Верхня панель сайту містить основні розділи, такі як: "Оренда автомобіля", "Послуги", "Партнери", "Акції", "Про нас". Це забезпечує зручну навігацію та покращує пошук потрібної інформації. Користувачі легко знайдуть необхідні розділи та швидко отримують доступ до потрібної інформації.
- Інформація про послуги. Веб-сайт надає детальну інформацію про різноманітні послуги які надає веб-ресурс. Клієнти мають змогу ознайомиться з різними типами автомобілів, умовами оренди, тарифами та додатковими послугами. Ця функція допомагає користувачам прийняти проінформоване рішення щодо вибору автомобіля та умов оренди.
- Форма бронювання. На головній сторінці сайту розташована зручна форма бронювання, яка дозволяє користувачу легко обрати дату та місце отримання автомобіля. Такий підхід забезпечує швидкий та зручний процес бронювання, при цьому зберігає простоту використання та інтуїтивно зрозумілий інтерфейс.

Розглянемо недоліки веб-сайту Car.renta.ua

- Навігація. Панель навігації має проблеми із переміщенням по сторінці. Наприклад: якщо прокрутить сторінку вниз до контактів, бачимо, що панель навігації не є динамічною, що ускладнює швидкий перехід по сторінкам для користувача.
- Мобільна адаптивність. Відсутність мобільної адаптивності - це серйозний недолік. У сучасному світі люди повсемістно використовують мобільні пристрої для отримання інформації. Тому ця функція дуже важлива, сайт повинен бути повністю адаптовано для перегляду на різних мобільних пристроях. Правильна мобільна адаптація покращує взаємодію з користувачами та забезпечує комфорт при використанні сайту.

- Відсутність розділу FAQ. Додавання цього розділу покращує загальну інформаційну підтримку клієнтам. Це буде дуже корисно для користувачів сайту, оскільки наддасть потенційні відповіді на запитання клієнтів, щодо умов прокату, вимог до документів, платежів та інше.

- Відгуки та рейтинги. Відображення відгуків та рейтингів клієнтів на сайте може збільшити кількість споживачів, допомогти переконати нових клієнтів в якості послуг Sixt. Розміщення реальних відгуків попередніх клієнтів створить довіру та підтвердить надійність компанії.

- Обмежена фільтрація при виборі автомобіля. Необхідно покращити фільтрацію, при виборі автомобіля користувачі повинні мати змогу сортувати пропозицію за ціною (от дешевого до дорогого), за маркою автомобіля та бачити розміщення пунктів прокату на мапі.

- Оренда додаткового обладнання. Додати можливості оренди додаткового обладнання. Наприклад: дитячі крісла, GPS-навігатори, зимові шини та інше. Така додаткова послуга дозволить клієнтам обирати необхідні аксесуари для прокатного автомобіля, що створить конкурентну перевагу до інших веб-ресурсів.

- Послуги водія. Пропозиція послуги досвідчений водій – це буде дуже корисно для клієнтів, які віддають перевагу не керувати автомобілем самостійно. Це особливо важливо для туристів або людей, які не знайомі з місцевою дорожньою інфраструктурою.

- Доставка та повернення автомобіля. Додавання послуги доставки та повернення автомобіля до зазначеного місця може бути зручним для клієнтів, особливо якщо вони не можуть відвідати офіс Sixt для отримання або здачі автомобіля.

- Страхування та захист. Крім базового страхування, сайт може запропонувати додаткові опції страхування та захисту автомобіля, такі як повне покриття від пошкоджень, страхування від крадіжки та інші додаткові гарантії.

- Відсутність інформації про способи оплати.

1.1.2 Переваги та недоліки веб-сайту Nars.car

Розглянемо переваги веб-сайту Nars.car

- Дизайн та користувацький інтерфейс. Сайт має сучасний та привабливий дизайн. Використання світлих відтінків та добре організований макет роблять його навігацію відносно простою та приємною для користувачів.
- Навігація. Верхнє меню містить основні розділи, такі як "Головна", "Про нас", "Автопарк", "Умови оренди", "Контакти" та "Бронювання". Це допомагає користувачам швидко знайти потрібну інформацію та перейти до потрібного розділу сайту.
- Умови прокату автомобілів. Інформація про умови прокату автомобілів представлена дуже грамотне через випадаючи списки, що робить її компактною та інформативною. Це дозволяє клієнтам швидко ознайомитися з правилами та вимогами прокату.
- Автопарк. Сайт надає інформацію про доступні автомобілі для оренди. Кожен автомобіль супроводжується зображенням, описом та характеристиками, що допомагає клієнтам вибрати відповідний автомобіль та отримати докладну інформацію про нього.
- Відгуки. На сайті реалізовані відгуки клієнтів, які відображаються у вигляді каруселі на головній сторінці. Це дозволяє відвідувачам швидко ознайомитися з думкою інших клієнтів та створює довіру до компанії.
- Контактна інформація. На сайті надана інформація про контакти компанії, що полегшує зв'язок з клієнтами. Надання адреси, номера телефону і електронної пошти допомагає користувачам швидко знайти способи зв'язку з командою Sixt.

Розглянемо недоліки веб-сайту Nars.car.

- Мобільна адаптивність. Мобільна адаптивність відіграє важливу роль у сучасному інтернет-просторі, оскільки багато користувачів віддають перевагу використанню мобільних пристроїв для доступу в Інтернет. Відсутність мобільної версії сайту може призвести до втрати клієнтів.

- Додаткові функції. Додаткові функції на сайті, не повністю реалізовані, це обмежує його привабливість для клієнтів. Наприклад: надання інформації про можливість оренди автомобіля з водієм або пропозиції різноманітних послуг всередині салону автомобіля можуть привернути додатковий інтерес з боку клієнтів. Реалізація таких функцій розширить можливості і зручність використання сайту, що може призвести до збільшення клієнтської бази і підвищення рівня задоволення клієнтів.
- Відсутність розділу FAQ. Відсутність розділу FAQ (часто задавані питання) на сайті обмежує інформаційну підтримку для клієнтів. Додавання розділу FAQ з відповідями на поширені питання про умови оренди автомобіля, вимоги, оплату та інші аспекти може значно покращити загальну інформаційну підтримку для клієнтів і скоротити потребу в індивідуальному зв'язку з операторами.
- Додаткові послуги. Сайт може запропонувати додаткові послуги, такі як оренда додаткового обладнання (навігаційні системи, дитячі крісла тощо), послуги водія або доставка та повернення автомобіля. Це дозволить клієнтам отримати більш гнучкий та зручний досвід оренди автомобіля.
- Система фільтрів. Більш гнучка система фільтрів при виборі автомобіля, надавання інформації про доступність автомобілів у різних цінових категоріях та відображення їх розташування на карті пунктів прокату. Такі функції допоможуть користувачам швидко і зручно вибрати підходящий автомобіль та місце отримання.
- Важним аспектом є також надання інформації про способи оплати на сайті. Клієнти очікують прозорості та зручності при оплаті оренди автомобіля, тому варто забезпечити інформацію про доступні способи оплати та можливість онлайн-оплати на сайті Sixt.

1.2 Постановка задачі для розробки веб-сайту

Після аналізу існуючих рішень стає очевидним, що ці веб-сайти мають схожі принципи роботи, але також мають ряд недоліків, які не відповідають вимогам сучасних сайтів. У обох системах виникають наступні проблеми: неповне

представлення необхідної інформації про доступні автомобілі, відсутність ефективного механізму взаємодії з клієнтами для швидкого вирішення питань, а також відсутність системи оплати.

Після усвідомлення проблем конкурентів, реалізація цього проекту буде розбита на два основних етапи:

а) Розгляд обраного стеку технологій "Perm" та його додаткових функцій, а також надання повної інформації про його можливості.

1. технології Perm;
2. можливості технології Perm, приклади базового написання коду;
3. переваги технології Perm;
4. розбір додаткових технологій.

б) Реалізація.

1. FRONT
2. BACEND

1.2.1 Функціональні задачі

При розробці проекту веб-сайту реалізовані наступні функціональні задачі:

Фронтенд веб-сайту: включає в себе реалізація детального та актуального опису автомобілів, включаючи технічні характеристики та ціни. Забезпечення доступності різних умов прокату. Сайт містить інформацію про різні плани та умови прокату автомобілів, включаючи тривалість прокату, ціни за годину, день або тиждень, вимоги до орендаря, наявність додаткових послуг тощо.

Механізм взаємодії з клієнтами: впроваджено ефективний механізм зворотного зв'язку, наприклад, чат-систему або розділ "Часті запитання" (FAQ), щоб клієнти могли задавати питання та отримувати оперативні відповіді. Такий підхід допоможе підвищити задоволеність клієнтів та поліпшити якість обслуговування.

Система оплати: важливо забезпечити зручний та безпечний спосіб оплати послуг оренди автомобілів. Розробка системи онлайн-платежів, яка дозволить клієнтам безпечно та зручно здійснювати оплату, буде значним покращенням.

Розширений пошук: створення зручної системи пошуку, яка дозволяє користувачам фільтрувати результати за різними параметрами, такими як місцезнаходження, тип автомобіля, тривалість оренди та інші критерії. Це допоможе клієнтам швидко знаходити відповідні прокатні пункти.

Облікові записи клієнтів: можливість створення облікових записів для клієнтів, які дозволять їм зберігати свої уподобання, отримувати персоналізовані рекомендації та спростити повторні бронювання.

Інформація про умови прокату: надання докладної інформації про правила та умови прокату, включаючи додаткові платежі, обмеження за віком, політику палива та інші важливі деталі. Це допоможе клієнтам прийняти обґрунтоване рішення та уникнути неприємних несподіванок.

1.2.2 Додатковий функціонал веб-сайту

Інтеграція з картами та навігацією. Однією з ключових функцій нашої системи буде інтеграція з популярними картами та навігаційними сервісами, такими як Google Maps. Це дозволить користувачам легко знаходити обрані пункти прокату автомобілів та отримувати детальні інструкції про маршрут. Завдяки цій покращеній функціональності клієнти зможуть економити час і впевнено планувати свої поїздки.

Розширені можливості для приватних власників автомобілі. Ми прагнемо пропонувати можливість не лише компаніям, але й приватним особам здавати свої автомобілі в оренду через нашу систему. Для цього передбачається функція створення особистого кабінету, де користувачі зможуть розміщувати інформацію про свої автомобілі та встановлювати власні умови оренди. Це дозволить максимально ефективно використовувати наявні ресурси та створити додаткові можливості для заробітку.

Покращений пошук і фільтрація. У проекті зручність і точність пошуку є важливими аспектами для клієнтів при виборі пункту прокату автомобілів. Тому в систему буде вбудована розширена функція пошуку та фільтрації, яка дозволить користувачам уточнити свої вимоги і отримати найбільш підходящі результати. Критерії фільтрації можуть включати тип автомобіля, місцезнаходження, ціновий діапазон, доступність певних послуг та інші параметри.

Оплата через TRC20. Ураховуючи стрімкий розвиток крипто-валют та інноваційність, ми плануємо впровадити в нашу систему оплату через стандарт TRC20. Це дозволить нашим клієнтам здійснювати транзакції з використанням популярних крипто-валют, таких як Bitcoin, Ethereum та інші. Впровадження такого підходу забезпечить додаткову гнучкість та безпеку в процесі оплати, а також приверне нових користувачів, які активно використовують крипто-валютні технології.

Оплата через TRC20 має кілька переваг :

По-перше, вона забезпечує швидкі та надійні транзакції завдяки використанню сучасних блокчейн-технологій. Це дозволяє уникнути затримок та проблем, пов'язаних з обробкою платежів через традиційні платіжні системи.

По-друге, оплата крипто-валютою дозволяє користувачам зберігати свою фінансову приватність і уникати надання банківської інформації або особистих даних. Це особливо актуально в епоху підвищеної стурбованості щодо захисту особистої інформації.

По-третє, використання крипто-валютних платежів може привернути увагу нових клієнтів, які активно цікавляться та використовують цифрові валюти у своєму повсякденному житті.

Висновок до розділу: У даному розділі було проведено аналіз існуючих систем для оренди автомобілів і сформульовано завдання щодо розробки нової системи з додатковими функціями. Під час аналізу двох систем у першому підрозділі виявлено, що вони мають ряд недоліків і не відповідають сучасним технологіям та можливостям. Тому необхідно розробити нову систему, яка враховуватиме сучасні вимоги та потреби користувачів. У другому підрозділі

поставлено завдання розробити фронтенд-частину системи, в якій будуть усунуті виявлені недоліки попередніх систем, а також доданий новий функціонал для поліпшення користувацького досвіду.

2 ОПИС ТЕХНОЛОГІЙ

Перед початком розробки потрібно розібратися з обраним стеком PERN - це комбінація технологій, яка дозволяє розробникам створювати повноцінні веб-додатки, використовуючи JavaScript як для клієнтської, так і для серверної частини. Він поєднує потужні інструменти для роботи з даними (PostgreSQL), розробки серверної частини (Express.js) та побудови користувацьких інтерфейсів (React.js), а також забезпечує гнучке середовище виконання (Node.js). Цей стек забезпечує ефективну розробку, масштабованість та надійність веб-додатків. Також розглянемо технологію для створення бота. Перейдемо до першої половини стеку.

2.1 Стек для FRONT-END частини

Розглянемо технології, які будуть використовуватися для написання фронтенд-частини. У розробці клієнтської частини використовуються такі класичні технології, як HTML, CSS, Bootstrap, React та JavaScript. Комбінація цих технологій дозволяє створювати сучасні та реактивні фронтенд-додатки. HTML та CSS забезпечують структуру та стилізацію, Bootstrap надає готові компоненти та шаблони для швидкої розробки, React дозволяє створювати перевикористовувані компоненти, а JavaScript додає інтерактивність та динамічну поведінку. Це потужний набір технологій, який забезпечує розробку якісних користувацьких інтерфейсів та поліпшує взаємодію з користувачами.

Давайте розглянемо кожен з інструментів більш детально. Розглянемо технології, які будуть використовуватися для написання фронтенду. У розробці клієнтської частини використовуються такі класичні технології, як: HTML, CSS, Bootstrap, React, JavaScript.

Поєднання цих технологій дозволяє створювати сучасні та реактивні фронтенд-додатки. HTML і CSS забезпечують структуру та стилізацію, Bootstrap надає готові компоненти та шаблони для швидкої розробки, React дозволяє створювати компоненти, які можна повторно використовувати, а JavaScript додає

інтерактивність та динамічну поведінку. Це потужний стек технологій, який забезпечує розробку якісних користувацьких інтерфейсів та покращує взаємодію з користувачами. Розглянемо кожен інструмент детальніше.

2.1.1 HTML

В загалі HTML є мовою розмітки гіпертексту, яка відіграє важливу роль у веб-розробці. Однак варто розуміти, що це не зовсім повноцінна мова програмування, оскільки вона не здатна виконувати обчислення або виконувати певний алгоритм. Більшість сучасних веб-сайтів створені саме за допомогою HTML. Вона дозволяє створювати та організовувати вміст веб-сторінок, забезпечуючи їх структуру та семантику. Головним чином використовуються елементи та теги для визначення різних компонентів сторінки та їх візуального представлення у браузері.

Переваг є його універсальність і доступність. HTML-сторінки можуть бути відкриті на будь-якому пристрої, що підтримує веб-браузер, включаючи комп'ютери, смартфони і планшети.

2.1.2 CSS

На сьогоднішній день неможливо уявити веб-сайт, який не використовує CSS. Це мова стилізації та оформлення, яка використовується для оформлення зовнішнього вигляду і стилізації веб-сторінок. В поєднанні з HTML, CSS дозволяє контролювати різні аспекти дизайну, такі як елементи, кольори, шрифти, розміри, відступи та багато іншого.

Переваги використання CSS:

- Роздільність структури та оформлення. CSS дозволяє відокремити структуру HTML-документа від його візуального оформлення. Це спрощує розробку та обслуговування веб-сайту, оскільки зміни в оформленні можуть бути внесені до CSS-файлу без необхідності змінювати HTML-код.

- Гнучкість та контроль. CSS надає широкий спектр можливостей для оформлення елементів на веб-сторінці. Ви можете контролювати кольори, шрифти, розміри, розташування елементів та багато іншого. CSS також підтримує медіа-запити, що дозволяє створювати адаптивний дизайн для різних пристроїв та екранів.

Недоліки використання CSS:

- Кросс-браузерна сумісність. Різні браузери можуть по-різному інтерпретувати та підтримувати CSS-правила. Це може призвести до відмінностей в відображенні веб-сторінок у різних браузерах, що вимагає тестування та додаткової роботи для забезпечення сумісності.
- Інспектованість. CSS-код можна легко інспектувати та змінювати за допомогою інструментів розробника веб-браузера. Це може становити певні загрози безпеці, оскільки зловмисники можуть модифікувати стилі і вносити зміни на веб-сайті.

2.1.3 Bootstrap

Bootstrap – на сьогоднішній день, це популярний інструментарій для розробки HTML, CSS та JS фреймворк веб-інтерфейсів. Його використання дозволяє значно прискорити та спростити розробку веб-сторінок. А його мануал містить детальні пояснення, та багато прикладів рішень, що полегшують розуміння та використання фреймворку.

Основний підхід, використовуваний в Bootstrap, полягає у розділенні сторінки на 12 стовпців однакової ширини. Це дає можливість гнучко розміщувати елементи на веб-сторінці шляхом визначення кількості стовпців, які займає кожен елемент в залежності від розміру екрану. Такий підхід дозволяє легко створювати адаптивні веб-сайти, які коректне відображатимуться на різних пристроях.

Завдяки використанню готових класів та атрибутів у бібліотеці Bootstrap, можна швидко та легко створити різноманітні елементи веб-інтерфейсу.Це

дозволяє швидко налаштувати вигляд та стиль вашого веб-сайту без необхідності писати власні CSS-стилі з нуля.

2.1.4 JavaScript

JavaScript - це високорівнева, об'єктно-орієнтована мова програмування, яка базується на стандарті ECMAScript. Он використовується для створення динамічних та інтерактивних веб-сайтів. JavaScript підтримується практично більшістю сучасними веб-браузерами і надає можливість взаємодії з користувачем, маніпулювання веб-сторінками та асинхронний обмін даними з сервером.

Причини використання JavaScript в нашому проекті основною технологією:

- Динамічний контент. JavaScript дозволяє створювати динамічний контент. Це включає можливість додавати анімацію, маніпулювати DOM-елементами, валідувати форми та забезпечувати інші інтерактивні можливості для користувачів.
- Взаємодія з користувачем це надає можливість збирати введенні дані від користувача
- Асинхронне програмування. JavaScript дозволяє здійснювати асинхронні запити до сервера без перезавантаження сторінки, що дозволяє забезпечити швидке і ефективне взаємодію з сервером. Це особливо важливо для функціональності, такої як валідація форм для обміну даними завантаження даних з бази даних або взаємодія зі сторонніми API.

2.1.5 React

React - це сучасна відкрита бібліотека мови JavaScript, за допомогою якої можна значно прискорити створення користувацького інтерфейсу (UI). Вона дозволяє розробляти компоненти UI користувацького інтерфейсу, які автоматично оновлюються при зміні даних. Це робить процес створення інтерактивних та реактивних веб-додатків простішим та ефективнішими.

Існує декілька причин, чому React широко використовується та рекомендується для розробки веб-додатків, і вони включають в себе:

- MVC (Model-View-Controller). Основна перевага React - працює тільки з інтерфейсами користувача, це відповідає архітектурному виду MVC шаблону.
- Компонентний підхід. React базується на компонентах, що робить розробку додатків більш модульною та перевикористовуваною
- Ефективний віртуальний DOM. React використовує віртуальний DOM, що дозволяє оптимізувати процес оновлення користувацького інтерфейсу. Замість оновлення кожного елемента окремо.
- Однонаправлений потік даних. У React дані передаються по ієрархії компонентів за допомогою однонаправленого потоку даних, що спрощує відстеження змін даних та управління станом додатка.

2.2 Стек для BAC-END частини

В цілому, серверна частина є основою для розробки потужних та функціональних додатків, оскільки вона забезпечує взаємодію між клієнтською частиною - веб-браузером та базою даних. Ось деякі з основних причин, чому серверна частина є невід'ємною складовою розробки додатків:

- Обробка бізнес-логіки
- Взаємодія з базою даних.
- Забезпечення безпеки.
- Масштабованість та оптимізація:
- Робота з зовнішніми сервісами.

Враховуючи вищезазначені причини, були обрані такі технології для реалізації серверної частини як Node.js, Express.js, PostgreSQL та ORM SEQULIZE. Ці технології надають потужні інструменти для розробки та оптимізації серверної частини додатків, розглянемо їх.

2.2.1 Node.js

Node.js - це середа виконання JavaScript на стороні сервера яка базується на двигуні Chrome V8, яка пропонує кілька переваг:

- Об'єднана мова програмування. Node.js дозволяє використовувати JavaScript як на клієнтській, так і на серверній стороні.
- Багата екосистема модулів. Node.js має широку екосистему модулів і пакетів, доступних через менеджер пакетів npm.

Як працює Node.js:

- Організація подвійного циклу (Event Loop): Node.js базується на подвійно-орієнтованій архітектурі, де всі операції введення/виведення виконуються асинхронно.
- Робота Node.js ґрунтується на неблокуючій (асинхронній) моделі введення/виведення.

2.2.2 Express.js

Express.js - це мінімалістичний і гнучкий веб-фреймворк для Node.js. Він надає набір інструментів і функцій для розробки веб-додатків і API, спрощуючи процес створення серверної частини додатків. Express.js дозволяє розробникам створювати маршрути, обробляти HTTP- запити та генерувати HTTP-відповіді.

Express.js має деякі переваги, які роблять його популярним та бажаним вибором для розробки серверних додатків. Розглянемо переваги Express.js:

- Простота і мінімалізм.
- Гнучкість. Express.js
- Дає можливість створення API:
- Інтеграція з іншими модулями та бібліотеками:

Express.js одним з найбільших популярних фреймворків для розробки веб-приложений на Node.js, але він теж має деякі недоліки, які варто враховувати.

Управління станом. Express.js не надає вбудованого механізму управління станом додатка. Це означає, що вам потрібно буде самостійно реалізувати патерни управління станом або використовувати сторонні бібліотеки, такі як Redux або MobX.

Express.js базується на обробці запитів і відповідей. При отриманні HTTP-запиту на сервер, Express.js шукає відповідний маршрут, який відповідає URL-адресі запиту, і викликає відповідний обробник для обробки запиту. Обробник може повертати HTML-сторінку, JSON-дані або виконувати інші дії, пов'язані з логікою додатка.

Express.js також підтримує використання концепції проміжного програмного забезпечення (middleware).

Проміжне програмне забезпечення - це функції, які виконують операції перед або після обробки запиту і можуть виконувати різні операції, такі як аутентифікація, перевірка доступу, обробка даних, обробка помилок, обробка тіла запиту. Воно дозволяє розробнику налаштовувати його глобальне для всього додатка або застосовувати до конкретних маршрутів.

Обробник помилок в Express може формувати та надсилати користувачеві інформативні повідомлення про помилку, щоб полегшити розуміння виниклої проблеми та надати рекомендації щодо її усунення.

2.2.3 PostgreSQL

Це потужна реляційна база даних з відкритим вихідним кодом. Вона забезпечує надійне та ефективне зберігання даних, а також широкий набір функцій для управління даними та виконання складних запитів. PostgreSQL працює на сервері і надає клієнтам можливість підключатися до бази даних та виконувати операції з даними. В якості серверного додатку PostgreSQL обробляє запити від клієнтів, виконує операції читання та запису даних, забезпечує безпеку, контролює доступ та підтримує цілісність даних

Основні принципи роботи PostgreSQL:

- Клієнт-серверна архітектура: PostgreSQL працює у режимі клієнт-сервер, де сервер надає базу даних та обробляє запити від клієнтів. Клієнти можуть бути різними додатками, які використовують PostgreSQL як сховище даних.
- Мова запитів SQL: PostgreSQL підтримує стандартну мову запитів SQL, яка використовується для створення, зміни та вибірки даних з бази даних. SQL дозволяє виконувати широкий спектр операцій, таких як створення таблиць, вставка, оновлення та видалення записів, агрегація даних та виконання складних запитів.
- Модель реляційної бази даних: PostgreSQL базується на моделі реляційної бази даних, де дані організовані в таблиці з визначеною структурою та зв'язками між ними. Це дозволяє легко зберігати та зв'язувати дані різних типів.
- Транзакційна обробка: PostgreSQL забезпечує транзакційну обробку даних, що означає, що операції з даними можуть бути виконані як один логічний блок роботи. У разі виникнення помилки в середині транзакції, всі зміни можуть бути скасовані, щоб забезпечити цілісність даних.
- Багатокористувацький доступ: PostgreSQL підтримує одночасний доступ кількох користувачів до бази даних. Він надає механізми керування доступом та безпекою, щоб дозволити або обмежити доступ до певних даних та виконання операцій.
- Розширюваність: PostgreSQL надає багато розширень, які дозволяють розробникам додавати нові функціональні можливості та типи даних. Він також підтримує збережені процедури, тригери та користувацькі функції, що робить його гнучким та потужним інструментом для розробки.
- PostgreSQL відома своєю надійністю, продуктивністю, розширюваністю та підтримкою передових функцій, таких як повнотекстовий пошук, географічні запити, обробка JSON. Вона також підтримує транзакційну модель, властивості ACID (атомарність, консистентність, ізольованість та стійкість) та механізми забезпечення цілісності даних. Вона широко використовується в різних типах додатків, від невеликих веб-сайтів до великих підприємств.

Переваги та недоліки та PostgreSQL:

- Масштабованість: PostgreSQL ефективно масштабується як вертикально (збільшення продуктивності шляхом покращення апаратного забезпечення), так і горизонтально (розподіл даних на кілька вузлів).
- Багатофункціональність: PostgreSQL підтримує широкий спектр функцій, включаючи географічні та геометричні типи даних, повнотекстовий пошук, обробку JSON, масиви, XML та багато іншого. Це робить його привабливим для різних типів проектів та використання.
- Відкритий вихідний код та активна спільнота: PostgreSQL є проектом з відкритим вихідним кодом, що означає, що він поширюється безкоштовно та має активну спільноту розробників. Це забезпечує постійний розвиток, підтримку та оновлення бази даних.
- Складність налаштування: Порівняно з деякими іншими системами управління базами даних, PostgreSQL може вимагати більше часу та зусиль для налаштування та початку роботи. Його розширені можливості можуть бути складними для новачків, і він може вимагати додаткових знань та досвіду у адмініструванні баз даних.
- Використання ресурсів: PostgreSQL може вимагати більше ресурсів комп'ютера та пам'яті, особливо при обробці великого обсягу даних. Це може впливати на продуктивність у разі недостатніх ресурсів або неправильної конфігурації.
- Необхідність додаткових модулів для деяких функцій: Хоча PostgreSQL надає багато функцій "з коробки", для деяких розширених можливостей, таких як повнотекстовий пошук або обробка JSON, можуть знадобитися додаткові модулі та налаштування.

2.2.4 Безпека веб-додатку

У сучасному цифровому світі, де дані є одним з найцінніших активів, забезпечення безпеки БД стає критично важливим завданням для компаній.

Вразливість та порушення безпеки даних можуть призвести до серйозних наслідків, включаючи витік конфіденційної інформації, фінансові збитки та пошкодження репутації. Тому вивчення потенційних загроз та використання відповідних заходів безпеки є невід'ємною частиною успішної розробки та експлуатації баз даних.

Ін'єкції SQL та несанкціонований доступ - це дві поширені загрози безпеки, пов'язані з базами даних та серверною частиною додатків.

Розглянемо їх детальніше:

1. Ін'єкції SQL: Ін'єкція SQL - це вид атаки, при якій зловмисник вбудовує шкідливі SQL-команди або фрагменти коду в користувацький ввід, який передається для обробки до бази даних. Метою ін'єкцій SQL є виконання небажаних дій або отримання несанкціонованого доступу до даних в базі даних.

Приклади ін'єкцій SQL можуть включати:

- Вбудовування шкідливого SQL-коду в форму вводу користувачем, що може призвести до виконання небажаних операцій у базі даних.
- Використання ін'єкцій SQL для обходу аутентифікації та отримання доступу до захищених даних.

2. Несанкціонований доступ:

Несанкціонований доступ - це спроба отримати доступ до системи, ресурсів або даних без належної авторизації чи дозволу. Зловмисник може намагатися обійти механізми аутентифікації та отримати несанкціонований доступ до конфіденційних або захищених даних.

Приклади несанкціонованого доступу можуть включати:

- Спроби взлому паролів або використання слабких облікових даних для отримання доступу до системи.
- Перебір паролів або використання вразливостей у системі для обходу механізмів аутентифікації.
- Загрози фізичної безпеки, такі як несанкціонований доступ до серверів або сховищ даних.

Безпека баз даних Забезпечення захисту даних в базі даних та запобігання загрозам безпеки, таким як ін'єкції SQL або несанкціонований доступ, є критично важливими завданнями при розробці серверної частини додатка. Нижче наведено деякі методи та практики, які можуть бути використані для забезпечення безпеки бази даних:

- Параметризовані запити або використання ORM: Використання параметризованих запитів або об'єктно-реляційних відображень (ORM) дозволяє уникнути ін'єкцій SQL. При такому підході дані вставляються в запити через параметри або об'єкти, а не вбудовуються безпосередньо в SQL-запити.

- Валідація та фільтрація введення: Необхідно проводити валідацію та фільтрацію вхідних даних, щоб запобігти впровадження шкідливого коду або некоректних даних. Це може включати використання регулярних виразів, перевірку типів, перевірку довжини тощо.

- Аутентифікація та авторизація: Реалізація системи аутентифікації та авторизації дозволяє керувати доступом до бази даних. Користувачі повинні бути аутентифіковані перед отриманням доступу до захищених даних, а потім авторизовані на основі їх ролей та прав доступу.

- Хешування паролів: При зберіганні паролів користувачів в базі даних рекомендується використовувати хешування та додавання унікального солі. Це забезпечує додатковий рівень захисту у разі витоку даних.

- Захист від CSRF (міжсайтової підробки запиту): Застосування механізмів захисту від CSRF дозволяє запобігти атакам, при яких зловмисник намагається виконати несанкціоновані операції від імені аутентифікованого користувача.

- Обмеження прав доступу: Розподіл прав доступу в базі даних дозволяє запобігти несанкціонованому доступу до конфіденційних даних. Кожен користувач або роль повинні мати лише ті права доступу, які необхідні для виконання їх функцій.

- Оновлення та моніторинг системи: Регулярне оновлення та моніторинг серверної частини додатка дозволяють швидко виправляти виявлені вразливості та запобігати новим загрозам безпеки.

- Шифрування даних: Для особливо конфіденційних даних можна застосовувати шифрування на рівні бази даних або поля, щоб запобігти несанкціонованому доступу до даних навіть при витоку або компрометації бази даних.

- Журналювання та моніторинг: Ведення докладних журналів операцій та моніторинг активності бази даних дозволяють виявляти підозрілу активність і реагувати на неї.

Це лише деякі методи та практики, які можуть бути використані для забезпечення безпеки бази даних, і більшість з наведеного вище вже реалізовано. Важливо розуміти, що безпека є постійним процесом, і потребує постійної уваги до нових загроз і оновлення заходів безпеки з часом.

2.2.5 ORM

ORM (Object-Relational Mapping) - це методика програмування, яка дозволяє розробникам працювати з базою даних, використовуючи об'єктно-орієнтований підхід. ORM-фреймворки дозволяють відображати об'єкти додатка на відповідні таблиці в базі даних, а також виконувати операції читання та запису даних без явного використання мови SQL. ORM спрощує та стандартизує взаємодію з базою даних і полегшує розробку та підтримку додатків.

- Спрощення розробки: ORM-фреймворки полегшують роботу з базою даних, усуваючи потребу вручну писати складні SQL-запити. Вони надають високорівневі методи для створення, читання, оновлення та видалення об'єктів, а також автоматично генерують SQL-запити на основі визначених моделей даних.

- Підвищення продуктивності: ORM-фреймворки оптимізують виконання SQL-запитів, надаючи можливість попереднього завантаження пов'язаних даних, кешування та інших оптимізацій. Вони також можуть автоматично створювати необхідні індекси та оптимізувати структуру бази даних для більш ефективного доступу до даних.

- Переносимість: ORM-фреймворки абстрагують додаток від конкретної реляційної бази даних. Це дозволяє легко перемикатися між різними СУБД без необхідності переписування великої кількості коду. ORM-фреймворки забезпечують міграції схеми бази даних та підтримують стандарти SQL, що полегшує переносимість коду між різними базами даних.

- Безпека: ORM-фреймворки надають механізми для захисту від вразливостей безпеки, таких як SQL-ін'єкції. Вони забезпечують автоматичну обробку користувацького вводу та запобігання SQL-ін'єкціям шляхом надання параметризованих запитів та інших механізмів безпеки.

- Відображення об'єктно-реляційної структури даних: ORM дозволяє відображати об'єкти в базу даних і навпаки, створюючи відповідність між таблицями бази даних та класами або об'єктами в мові програмування. Це дозволяє працювати з даними як з об'єктами, а не окремими записами в таблицях.

- Генерація SQL-запитів: ORM автоматично генерує SQL-запити для виконання операцій з базою даних, таких як створення, читання, оновлення та видалення даних. Розробник повинен лише викликати відповідні методи ORM, і він отримає відповідний SQL-запит без необхідності явного написання SQL-коду.

- Керування взаємозв'язками між об'єктами: ORM дозволяє визначити взаємозв'язки між об'єктами, такі як один-до-багатьох, багато-до-багатьох та інші. ORM автоматично створює відповідні таблиці та відстежує зв'язки між об'єктами, забезпечуючи цілісність даних та зручний доступ до пов'язаних даних.

- Автоматичне створення схеми бази даних: ORM може автоматично створювати структуру бази даних на основі визначених класів або моделей. Це спрощує та прискорює процес створення та оновлення схеми бази даних без необхідності явного створення таблиць та зв’язків.

ORM полегшує роботу з базою даних, спрощує розробку та підтримку додатків, а також покращує переносимість коду між різними базами даних. Він також забезпечує високий рівень абстракції, приховуючи деталі роботи з базою даних і дозволяючи розробнику фокусуватися на бізнес-логіці додатка.

2.2.6 Недоліки ORM

Незважаючи на безліч переваг, ORM також має деякі недоліки. Ось деякі з них:

- Складність налаштування: ORM може бути складним у налаштуванні, особливо при роботі з складними схемами баз даних. Потрібно правильно конфігурувати відповідні відображення класів та відношень між таблицями, щоб ORM виконував запити та операції з даними правильно.
- Надмірність коду: ORM може призвести до надмірності коду в додатку. Деякі ORM вимагають певної структури класів та маппінгу, що може призвести до підвищеної складності та обсягу коду.
- Продуктивність: Використання ORM може призвести до певного зниження продуктивності через додатковий шар абстракції між додатком та базою даних. У деяких випадках прямі SQL-запити можуть бути більш продуктивними, ніж використання ORM.
- Обмеження ORM: ORM іноді може мати деякі обмеження у функціональності порівняно з прямим використанням SQL. Деякі складні або складні запити можуть бути складно виконати за допомогою ORM, і у деяких випадках може знадобитися написання власних SQL-запитів.
- Залежність від ORM: Використання ORM призводить до залежності від конкретної бібліотеки ORM. Це може ускладнити перехід на іншу ORM або пряму роботу з SQL, якщо потрібно змінити або оптимізувати код.

Важливо зазначити, що недоліки ORM можуть бути відносними і залежать від конкретного сценарію використання та вимог проекту. При виборі ORM слід ретельно оцінити його переваги та недоліки, а також порівняти з іншими доступними варіантами роботи з базою даних.

2.2.7 Взаємодія ORM з базою даних

Ось основні кроки взаємодії з базою даних при використанні ORM:

- Визначення моделей: Спочатку необхідно визначити моделі даних, які представляють таблиці бази даних у вигляді класів або об'єктів у мові програмування. Моделі описують структуру та зв'язки даних в базі даних.
- Створення з'єднання: ORM-фреймворк забезпечує механізм для встановлення з'єднання з базою даних. Це може включати налаштування параметрів з'єднання, таких як адреса, ім'я користувача, пароль та порт.
- Виконання операцій CRUD: За допомогою ORM можна виконувати операції створення (Create), читання (Read), оновлення (Update) та видалення (Delete) даних. Для кожної операції ORM-фреймворк надає відповідні методи або функції, які дозволяють взаємодіяти з базою даних за допомогою об'єктів моделей.
- Міграції та синхронізація схеми: ORM-фреймворки надають механізми для автоматичного створення або оновлення структури бази даних на основі визначених моделей. Це включає створення таблиць, індексів, обмежень та інших елементів схеми бази даних.
- Операції запитів: ORM-фреймворки надають спеціальні методи або мову запитів, що дозволяють виконувати запити до бази даних для отримання даних з використанням об'єктно-орієнтованого підходу. Запити можуть бути простими (наприклад, отримання всіх об'єктів певного типу) або складними (наприклад, об'єднання даних з декількох таблиць).
- Керування транзакціями: ORM-фреймворки надають механізми для керування транзакціями, що дозволяє групувати кілька операцій бази даних в одну логічну транзакцію з можливістю відкату змін у разі помилки.
- Обробка помилок та винятків: ORM-фреймворки надають механізми для обробки помилок, пов'язаних з базою даних, і можливості обробки винятків, щоб гарантувати надійність та цілісність даних.

І тепер, після вивчення його переваг, недоліків і взаємодії, перейдемо до вибору ORM-фреймворка

2.2.8 Sequelize

Sequelize - це ORM (об'єктно-реляційне відображення) для Node.js, яке спрощує взаємодію з реляційними базами даних. Воно надає зручний і гнучкий інтерфейс для роботи з даними, абстрагуючи розробника від низькорівневих SQL-запитів і дозволяючи замість цього працювати з об'єктами та класами.

Як працює Sequelize з базою даних:

- Підключення до бази даних. Sequelize дозволяє встановити з'єднання з базою даних, вказавши відповідні параметри, такі як ім'я хоста, порт, облікові дані та ім'я бази даних.
- Визначення моделей. У Sequelize визначаються моделі даних, які представляють таблиці в базі даних. Модель визначає структуру даних, атрибути, зв'язки та обмеження таблиці. Моделі створюються з використанням класів або об'єктів, і кожна модель відповідає певній таблиці в базі даних.
- Створення схеми бази даних. Sequelize може автоматично створювати схему бази даних на основі визначених моделей. Це включає створення таблиць, індексів, обмежень та зв'язків між таблицями. Sequelize надає методи для виконання міграцій, що дозволяють керувати змінами в схемі бази даних.
- Виконання операцій CRUD. За допомогою Sequelize можна виконувати операції створення, читання, оновлення та видалення даних з бази даних. Він надає зручні методи для створення нових записів, отримання даних з бази даних, оновлення існуючих записів та видалення записів. Sequelize автоматично генерує відповідні SQL-запити на основі визначених моделей.
- Запити та фільтрація даних. Sequelize надає можливість формувати складні запити до бази даних за допомогою різних операторів та умов. Це включає вибірку, фільтрацію, сортування, групування та агрегацію даних. Розробник може використовувати методи Sequelize для формування запитів та отримання лише необхідних даних.
- Транзакції та обробка помилок. Sequelize підтримує транзакції бази даних, що дозволяє виконувати кілька операцій як одну атомарну операцію. Це корисно

для забезпечення цілісності даних та збереження їх консистентності при виконанні пов'язаних операцій. Sequelize також надає обробку помилок, включаючи валідацію даних, обробку конфліктів та обробку винятків при виконанні запитів.

Далі розглянемо один із важливих аспектів для нашого додатка - це безпека, і ORM забезпечує безпеку даних у базі даних наступними способами:

- Параметризовані запити. Sequelize використовує параметризовані запити для запобігання SQL-ін'єкціям. Замість вставки значень безпосередньо в SQL-запити, Sequelize замінює їх параметрами, що унеможливлює можливість впровадження шкідливого коду.
- Перевірка даних. Sequelize надає механізми для перевірки даних перед збереженням у базу даних. Ви можете визначити правила валідації для кожної моделі даних, такі як перевірка наявності обов'язкових полів, перевірка формату даних та інші власні правила. Це допомагає запобігти збереженню некоректних даних у базі даних.
- Аутентифікація та авторизація. Sequelize не забезпечує прямо механізми аутентифікації та авторизації, але може інтегруватись з іншими пакетами або фреймворками для забезпечення безпеки доступу до даних. Ви можете використовувати Sequelize для виконання запитів, пов'язаних з аутентифікацією та авторизацією, і забезпечення контролю доступу до бази даних.
- Ролі та дозволи. Sequelize дозволяє визначати різні ролі та дозволи для користувачів бази даних. Ви можете керувати доступом користувачів до певних таблиць або полів, обмежуючи їх можливості вносити зміни або отримувати конфіденційні дані.
- Хешування паролів. Sequelize не надає вбудованих механізмів хешування паролів, але ви можете використовувати додаткові пакети або методи хешування, які надає мова програмування, для безпечного зберігання паролів користувачів у базі даних.
- Шифрування даних. Якщо потрібне додаткове шифрування даних у базі даних, Sequelize може працювати з шифруванням на рівні додатка. Ви можете

шифрувати конфіденційні дані перед збереженням та розшифровувати їх під час вилучення з бази даних.

Важливо зазначити, що безпека даних залежить не тільки від Sequelize, але і від загальної архітектури додатка, мережевої безпеки, доступу до сервера бази даних та інших факторів. Sequelize надає інструменти та практики для безпечної роботи з даними в базі даних, але безпеку слід забезпечити на всіх рівнях додатка.

Висновки до розділу

Ми розглянули стек технологій PERN (PostgreSQL, Express.js, React, Node.js), який є потужним інструментарієм для розробки ефективних та масштабованих веб-додатків.

Node.js є потужним інструментом для розробки серверних застосунків на JavaScript. Він базується на подіях та має асинхронну модель виконання, що дозволяє обробляти багато запитів одночасно без блокування потоків виконання. Node.js став популярним вибором для багатьох розробників, особливо в галузі веб-розробки та мікросервісної архітектури, завдяки своїй відзивчивості та продуктивності.

Express.js є мінімалістичним та гнучким веб-фреймворком для Node.js. Він надає розробникам прості та зрозумілі інструменти для обробки HTTP-запитів, налаштування маршрутизації та роботи з проміжним програмним забезпеченням. Express.js спрощує створення веб-додатків та API, надаючи чіткі та гнучкі механізми для обробки запитів і відповідей. Завдяки проміжному програмному забезпеченню, розробник може додавати зручні функції на різних етапах обробки запитів, таких як аутентифікація, логування або обробка помилок. Express.js спрощує розробку серверних додатків і дозволяє створювати гнучкі та масштабовані API.

При виборі бази даних ми розглянули різні типи даних і їх переваги та недоліки. Реляційні бази даних (RDBMS) були обрані як надійний і широко поширений тип баз даних, що забезпечує строгу структуру даних, можливість

виконання складних запитів та обробку транзакцій. Для взаємодії з базою даних у проекті ми обрали ORM Sequelize, який надає об'єктно-реляційне відображення і спрощує роботу з базою даних за допомогою JavaScript-об'єктів та методів. Sequelize полегшує створення та виконання запитів до бази даних, а також надає механізми безпеки і захисту від атак SQL-ін'єкцій.

При розробці фронтенд-частини проекту ми вивчили HTML, CSS, JavaScript, React та Bootstrap. HTML є основною мовою розмітки для створення структури веб-сторінок. CSS використовується для задання стилів та зовнішнього вигляду елементів на сторінці. JavaScript є мовою програмування, яка використовується для додавання інтерактивності та функціональності на веб-сторінках. React є популярною JavaScript-бібліотекою для розробки користувацьких інтерфейсів, що дозволяє створювати багатокомпонентні та масштабовані веб-додатки. Bootstrap надає набір готових стилів і компонентів для спрощення створення адаптивних та привабливих користувацьких інтерфейсів.

Загалом, використання Node.js з Express.js разом з базою даних PostgreSQL і ORM Sequelize дозволить нам розробляти ефективні, масштабовані та безпечні веб-додатки. Цей стек технологій надає потужні інструменти для розробки та керування додатками, а також забезпечує високу продуктивність та захист даних.

3 ДЕМОНСТРАЦІЯ РОЗРОБЛЕНОГО ВЕБ-ДОДАТКА

3.1 Опис програмного продукту

Перед переглядом роботи, розкриємо можливості авторизованого та неавторизованого користувача.

Почнемо з розгляду можливостей неавторизованого користувача. Неавторизований користувач має обмежений функціонал і перед використанням повноцінного функціоналу додатку повинен спочатку пройти процедуру реєстрації. Після реєстрації необхідно здійснити аутентифікацію та авторизацію для отримання повного доступу до системи. Після успішного входу в систему, користувач отримує доступ до основного функціоналу, який дозволяє йому досягти своїх цілей у веб-додатку.

Неавторизований користувач має такі можливості:

- Перегляд головної сторінки, де представлена загальна інформація про додаток.
- Зареєструватися в системі, заповнивши необхідні дані для створення облікового запису.
- Виконати процедуру аутентифікації, вводить свої облікові дані для перевірки ідентифікації.
- Використовувати розділ з Частими запитаннями (FAQ) для отримання відповідей на поширені питання.
- Ознайомитися з умовами отримання автомобіля з доступних варіантів.
- Вивчення даних про доступні автомобілі.

Авторизованому користувачу надається основний функціонал для користування веб-додатком:

- Ознайомлення з правилами для створення свого кабінету, де з'явиться можливість надання свого автомобіля.
- Вибір способу оплати.

Адміністратор має функціонал, схожий на звичайного користувача, але з більшими привілеями:

- Можливість додавати нових адміністраторів.
- Можливість видаляти існуючі облікові записи користувачів, якщо вони порушують конфіденційні права.

3.2 Опис обраної архітектури веб-додатку

MVC - У процесі проектування даного програмного забезпечення було використано архітектурний шаблон MVC (Model-View-Controller). Цей шаблон дозволяє ефективно розділити відповідальності між компонентами додатку, спрощуючи розробку, підтримку та розширення коду. Завдяки MVC забезпечується більша гнучкість у внесенні змін до додатку, а також сприяє повторному використанню коду. Це сприяє покращенню продуктивності розробки і забезпечує більш зрозумілу та організовану структуру програмного забезпечення.

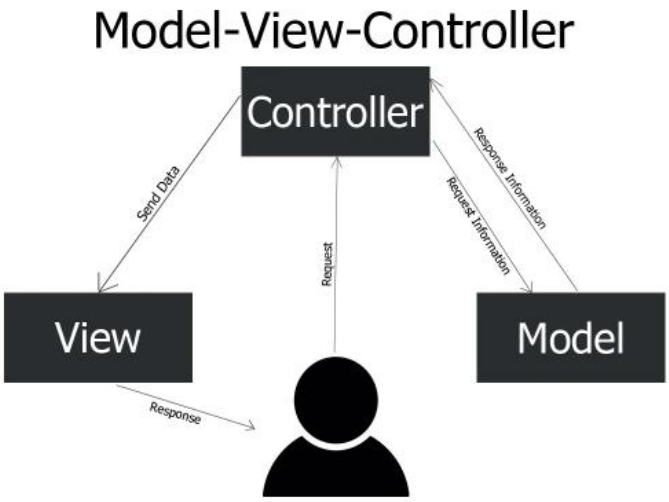

Рисунок 3.1– Шаблон (MVC)

Розглянемо, з яких компонентів складається цей шаблон.

1. (Model): Модель відповідає за управління даними та бізнес-логікою додатку. Вона представляє собою об'єкти, які зберігають інформацію та виконують

операції над даними. Модель зазвичай включає в себе структури даних, методи для доступу до даних, валідацію та логіку обробки.

2. (View): Представлення відповідає за відображення даних користувачу і інтерфейсу користувача. Воно отримує дані з моделі і відображає їх у зрозумілому для користувача вигляді. Представлення може бути HTML-сторінкою, шаблоном, графічним елементом або іншим способом відображення інформації.
3. (Controller): Контролер відповідає за обробку вхідних подій та взаємодію з моделлю та представленням. Він отримує запити від користувача, обробляє їх, оновлює модель та вибирає відповідне представлення для відображення результатів. Контролер також відповідає за маршрутизацію запитів та визначення правил взаємодії між моделлю і представленням.

Варто відзначити, що цей патерн проектування розподіляє додаток на окремі компоненти, такі як модель, представлення та контролер, але водночас передбачає тісну взаємодію між ними.

3.3 Представлення розробленого веб-додатку

В попередніх розділах ми детально розглянули всі аспекти стеку PERN, оглянули його переваги. У цьому розділі ми представимо реалізовану архітектуру серверної частини, наддамо інструкції щодо роботи з нею, продемонструємо користувацький інтерфейс

3.3.1 Створення веб-додатку

Створення веб-додатку розпочинається з серверної сторони. Розглянемо готову файлову архітектуру серверного веб-додатку (рис. 3.2).

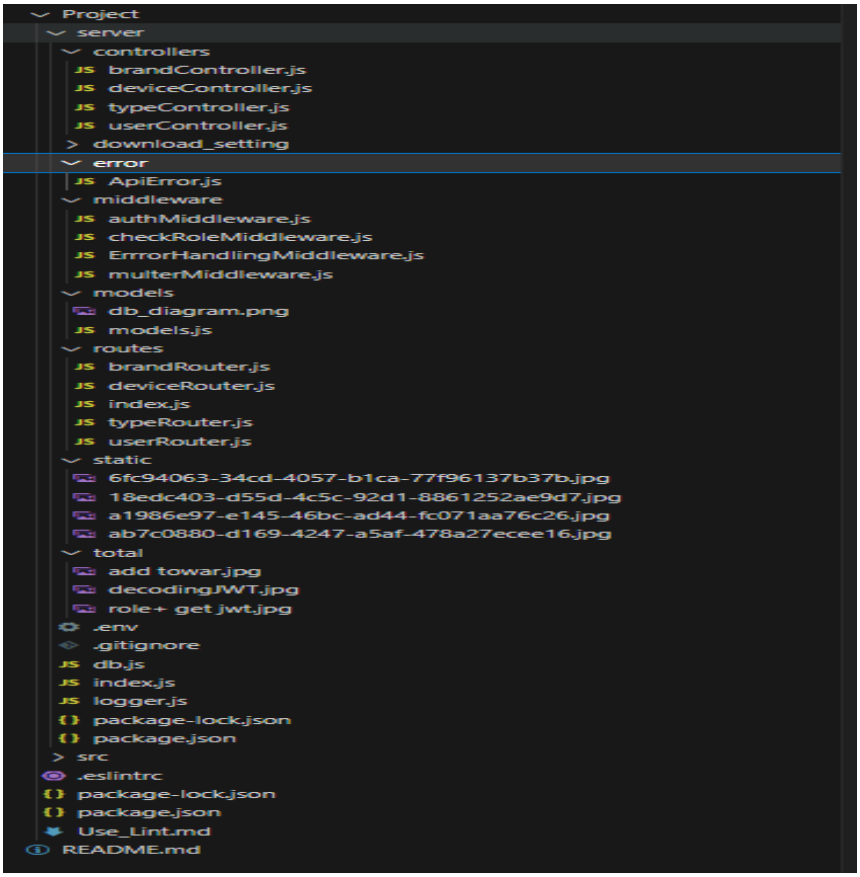

Рисунок 3.2 – Файлова структура додатку “backend”

Згідно з рисунком, архітектурою фреймворку, серверна частина сайту зберігається у окремих файлах, кожен з яких виконує свою задачу. Тепер розглянемо кожну частину проекту і розпишемо функціонал, за який вона відповідає.

- Налаштування всіх необхідних пакетів для Express.js, бази даних та ORM: Здійснюється встановлення та налаштування всіх необхідних пакетів, що дозволяють використовувати Express.js, працювати з обраною базою даних та підключати ORM Sequelize.

- Налаштування Express.js та глобальної маршрутизації: Створюється екземпляр додатка Express.js і здійснюються налаштування, такі як визначення порту, на якому буде працювати сервер, та інші загальні параметри. Також відбувається налаштування глобальної маршрутизації, визначення основних маршрутів, які будуть доступні на сервері.

- Створення бази даних та її налаштування: Створюється база даних у обраній СУБД PostgreSQL та налаштовуються таблиці та відношення між ними. Це може включати визначення схеми бази даних, створення таблиць, встановлення зв'язків та обмежень.

- Підключення ORM Sequelize та налаштування сутностей для бази даних: Виконується підключення ORM Sequelize до бази даних та налаштування моделей (сутностей), які представляють таблиці в базі даних. Визначаються поля моделей, їх типи, зв'язки з іншими моделями та інші налаштування, необхідні для роботи з даними в базі даних через ORM.

- Налаштування 5 контролерів для кожного маршруту: Створюються контролери, які обробляють запити до різних маршрутів додатка. Кожен контролер відповідає за обробку запитів, взаємодію з моделями через ORM Sequelize та формування відповідей на запити.

- Створення 4 Middleware: Реалізуються 4 Middleware: AuthMiddleware, CheckRoleMiddleware, ErrorHandlingMiddleware та multerMiddleware, щоб виконувати відповідні завдання.

- Створення повного списку ApiError та його впровадження в усій проект: Створюється повний список класів ApiError, які представляють помилки, які можуть виникати під час роботи додатка. Класи ApiError містять інформацію про помилку, статусний код, повідомлення та інші додаткові дані. Ці класи ApiError впроваджуються в усьому проекті, щоб обробляти та повертати помилки в стандартизованому форматі.

3.3.2 Аспекти коду

Ми наддамо більше уваги детальному розгляду важливих аспектів коду та розглянемо їх більш докладно.

Почнемо розгляд необхідних пакетів, без яких проект не може працювати (рис. 3.3).

Рисунок 3.3 — Пакети для налаштування всього коду

Давайте розглянемо опис кожного пакету та його функціональність:

- @aws-sdk/client-s3: Цей пакет надає клієнтську бібліотеку для взаємодії з сервісом Amazon S3 (Simple Storage Service) від Amazon Web Services (AWS). Він дозволяє завантажувати, завантажувати та керувати об'єктами у сховищі S3.

- bcrypt: Пакет bcrypt надає функції для хешування паролів за допомогою алгоритму bcrypt. Він забезпечує безпечне зберігання паролів у базі даних, дозволяючи зберігати лише хеші паролів замість початкових значень.

- crypto-js: Цей пакет надає набір функцій для виконання різних криптографічних операцій, таких як хешування, шифрування та дешифрування даних. Він надає зручні методи для роботи з різними алгоритмами шифрування та хешування.

- dotenv: Пакет dotenv дозволяє завантажувати змінні середовища з файлу .env до процесу додатка. Це дозволяє зберігати конфіденційні дані, такі як ключі API або

налаштування бази даних, у окремому файлі і завантажувати їх при запуску додатка.

- express: Цей пакет надає фреймворк Express.js, який є популярним вибором для розробки веб-додатків на Node.js. Express.js спрощує обробку HTTP-запитів, налаштування маршрутизації та роботу з проміжним програмним забезпеченням.
- express-fileupload: Пакет express-fileupload надає зручні функції для обробки завантаження файлів на сервер за допомогою Express.js. Він спрощує отримання файлів з запиту та їх збереження на сервері.
- jsonwebtoken: Цей пакет надає функції для створення та перевірки JSON Web Tokens (JWT). JWT - це стандарт для створення токенів аутентифікації, які можуть передаватися між клієнтом та сервером для підтвердження ідентичності користувача.
- multer: Пакет multer надає middleware для обробки даних форми, відправлених на сервер, включаючи завантаження файлів. Він дозволяє легко отримувати дані форми, включаючи файли, в додатках Express.js.
- pg: Цей пакет надає драйвер PostgreSQL для Node.js. Він дозволяє встановлювати з'єднання з базою даних PostgreSQL, виконувати запити та взаємодіяти з даними у базі даних.
- prettier: Пакет prettier надає інструмент для автоматичного форматування коду. Він може бути інтегрований у робочий процес розробника, щоб забезпечити єдність стилю кодування у проекті.
- sequelize: Цей пакет надає ORM (Object-Relational Mapping) для роботи з різними реляційними базами даних, включаючи PostgreSQL, MySQL та SQLite. Він спрощує взаємодію з базою даних, надаючи об'єктно-орієнтований підхід до роботи з даними.
- uuid: Пакет uuid надає функції для генерації унікальних ідентифікаторів (UUID). UUID широко використовується в додатках для унікальної ідентифікації об'єктів.

- winston: Цей пакет надає потужний і гнучкий логер для Node.js додатків. Він дозволяє записувати логи в різні місця, такі як консоль, файли або база даних, та налаштовувати форматування та рівні логування.

Всі ці пакети надали можливість розробляти і працювати з серверною частиною мого додатка, забезпечуючи функціональність, безпеку, зв'язок з базою даних та зручність роботи з Express.js та іншими сервісами.

3.3.3 Налаштування БД

Спочатку імпортується клас Sequelize з пакету sequelize. Деструктуризація дозволяє нам використовувати лише потрібний клас з пакету.

Потім ми експортуємо новий екземпляр Sequelize, який представляє підключення до бази даних PostgreSQL:

Потім налаштовуем змінні env:

- process.env.DB_NAME - це змінна середовища, яка містить ім'я бази даних, до якої ми хочемо підключитися.
- process.env.DB_USER - це змінна середовища, яка містить ім'я користувача бази даних.
- process.env.DB_PASSWORD - це змінна середовища, яка містить пароль для користувача бази даних.
- dialect: "postgres": Це параметр, який вказує Sequelize використовувати PostgreSQL як діалект бази даних.
- process.env.DB_HOST - це змінна середовища, яка містить хост бази даних, до якої ми хочемо підключитися.
- process.env.DB_PORT - це змінна середовища, яка містить порт бази даних.

З використанням цих параметрів, Sequelize буде створювати з'єднання з базою даних PostgreSQL і надавати доступ до різних функцій і методів для роботи з даними в базі даних.

3.3.4 Налаштування глобального роутингу

Після налаштування БД Postgres, та створення всіх сутностей, переходимо до налаштування глобального роутингу (рис. 3.3).

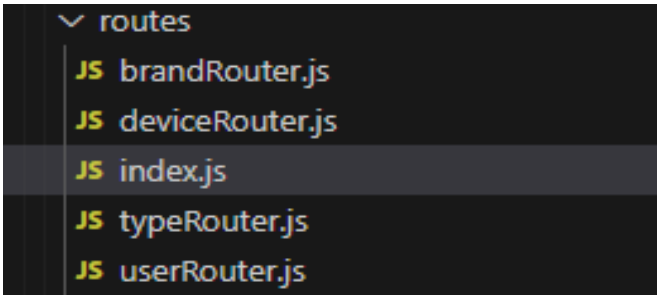

Рисунок 3.3 — Створення глобального роутінгу

```
Project > server > routes > JS index.js > ...
1   const Router = require("express");
2   // получаем роут из експрес
3   const router = new Router();
4   const deviceRouter = require("./deviceRouter")
5   const userRouter = require("./userRouter");
6   const brandRouter = require("./brandRouter");
7   const typeRouter = require("./typeRouter");
8
9   // Объединил 4 роутера в один
10  router.use("/user", userRouter);
11  router.use("/type", typeRouter);
12  router.use("/brand", brandRouter);
13  router.use("/device", deviceRouter);
14
15  module.exports = router;
16
```

Рисунок 3.4 — Налаштування маршрутизації

Цей код відноситься до налаштування маршрутизації в Express.js. Він імпортує модулі express та створює новий роутер (Router).

У першому блоці коду (рядки 3-8) ми імпортуємо різні роутери з відповідних модулів (userRouter, typeRouter, brandRouter, deviceRouter). Кожен роутер представляє набір маршрутів та обробників для певного функціоналу або ресурсу в додатку. В кінці коду ми експортуємо створений роутер, щоб його можна було використовувати в інших модулях або файловій системі додатку. Це дозволяє

об'єднати та організувати маршрути в одному місці і підключати їх до додатку в основному файлі сервера.

3.3.5 Контролери

Перш ніж перейти до розгляду кожного контролера у файлі, потрібно зрозуміти, для чого вони потрібні (рис. 3.5).

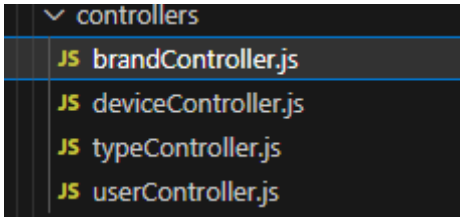

Рисунок 3.5 — Контролери

У Express.js контролери використовуються для обробки запитів та визначення логіки додатка. Вони є посередниками між маршрутами та моделями даних, виконуючи операції, пов'язані з обробкою запитів та формуванням відповідей. Тепер перейдемо до файлів.

У файлах typeController та brandController практично однакова структура. Вони надають можливості для валідації даних на клієнтській стороні під час створення та отримання інформації з бази даних. Ці контролери можуть бути використані для обробки запитів від клієнта за допомогою Axios.

Контролер DeviceController також включає асинхронну функцію create(req, res, next), яка використовує унікальні ключі brandId та typeId для створення запису infocar у базі даних. Його основна функція полягає об'єднанні всієї необхідної інформації, для зберігання в базі даних, і надає її через GET-запит.

Контролер userController відповідає за створення користувача з обов'язковими полями email та password. Він також містить допоміжні методи, такі як аутентифікація, авторизація та зміна пароля. Для забезпечення безпеки в

контролерах використовується модуль bcrypt для хешування паролів та генерації JWT-токенів

Розглянемо JWT - це стандарт для створення токенів, які використовуються для аутентифікації та авторизації в веб-додатках. JWT є компактним, та самодостатнім способом передачі інформації між сторонами у форматі JSON-об'єкта.

Процес роботи з JWT включає наступні кроки:

- Аутентифікація: Користувач надає свої облікові дані (наприклад, ім'я користувача та пароль) для аутентифікації на сервері.
- Видача токена: Після успішної аутентифікації сервер генерує JWT-токен і повертає його клієнту.
- Збереження токена: Клієнт зберігає отриманий JWT-токен у локальному сховищі (наприклад, localStorage) або у «cookie-файл». Токен буде використовуватись для ідентифікації користувача при подальших запитах.
- Передача токена: При кожному запиті клієнт включає JWT-токен у заголовок запиту або в іншу форму авторизації (наприклад, у заголовок "Authorization").
- Перевірка автентичності: Сервер отримує запит з JWT-токеном і перевіряє його автентичність та цілісність. Це включає перевірку підпису токена та його терміну дії.
- Витягнення даних: Якщо JWT-токен пройшов перевірку автентичності, сервер розшифровує його вміст та витягує інформацію про користувача або інші дані, необхідні для авторизації та обробки запиту.
- Обробка запиту: Сервер використовує витягнуті дані з JWT-токена для виконання запитаних операцій або надання доступу до певних ресурсів.

Тепер розглянемо процес створення користувача з визначеною роллю, та розкадрування JWT-токена для перевірки правильності його роботи. Для цього використав Postman для створення та тестування всіх HTPP-запитів. Рис(3.8, 3.9)

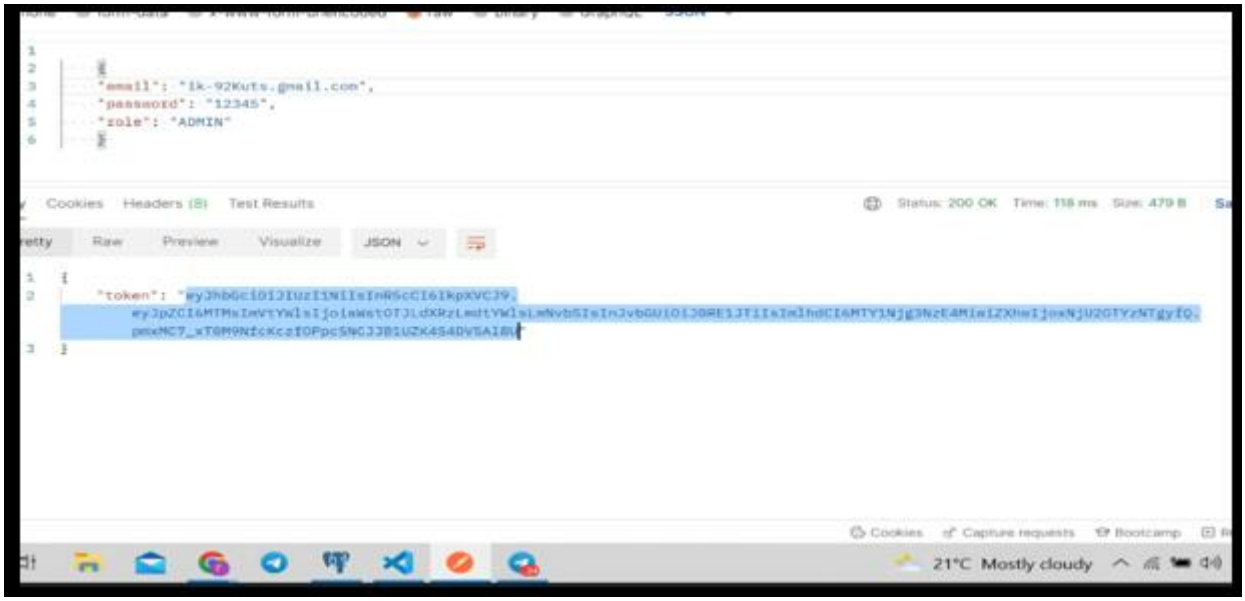

Рисунок 3.8 – Приклад створення користувача

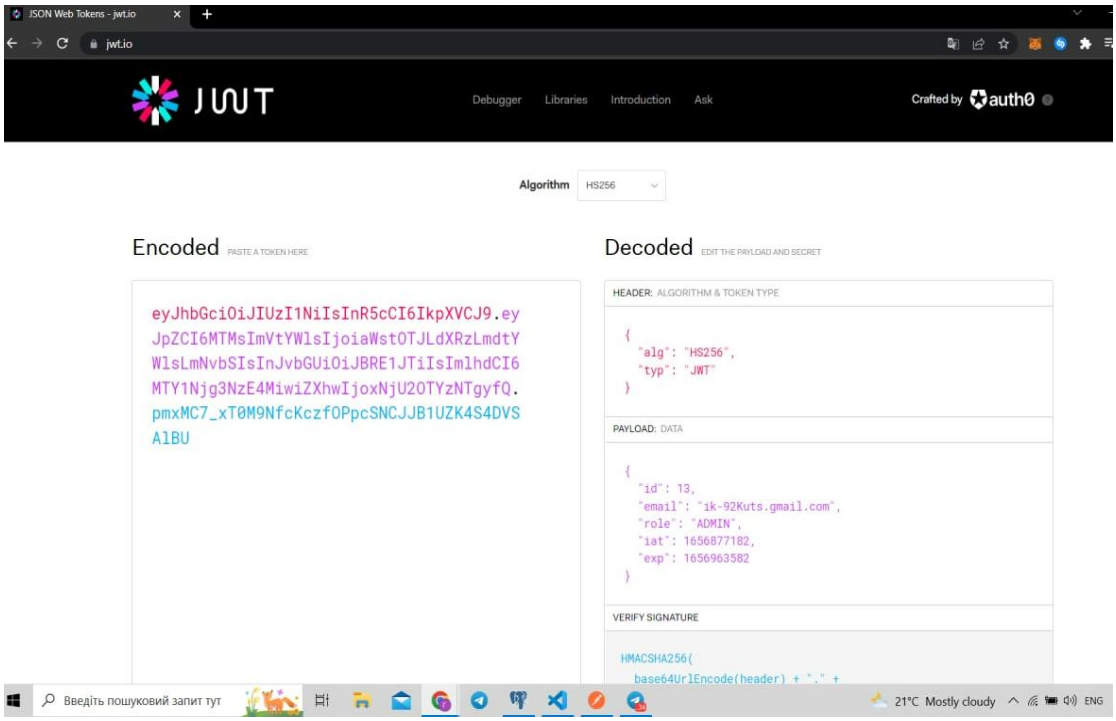

Рисунок 3.9 — Приклад створення користувача

Ми перевірили, що процес створення користувача з електронним адресом email “ik-92Kuts.gmail.com” та роллю «адмін» був успішним. Після відправки відповідного запиту, сервер обробив дані коректне і створив нового користувача з заданими параметрами. Це підтверджує, що функціонал створення користувачів працює належним чином.

3.3.6 Middleware

Кожна наша функція middleware отримує доступ до об'єктів request (запит), response (відповідь) і next (функція, яка викликає наступний middleware або обробник маршруту). У нашому випадку, Middleware виконує різні операції, такі як обробка та аналіз даних запиту, зміна об'єкта запиту або відповіді, виконання аутентифікації або авторизації, обробка помилок.

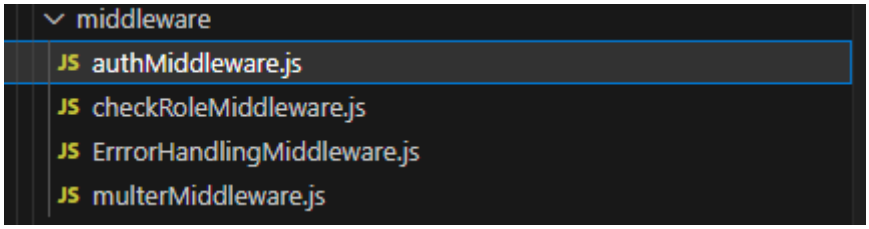

Рисунок 3.10 — Каталог middleware

- AuthMiddleware: Цей middleware використовується для аутентифікації користувачів. Він може перевіряти наявність і валідність токена аутентифікації, перевіряти дані користувача та дозволяти або відхиляти доступ до захищених маршрутів.
- CheckRoleMiddleware: Цей middleware перевіряє роль користувача і дозволяє доступ лише певним ролям. Наприклад, він може перевіряти, чи має користувач адміністративні привілеї або доступ до певних ресурсів.
- ErrorHandlingMiddleware: Цей middleware використовується для обробки помилок у додатку. Він може перехоплювати та обробляти винятки, надсилати відповідні повідомлення про помилку клієнту та виконувати інші дії для обробки ситуацій з помилками. Для цього був створений додатковий список ApiError.js помилок, які він повинен обробляти.
- MulterMiddleware: Цей middleware використовується для обробки форм даних у форматі мультіпарт, зокрема для завантаження файлів. Він дозволяє завантажувати файли на сервер та обробляти їх у запиті. Цей middleware забезпечує

коректну обробку завантажених файлів та їх збереження на сервері для подальшого використання або обробки.

3.4 Демонстрація роботи

У цьому розділі буде надано демонстрацію користувацького інтерфейсу та наведена інструкція щодо використання запропонованого веб-додатку.

3.4.1 Початкова сторінка

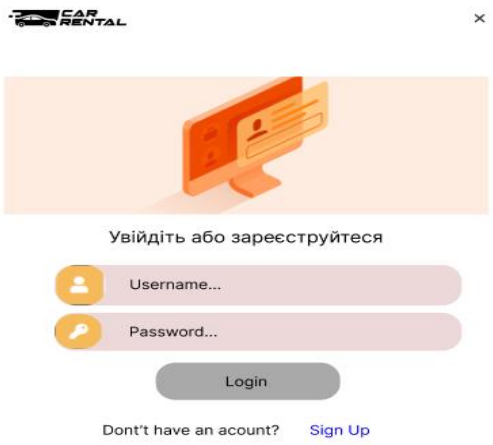

Рисунок 3.11— Початкова сторінка веб-сайт

При запуску нашого веб-сайту ви зустрічаєтесь зі стартовою сторінкою, де ви матиме можливість увійти або зареєструватися. Якщо у вас відсутній обліковий запис, ви також маєте можливість продовжити використовувати сайт як звичайний користувач. Якщо користувач обирає реєстрацію, він буде перенаправлений на сторінку, де йому потрібно заповнити всі обов'язкові поля, такі як "Ім'я користувача" (Username), "Електронна пошта" (Email), "Пароль" (Password) і "Повторіть пароль" (Repeat Password). На рис. 3.12 зображена сторінка реєстрації.

Після заповнення всіх необхідних полів і натискання кнопки "Зареєструватися", ваш обліковий запис буде створений і ви будете перенаправлені на головну сторінку, де ви зможете користуватися всіма функціями сайту.

Якщо користувач вже має обліковий запис, то ви можете ввести свої дані (електронну пошту і пароль) на стартовой сторінці, і після успішної аутентифікації вас також перенаправляють на головну сторінку.

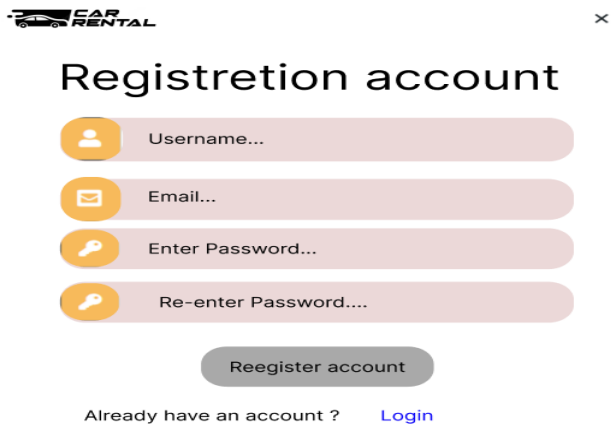

Рисунок 3.12 — Сторінка реєстрації

Якщо користувач не заповнив всі обов'язкові поля під час реєстрації, його реєстрація не буде здійснена. У такому випадку поряд з полями, де відсутні або не правильно введені данні, з'явиться повідомлення про помилку. Це повідомлення конкретизує, які саме дані необхідно виправити, приклад рис. 3.13.

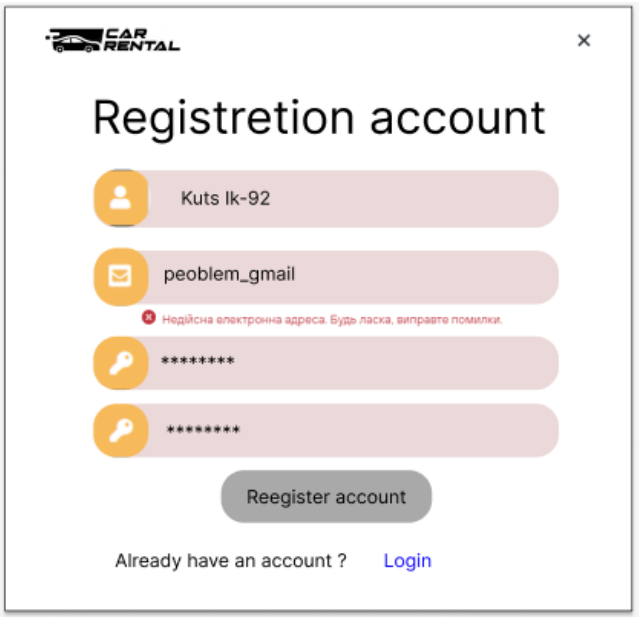

Рисунок 3.13 — Приклад реєстрації користувача

Якщо поле "Ім'я користувача" (Username) порожнє, з'явиться повідомлення "Будь ласка, введіть ім'я користувача".

Якщо поле "Електронна пошта" (Email) має неправильний формат, з'явиться повідомлення "Будь ласка, введіть коректну електронну пошту".

Якщо поле "Пароль" (Password) порожнє, з'явиться повідомлення "Будь ласка, введіть пароль".

Якщо поле "Повторіть пароль" (Repeat Password) не збігається з полем "Пароль" (Password), з'явиться повідомлення "Паролі не співпадають".

Ці повідомлення допомагають користувачеві зрозуміти, які поля є обов’язковими і що саме потрібно заповнити, щоб успішно завершити реєстрацію.

3.4.2 Можливості користувача

Розглянемо доступний функціонал додатку для не авторизованого користувача.

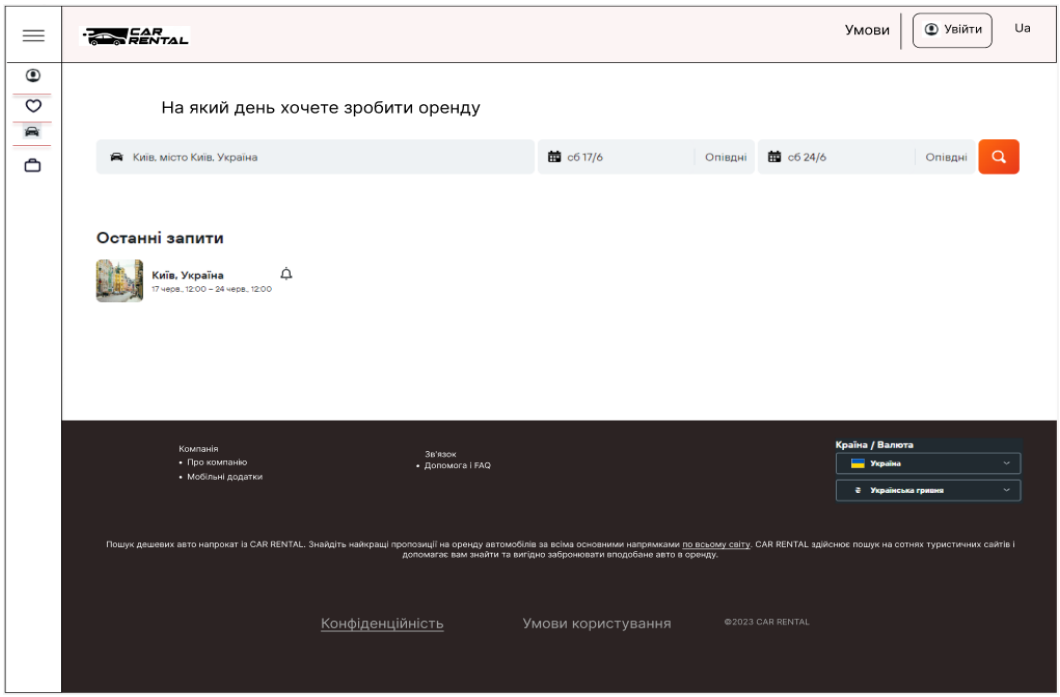

Рисунок 3.14 — Веб-сторінка для не авторизованого користувача

Функціонал для незареєстрованого користувача. При відкритті веб-додатку, для користувача відкривається стартова сторінка. На цій сторінці можливо ознайомитись з загальною інформацію про додаток, його функціоналом та можливостями.

У верхньому меню доступні такі вкладки (рис. 3.15):

- Увійти
- Перегляд авто
- Умови прокату
- Довідка

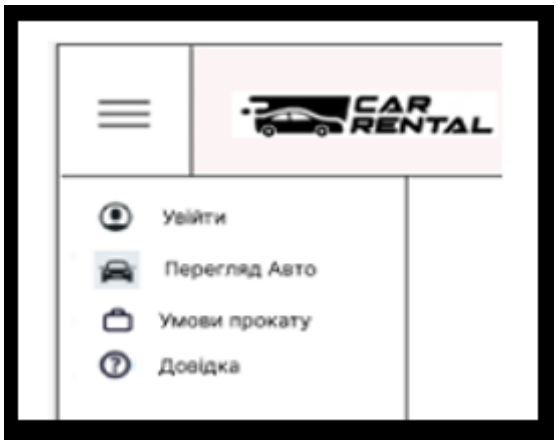

Рисунок 3.15 — Меню для не авторизованого користувача

На головній сторінці веб-додатку знаходиться кнопка "Увійти", яка дозволяє перейти до сторінки входу в систему (авторизації).

Перегляд Авто. Користувач може переглянути загальну інформацію, доступний список автомобілів та їх характеристики (рис. 3.16).

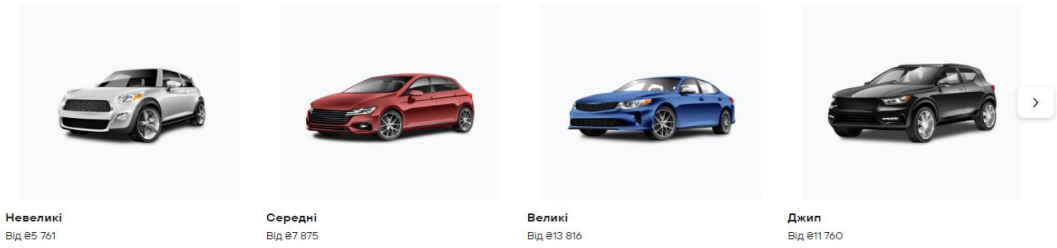

Рисунок 3.16 – Доступний список автомобілів

Умови прокату. На цій сторінці розміщена інформація про умови прокату автомобілів: вартість, терміни, умови повернення, способи оплати та інше.

Довідка. Ця сторінка містить детальні пояснення щодо використання веб-додатку, інструкції з реєстрації та інші корисні відомості (рис. 3.17).

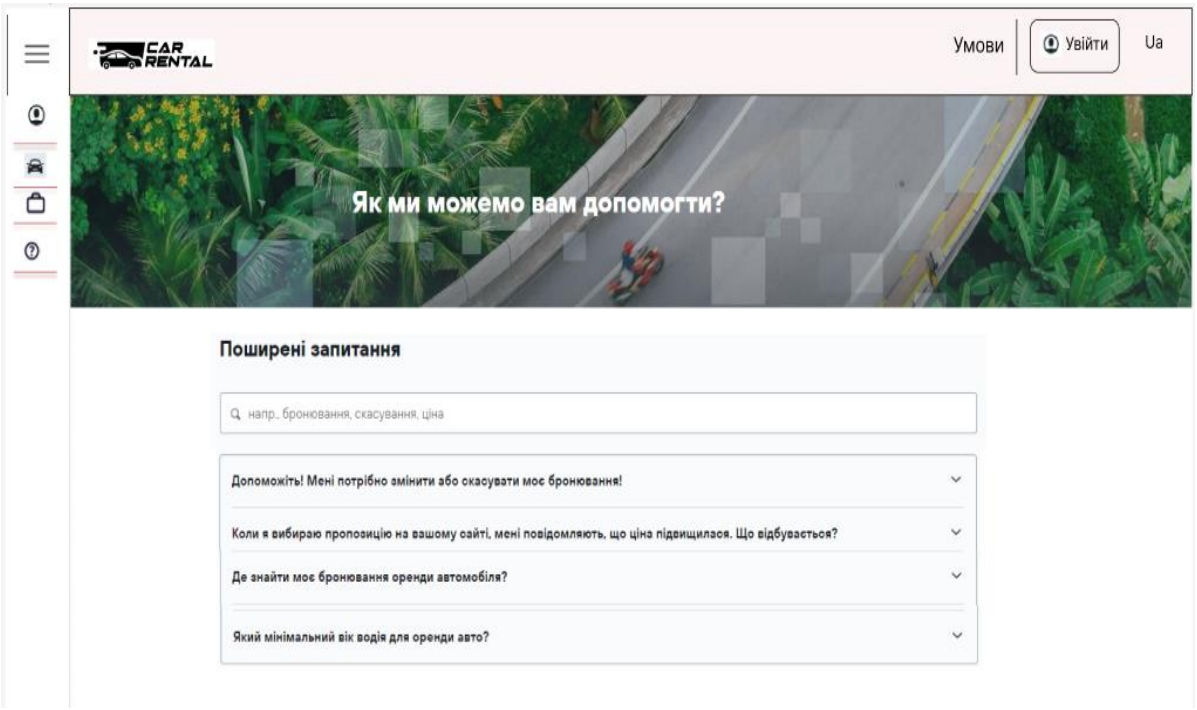

Рисунок 3.17 — FAQ

Незареєстрований користувач не може отримати доступ до деяких функцій, які доступні тільки зареєстрованим користувачам або адміністраторам.

Ми розглянули всі можливості які може отримати звичайний користувач (не авторизований). Зараз розглянемо функціональні можливості для зареєстрованого користувача (user). Авторизований користувач має можливість переглядати та редагувати свої особисті дані в профілі.

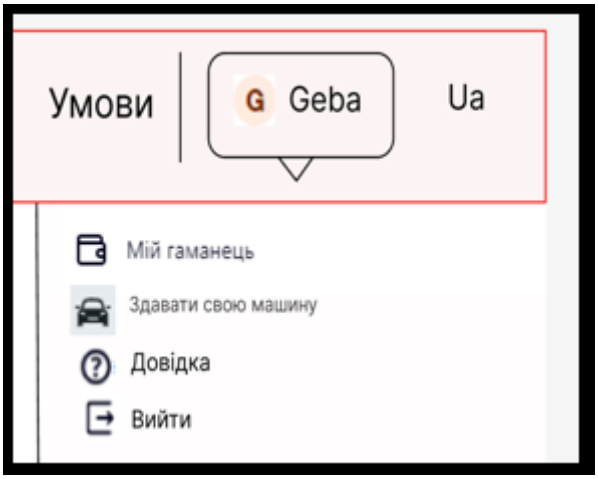

Рисунок 3.18 — Меню авторизованого користувача

Гаманець – ця функція зроблена для зручності постійних споживачів, які часто користуються услугами веб-сайту, та бажають прискорити процес оплати послуг. Цей функціонал дозволяє прискорити процес оплати. Проте, перед отриманням доступу до гаманця, для забезпечення безпеки була створена додаткова проміжна перевірка. Це додаткова валідація, яка перевіряє правильність і достовірність введених даних користувача. Це зроблено з метою запобігання несанкціонованому доступу до гаманця та забезпечення (рис. 3.19).

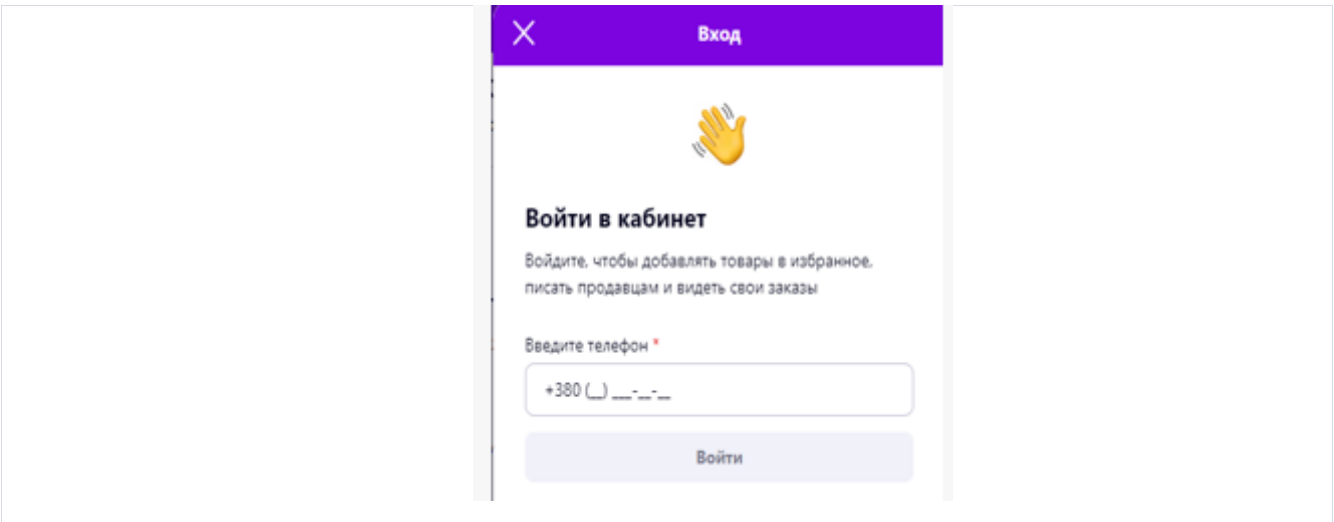

Рисунок 3.19 — Вхід і кабінет користувача

Після пройденої перевірки користувач отримує доступ до гаманця і може використовувати його для зручної та швидкої оплати послуг.

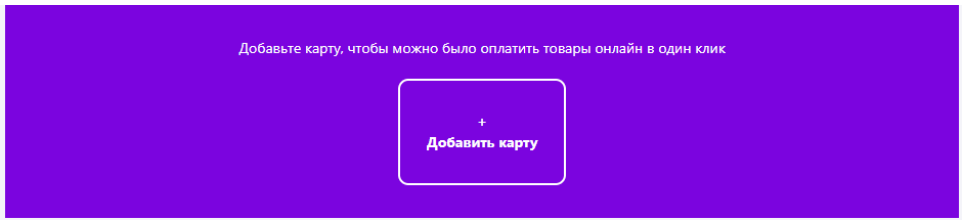

Рисунок 3.20 — Гаманець користувача

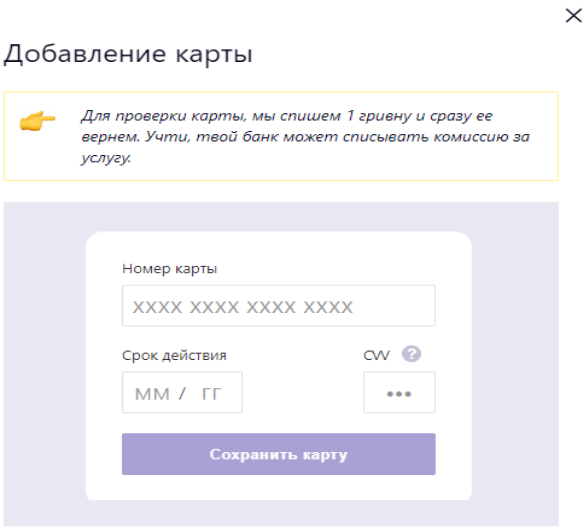

Рисунок 3.21 — Сторінка розрахунку користувача

Далі розглянемо додаткову функцію - систему оплати крипто-валютою. Для цього ми скористалися послугами CloudCrypto. Спочатку ми налаштували основні параметри для нашого веб-сайту, а потім вибрали, який вид крипто валюти ми будемо приймати до сплати.

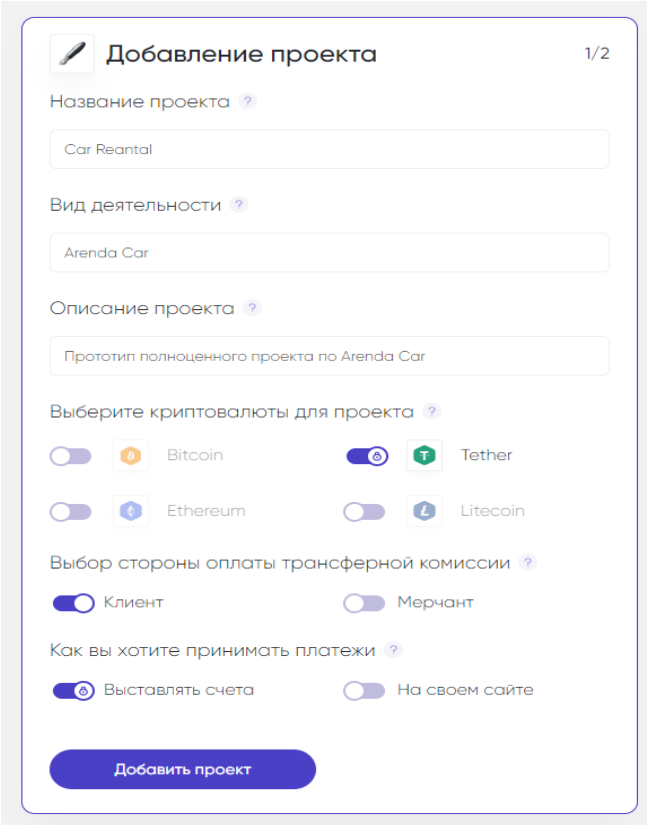

Рисунок 3.22 — Налаштування системи оплати крипто-валюти

В першому розділі ми розглянули, чому саме Tether є перевагою. Tether є стейблкоїном, який пов'язаний з фіатною валютою, у даному випадку з американським доларом, в співвідношенні 1:1. Це означає, що його ціна залишається стабільною і передбачуваною. Це чудовий варіант для оплати, оскільки він несе мінімальний ризик волатильності, який притаманний іншим крипто-валютам.

Рисунок 3.23 — Отримання ID для проведення транзакцій

В подальшому ми впроваджуємо наш проект через API систему та поводимо першу угоду через наш веб-додаток.

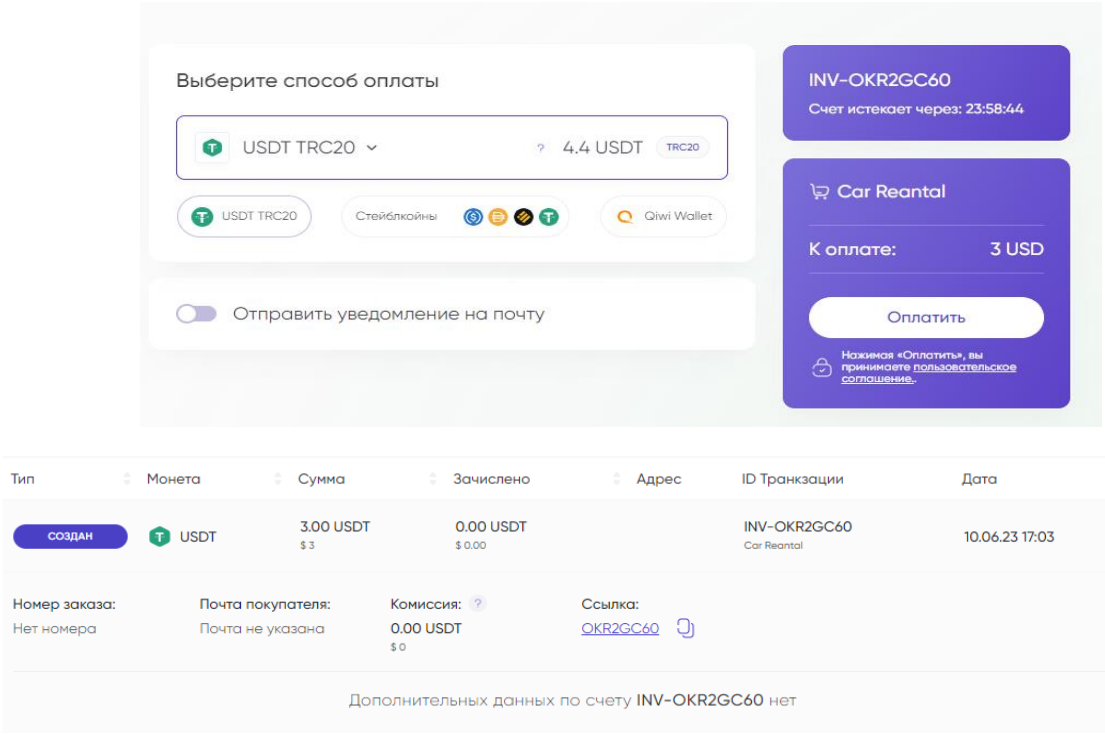

Рисунок 3.24 — Приклад угоди (розрахунок крипто-валюта)

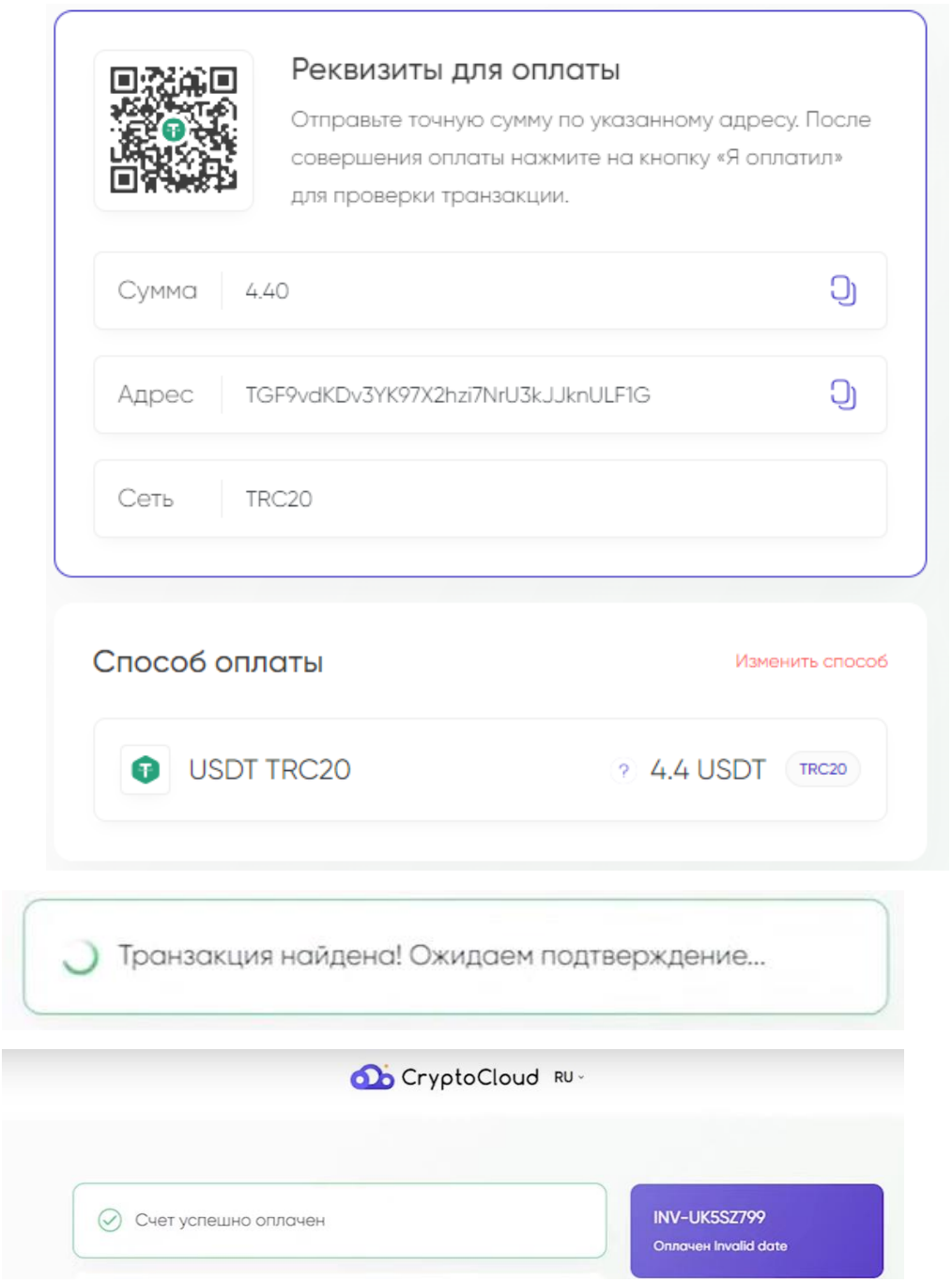

Рисунок 3.25 — Приклад проведення оплати

Пошук та оренда автомобіля.

Авторизований користувач має можливість використовувати пошукову функцію для пошуку точки прокату та доступних автомобілів. При пошуку він може застосувати фільтри відбору щоб обрати підходящий варіант.

Наступним кроком є оренда автомобіля. Для цього необхідно ознайомитися з правилами та вимогами відносно документальної підтримки. Після розгляду заявки адміністратором буде розміщено маркер на Google Maps (рис. 3.26).

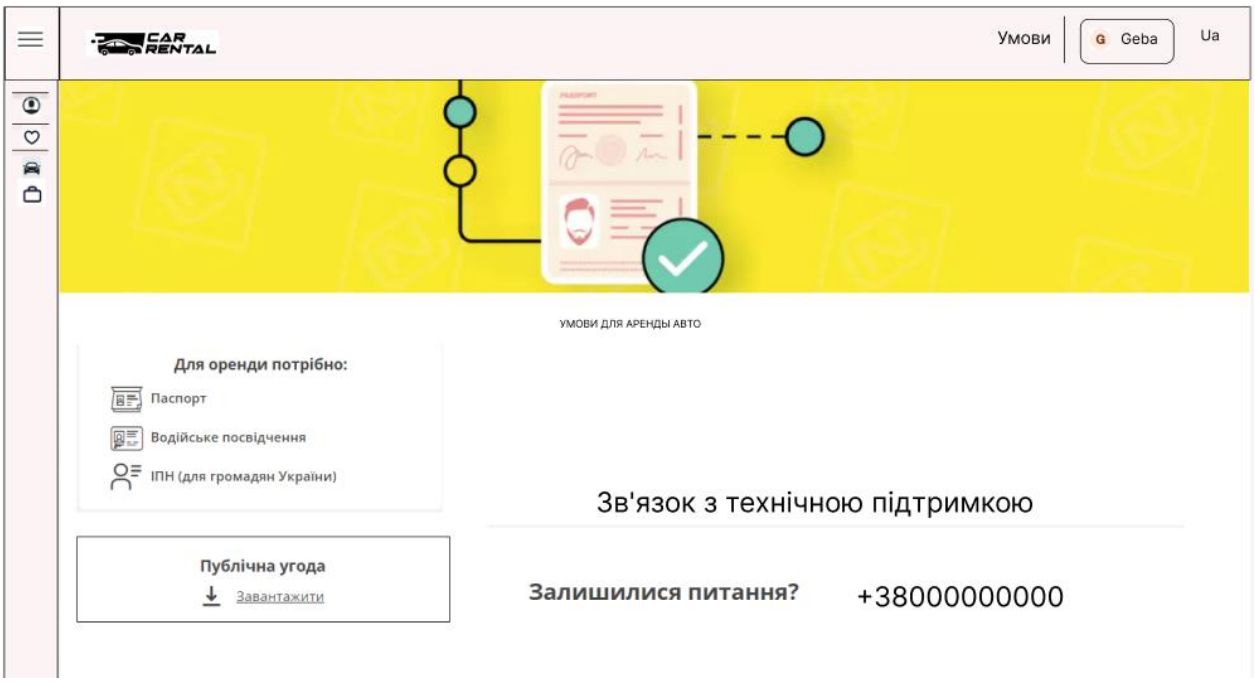

Рисунок 3.26 — Приклад бронювання автомобіля

Висновок до розділу

У третьому розділі була проведена презентація дипломного проекту, в рамках якого був розроблене веб-додаток. Розкриті технічні характеристики та функціональні можливості. Один із основних аспектів на який було звернута увага це зручні умови оплати, що значно поліпшує сервіс для клієнтів. Крім того для забезпечення високо рівня безпеки впроваджена додаткова проміжна перевірка. Реалізовано система пошуку та реєстрації точок прокату автомобілів.

ВИСНОВКИ

У даному дипломному проекті була розглянута тема «Web-site Система пошуку та реєстрації точок прокату автомобілів».

Для реалізації проекту було проведено постановку задачі, визначення цілей та опис предметної області. Аналіз існуючих веб-додатків дозволив виявити основні потреби та вимоги користувачів, а також визначити функціональні можливості системи. Для реалізації проекту були запропоновані архітектурні рішення, зокрема використання клієнт-серверної архітектури шаблону MVC. Ці рішення дозволять створити ефективний, масштабований та безпечний веб-застосунок, забезпечуючи зручний інтерфейс для користувачів та ефективну роботу з даними.

Також при розробці веб-системи для оренди автомобілів особлива увага була приділена безпеці даних. Були застосовані сучасні методи шифрування та захисту від несанкціонованого доступу, щоб забезпечити конфіденційність та цілісність інформації користувачів. Гнучкість системи є одним із головних переваг обраного стека PERN, який наддав надійну та масштабовану архітектуру. PostgreSQL гарантує надійне зберігання даних, а Express та Node.js забезпечує високу продуктивність системи. Інтерфейс веб-додатку розроблено з урахуванням передових практик користувацького досвіду та сучасного дизайну.

Реалізовано інноваційна система пошуку та реєстрації точок прокату автомобілів. Така система надає користувачеві зручну та ефективну платформу, що значно спростить процес пошуку точок доступних для прокату автомобілів.

Додаткові корисні функції, такі як сплата через крипто-валюту та можливість здачі автомобіля оренду, поліпшує користувацький досвід та розширює можливості системи. В результаті, цей веб-додаток пропонує зручний, безпечний та інноваційний спосіб пошуку та реєстрації точок прокату автомобілів, який відповідає вимогам користувачів.

Отже, розробка веб-додатку для оренди автомобілів є актуальною сферою приватних послуг, яка має попит. Реалізація цього проекту з урахуванням нових функцій і сучасних технологій буде користуватись попитом у користувачів.

ПЕРЕЛІК ВИКОРИСТОВАНИХ ДЖЕРЕЛ

1. Руководство по Node.js [Електронний ресурс]. – Режим доступу: https://metanit.com/web/nodejs/ - дата доступу 10.05.23
2. Посібник з AMP-оптимізатору для Node.js [Електронний ресурс]. – Режим доступу: https://amp.dev/ru/documentation/guides-and-tutorials/optimize-and-measure/amp-optimizer-guide/node-amp-optimizer - дата доступу 10.05.23
3. Express Інтеграція з базами даних [Електронний ресурс]. – Режим доступу: https://expressjs.com/ru/guide/database-integration.html#mysql -дата доступу 15.05.23
4. Express Маршрутизація [Електронний ресурс]. – Режим доступу: https://expressjs.com/ru/guide/routing.html - дата доступу 15.05.23
5. Довідник. Матеріал з Вікіпедії [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/JSON_Web_Token - дата доступу 10.05.23
6. Безпека JSON Web Tokens (JWT) [Електронний ресурс]. – Режим доступу: https://cyberpolygon.com/ru/materials/security-of-json-web-tokens-jwt/ - дата доступу 10.05.23
7. Відео уроки. JavaScript.Ninja [Електронний ресурс]. – Режим доступу: https://www.youtube.com/@JavaScriptNinja - дата доступу 15.04.23
8. Відео уроки. Security with PostgreSQL [Електронний ресурс]. – Режим доступу: https://www.youtube.com/@EDBPostgres - дата доступу 20.04.23
9. Відео уроки . What is ORM? Object Relational Mapping [Електронний ресурс]. – Режим доступу: https://www.youtube.com/@JavaGuides - дата доступу 17.04.23
10. React. JavaScriptt – бібліотека для створення інтерфейсів користувача [Електронний ресурс]. – Режим доступу: https://ru.legacy.reactjs.org/docs/getting-started.html - дата доступу 17.04.23
11. Get started with Bootstrap [Електронний ресурс]. – Режим доступу: https://getbootstrap.com/docs/5.3/getting-started/introduction/ - дата доступу 19.04.23

12. Умови та Правила надання послуг у системі LiqPay [Електронний ресурс]. – Режим доступу: https://www.liqpay.ua/ru/information/terms - дата доступу 17.05.23

13. Binance. Що таке USDT TRC20 [Електронний ресурс]. – Режим доступу:https://www.binance.com/ru/blog/chain/%D1%87%D1%82%D0%BE-%D1%82%D0%B0%D0%BA%D0%BE%D0%B5-usdt-trc20-7898189773550076013 дата доступу 17.05.23

14. Mobx. Бліотека для управление состоянием [Електронний ресурс]. – Режим доступу https://mobx.js.org/README.html

15. Mapbox. Бліотека для правцювання с картами [Електронний ресурс] Режим доступу https://docs.mapbox.com/