МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)
на тему:

«Система управління орендою автомобілів»

Виконав студент групи ІПЗп-21-1
ПОВОРОЗНЮК Ілля Васильович

Керівник роботи:
ЧИЖМОТРЯ Олена Геннадіївна

Рецензент:
ЛОКТІКОВА Тамара Миколаївна

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»
Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«01» серпня 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **Поворознюк Ілля Васильович**
Керівник роботи: **ЧИЖМОТРЯ Олена Геннадіївна**
Тема роботи: **«Система управління орендою автомобілів»**,
затверджена Наказом закладу вищої освіти від **«01» серпня 2025 р., №270/с**
Вихідні дані для роботи: **Об'єктом дослідження веб-застосунок для оренди автомобілів. Предметом дослідження є процеси та підходи проектування, розробки та зручний створення, оплата та перегляд інформації про орендований атомобіль.**
Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	**Консультант**	Завдання видав	Завдання прийняв
1	Олена ЧИЖМОТРЯ	02.08.2025	02.08.2025
2	Олена ЧИЖМОТРЯ	02.08.2025	02.08.2025
3	Олена ЧИЖМОТРЯ	02.08.2025	02.08.2025

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра включає веб-систему "Оренда автомобілів" та супровідну пояснювальну записку. Пояснювальна записка налічує 75 сторінок, містить 28 ілюстрації, 19 таблиць, 3 додатки та 30 використаних літературних джерел.

Мета роботи полягає у створенні зручної та захищеної веб-системи для оренди автомобілів із верифікацією користувачів і безпечним доступом до контактних даних. Для реалізації використано фреймворк Laravel, базу даних MySQL, Craftable, SaaS і jQuery для клієнтської частини, а також Vue, для адмінпанелі. Застосовано принципи об’єктно-орієнтованого програмування, реляційну модель бази даних та методи захисту (CSRF-токени, middleware). Розроблено функціонал реєстрації, авторизації, створення та сортування пропозицій, обробки запитів і адміністративного контролю. Інтерфейс оптимізований для різних ролей користувачів, забезпечуючи просту та інтуїтивну навігацію. Тестування підтвердило відповідність системи заявленим вимогам.

КЛЮЧОВІ СЛОВА: ВЕБ-ЗАСТОСУНОК, ОРЕНДА АВТОМОБІЛІВ, LARAVEL, MYSQL, СОРТУВАННЯ, ЗАСТОСУНОК

ABSTRACT

The bachelor's thesis comprises a web-based platform “Car Rental” and an accompanying explanatory note. The explanatory note includes 75 pages, 28 illustrations, 19 tables, 3 appendices, and 30 cited literary sources.

The objective of the work is to develop a user-friendly and secure web platform for car rental services, featuring user verification and protected access to contact details. The implementation leverages the Laravel framework, a MySQL database, Craftable SaaS, and jQuery for the client-side, alongside Vue for the admin panel. The project employs object-oriented programming principles, a relational database model, and security techniques (CSRF tokens, middleware). Developed functionalities include registration, authorization, creation and sorting of rental offers, request handling, and administrative oversight. The interface is tailored to various user roles, ensuring straightforward and intuitive navigation. Testing validated the system’s compliance with the specified requirements.

KEYWORDS: WEB APPLICATION, CAR RENTAL, LARAVEL, MYSQL, SORTING, APPLICATION

ЗМІСТ

РЕФЕРАТ 3

ABSTRACT 4

ВСТУП 7

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ 9

1.1. Постановка задачі 9

1.2. Аналіз аналогів програмного продукту 10

1.3. Вибір архітектури веб-орієнтованої системи 16

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення 18

Висновки до першого розділу 20

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ 22

2.1. Визначення варіантів використання та об’єктно-орієнтованої структури системи 22

2.2. Розробка бази даних системи 31

2.3. Проєктування та реалізація алгоритмів роботи системи 41

2.4. Реалізація функціоналу системи управління орендою автомобілів 43

Висновки до другого розділу 45

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ УПРАВЛІННЯ ОРЕНДОЮ АВТОМОБІЛІВ 47

3.1. Порядок встановлення та налаштування параметрів платформи 47

3.2. Структура інтерфейсу веб застосунку «Система управління орендою автомобілів» 48

3.3. Тестування роботи системи 55

Висновки до третього розділу 60

ВИСНОВКИ 62

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 63

ДОДАТКИ 66

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

СУБД – система управління базами даних

ORM – Object-Relational Mapping

CRUD – Create, Read, Update, Delete

MVC – Model-View-Controller

БД – база даних

CSRF – Cross-Site Request Forgery

ПЗ – програмне забезпечення

ВСТУП

Актуальність теми даної роботи зумовлена постіним розвитком та зростанням оренди автомобіля завдяки зручності, доступності та гнучкості, які вона пропонує. У сучасному світі автомобіль є важливим інструментом для подорожей, роботи чи повсякденних потреб, але не всі готові купувати власне авто. Веб-платформи для оренди автомобілів спрощують процес пошуку, бронювання та оплати транспорту, роблячи його швидким і зручним. Водночас створення таких платформ пов’язане з викликами, адже потрібно забезпечити зручний інтерфейс, безпечні транзакції та легкий доступ до інформації про автомобілі. Розвиток цифрових технологій відкриває нові можливості для автоматизації цих процесів, що робить тему актуальною та затребуваною.

Метою роботи є розробка веб-платформи для оренди автомобілів, яка забезпечує зручне бронювання, оплату та перегляд інформації про орендовані автомобілі, а також відповідає сучасним стандартам зручності, безпеки та ефективності.

Для досягнення мети необхідно **виконати такі завдання**:

- Проаналізувати вимоги до веб-платформ для оренди автомобілів та вивчити досвід існуючих рішень.
- Визначити оптимальну структуру та архітектуру платформи, враховуючи потреби користувачів.
- Оцінити сильні та слабкі сторони аналогічних сервісів на ринку.
- Підібрати та обґрунтувати вибір технологій для розробки платформи.
- Розробити основні компоненти системи, включаючи інтерфейс користувача, алгоритми бронювання та обробки платежів.
- Провести тестування платформи для забезпечення її надійності, швидкості та зручності використання.

Об’єктом дослідження є веб-платформа для оренди автомобілів, яка забезпечує ефективне управління процесами пошуку, бронювання та оплати транспортних засобів.

Предметом дослідження є процеси проектування, розробки та реалізації веб-платформи для оренди автомобілів, включаючи створення зручного інтерфейсу, інтеграцію платіжних систем і забезпечення безпеки даних. Особлива увага приділяється використанню сучасних веб-технологій для підвищення якості користувацького досвіду, швидкості роботи платформи та її адаптивності до різних пристроїв. У процесі розробки застосовуються методи об’єктно-орієнтованого програмування, сучасні інструменти веб-розробки, а також технології для роботи з базами даних і API, що забезпечують надійність, масштабування та безпеку системи.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ

1.1. Постановка задачі

Цей проєкт присвячено розробці сучасного та зручного сайту, який дозволить користувачам легко знаходити, бронювати та оплачувати автомобілі для оренди. Платформа має бути максимально простою і зрозумілою у використанні: від моменту входу на сайт до завершення оплати. Важливо, щоб сервіс підтримував різні мови та валюти, забезпечуючи доступність для людей з усього світу, незалежно від їхнього місця проживання чи рідної мови.

Оренда автомобілів набуває все більшої популярності завдяки своїй зручності, доступності та гнучкості. Це дозволяє пересуватися без необхідності купувати власне авто, чи то для подорожей, роботи, чи повсякденних потреб. Щоб платформа для оренди авто була дійсно корисною, вона має бути швидкою, безпечною і простою у навігації. Передбачається створення сервісу, де користувач зможе за кілька хвилин знайти потрібний автомобіль, вибрати зручні умови оренди, забронювати його та оплатити без зайвих труднощів. Безпека даних і транзакцій при цьому залишається ключовим пріоритетом, адже довіра користувачів є основою успіху.

Для реалізації проєкту спочатку буде проведено аналіз роботи схожих сервісів на ринку, щоб визначити їхні сильні та слабкі сторони. Це допоможе взяти найкращі ідеї та уникнути типових помилок. Далі буде створено зручну структуру сайту та продуманий дизайн, який виглядатиме сучасно і працюватиме однаково добре на смартфонах, планшетах чи комп’ютерах. Використовуватимуться найновіші технології, які забезпечать швидкість, надійність і можливість подальшого масштабування платформи. Буде розроблено зрозумілий інтерфейс, щоб навіть користувачі з мінімальним досвідом роботи в інтернеті могли легко розібратися. Передбачається додавання зручних функцій для пошуку автомобілів, бронювання та оплати, а також інтеграція підтримки різних мов і валют для забезпечення глобальної доступності. Особлива увага приділятиметься безпеці: дані користувачів

захищатимуться за допомогою шифрування та сучасних протоколів, щоб гарантувати конфіденційність і безпеку платежів.

Після завершення розробки платформа буде ретельно протестована, щоб переконатися у відсутності збоїв. Мета — забезпечити, щоб користувачі отримували лише позитивний досвід від роботи з сервісом, а процес оренди автомобілів був для них приємним і безтурботним.

1.2. Аналіз аналогів програмного продукту

Перш ніж розпочати розробку або вдосконалення програмного продукту, важливо провести аналіз існуючих рішень на ринку. Це дозволяє оцінити доступні можливості, визначити сильні та слабкі сторони конкурентів, а також знайти ідеї для покращення власного сервісу. Аналіз аналогів допомагає зрозуміти поточний стан галузі, виявити тенденції та сформувати конкурентоспроможну стратегію розвитку продукту.

Під час аналізу ринку сервісів оренди автомобілів було визначено три основні платформи: K-Rent, Renty та Ukr-Prokat. Нижче наведено детальний розгляд кожного з них, з акцентом на функціональні можливості, переваги та недоліки.

Сервіс K-Rent

Посилання на вебсайт: https://k-rent.com.ua/.

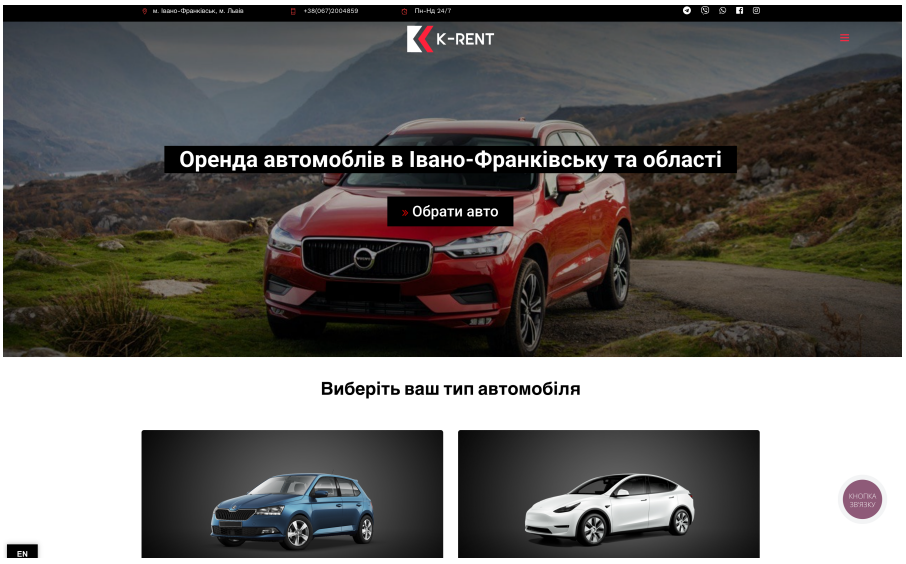


Рис. 1.1 – Вигляд головної сторінки вебсайту K-Rent

Сервіс K-Rent є українською платформою для прокату автомобілів, яка пропонує широкий асортимент транспортних засобів у різних містах країни, що робить його зручним для місцевих користувачів, які шукають гнучкі умови оренди. Платформа вирізняється інтуїтивно зрозумілою системою навігації, яка дозволяє швидко відфільтрувати автомобілі за типом, місцем отримання чи іншими критеріями, а також забезпечує можливість бронювання онлайн із додатковими опціями, такими як страхування, GPS-навігація чи дитяче крісло, що значно підвищує комфорт для клієнтів. Підтримка користувачів реалізується через чат і телефонну гарячу лінію, що сприяє оперативному вирішенню питань, а наявність функції довгострокової оренди та зручного мобільного додатку додає зручності для регулярних орендарів. Сортування автомобілів за ціною, маркою чи класом полегшує вибір, адаптуючи пропозиції до потреб кожного клієнта.

Переваги сервісу включають різноманітність автомобілів, що охоплюють економ-, бізнес- та преміум-класи, що дає змогу задовольнити широкий спектр потреб, від бюджетних поїздок до розкішних подорожей, а також наявність додаткових послуг, таких як страхування чи аксесуари, які роблять оренду більш безпечною та зручною. Можливість оплати як онлайн, так і при отриманні автомобіля забезпечує гнучкість, дозволяючи користувачам обирати зручний спосіб транзакції залежно від обставин. Водночас недоліки полягають у тому, що в деяких невеликих містах асортимент авто обмежений, що може ускладнити вибір для жителів регіонів, а функції, такі як доставка автомобіля, доступні лише в найбільших містах, що створює незручності для користувачів у віддалених районах. Загалом сервіс демонструє сильні сторони в адаптації до потреб клієнтів, але потребує розширення географічного охоплення та покращення доступності послуг у менш урбанізованих зонах.

Сервіс Renty

Посилання на вебсайт: https://renty.ae.

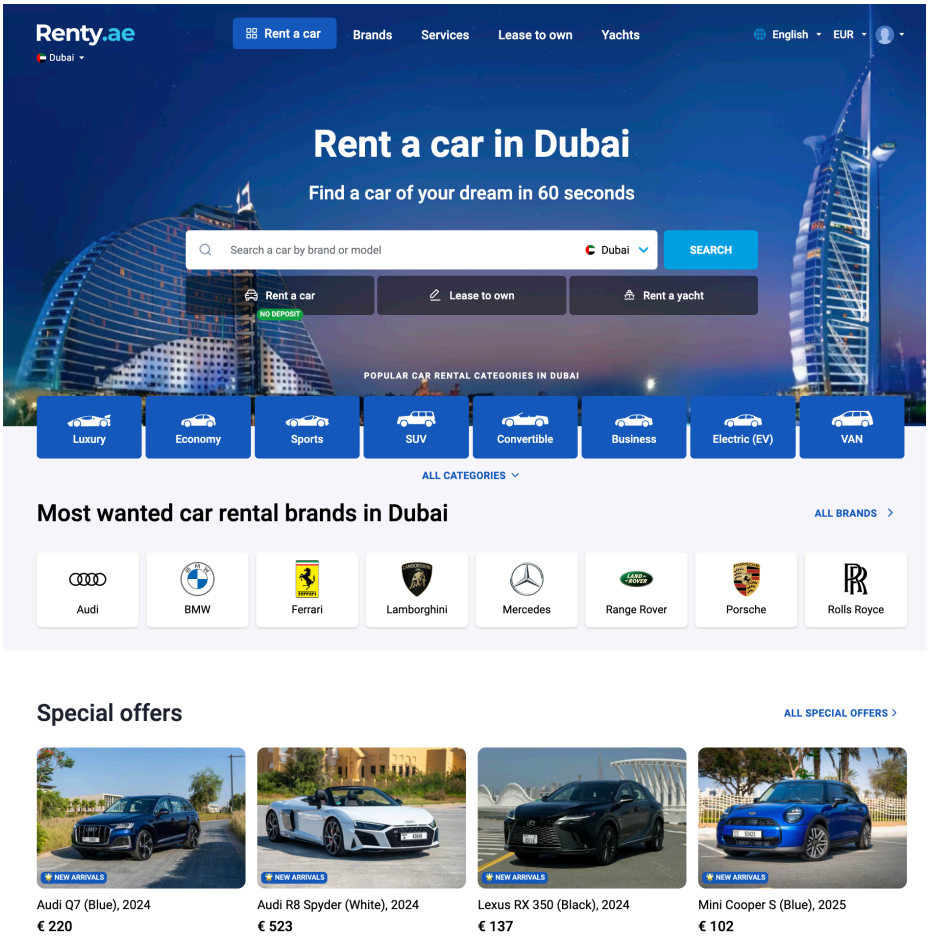


Рис. 1.2 – Вигляд головної сторінки вебсайту Renty

Renty є міжнародним сервісом оренди автомобілів, який активно функціонує в Дубаї та Об'єднаних Арабських Еміратах, пропонуючи клієнтам як стандартні моделі, так і розкішні авто, включаючи спорткари, що робить його привабливим для заможних туристів та місцевих жителів. Платформа вирізняється можливістю оренди ексклюзивних транспортних засобів, яких важко знайти в інших сервісах, а також пропонує зручну доставку автомобіля безпосередньо до клієнта, що значно спрощує процес використання. Користувачі мають доступ до різноманітних способів оплати, таких як готівка, банківські картки та навіть криптовалюта, що додає гнучкості, а онлайн-бронювання через офіційний сайт або мобільний додаток забезпечує швидкий доступ до послуг у будь-який час. Підтримка клієнтів доступна цілодобово, що гарантує оперативне вирішення питань, а розширена система фільтрації за маркою, класом автомобіля та вартістю дозволяє легко підібрати оптимальний варіант.

Переваги сервісу полягають у його унікальному асортименті, де представлені ексклюзивні моделі, які приваблюють любителів розкоші, а також у швидкому процесі оформлення оренди, що економить час, особливо для туристів і бізнесменів, які потребують оперативних рішень. Зручність для мандрівників забезпечується завдяки доставці та гнучким умовам, що робить Renty зручним інструментом для короткострокових і довгострокових поїздок. Однак недоліки включають високу вартість прокату, яка може бути недоступною для середнього класу, а також обмеженість географічного покриття, оскільки сервіс працює виключно в ОАЕ, що суттєво звужує його аудиторію поза межами цього регіону. Загалом платформа демонструє сильні сторони в сегменті преміум-оренди, але потребує розширення доступності та зниження цін для ширшої конкурентоспроможності.

Сервіс Ukr-Prokat

Посилання на вебсайт: https://ukr-prokat.com/.

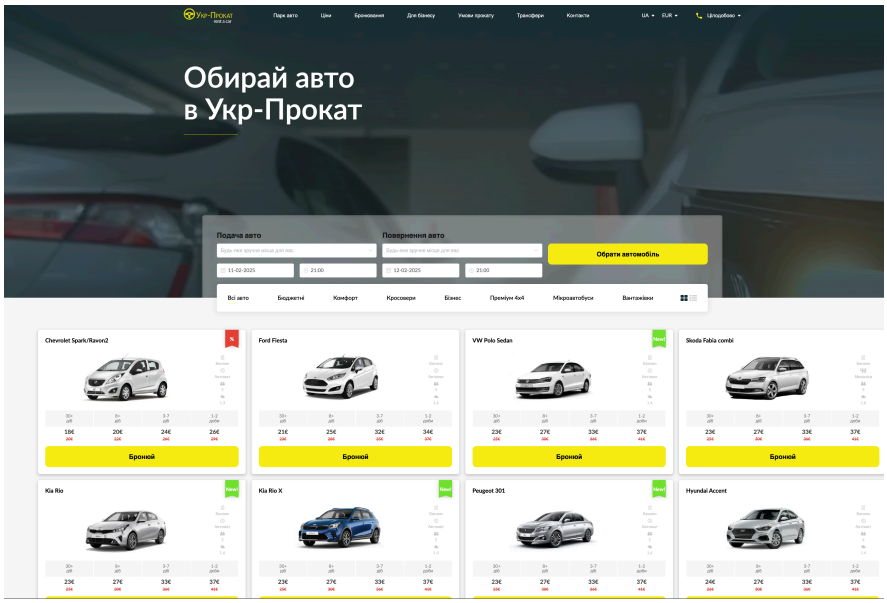


Рис. 1.3 – Вигляд головної сторінки вебсайту Ukr-Prokat

Ukr-Prokat є українською мережею прокату автомобілів, яка функціонує в багатьох містах країни, пропонуючи своїм клієнтам широкий вибір транспортних засобів від економ-класу до преміум-сегменту, що дозволяє задовольнити потреби як звичайних мандрівників, так і тих, хто шукає

комфортніші умови. Платформа вирізняється можливістю замовлення автомобіля з водієм, що є зручним для тих, хто не бажає керувати самостійно, а також пропонує гнучкі умови оренди, включаючи як короткострокові, так і довгострокові угоди, адаптовані до різних сценаріїв використання. Страхування включене до вартості оренди, що забезпечує додатковий спокій для користувачів, а цілодобова підтримка клієнтів гарантує вирішення будь-яких питань у будь-який час. Система сортування автомобілів за популярністю, ціною та класом полегшує вибір, роблячи процес максимально зручним.

Переваги сервісу проявляються в розгалуженій мережі пунктів прокату, що охоплює значну частину України, забезпечуючи доступність для жителів різних регіонів, а також у відносно доступних цінах, які роблять оренду привабливою для широкої аудиторії, і прозорих умовах, що виключають приховані платежі та сприяють довірі до платформи. Водночас недоліки полягають у меншому асортименті преміум-автомобілів, що може розчарувати клієнтів, які шукають розкішні моделі, а також у можливих обмеженнях щодо міжміських поїздок, які можуть ускладнити плани тих, хто планує подорожувати за межі одного міста. Загалом Ukr-Prokat демонструє сильні сторони в доступності та локальній адаптації, але потребує розширення пропозицій преміум-сегменту та гнучкості в географічному використанні автомобілів.

Таблиця 1.1

Таблиця порівняння аналогів

Функція / Сервіс	**K-Rent**	**Renty**	**Ukr-Prokat**
Оренда авто в Україні	+	-	+
Преміальні авто	+	+	-
Доставка авто	-	+	-
Мобільний додаток	+	+	-
Довгострокова оренда	+	-	+
Страхування включене	-	+	+

Продовження таблиці 1.1

Цілодобова підтримка	+	+	+
Фільтрація та сортування	+	+	+

За результатами аналізу платформ для оренди автомобілів, сервіс Renty вирізняється завдяки своїм унікальним можливостям і високому рівню зручності. Його орієнтація на ексклюзивні автомобілі, такі як розкішні моделі та спорткари, робить його привабливим для преміум-сегменту. Швидке оформлення оренди, можливість доставки авто до клієнта та підтримка різних способів оплати, включно з криптовалютою, забезпечують гнучкість і комфорт для користувачів. Цілодобова підтримка клієнтів і розширена система фільтрації дозволяють швидко знаходити потрібний автомобіль, що особливо цінно для туристів і бізнесменів.

K-Rent демонструє сильні сторони у широкому виборі автомобілів різних класів і наявності додаткових послуг, таких як страхування чи GPS, але обмежена географічна доступність у менших містах знижує його універсальність. Ukr-Prokat пропонує доступні ціни та широку мережу пунктів прокату по Україні, але менший вибір преміум-авто та обмеження на міжміські поїздки роблять його менш конкурентоспроможним у порівнянні з Renty.

Для розробки власної конкурентоспроможної ми оренди автомобілів варто взяти за основу сильні сторони Renty, зокрема швидке оформлення бронювання, підтримку мультимовності та мультивалютності для глобальної аудиторії, а також зручну систему фільтрації. Доцільно додати елементи, які підвищують доступність, як у K-Rent, наприклад, широкий вибір автомобілів для різних бюджетів і додаткові послуги для комфорту. Прозорі умови оренди та велика географічна покритість, як у Ukr-Prokat, також стануть важливими для залучення широкої аудиторії. Власна платформа вирізняється акцентом на безпеку транзакцій через сучасні протоколи шифрування, адаптивний дизайн для зручного використання на будь-яких пристроях, а також інтеграцією

мультимовності та мультивалютності для забезпечення доступності для міжнародних користувачів. Це дозволить створити ефективний, безпечний і зручний сервіс, який відповідатиме потребам широкого кола клієнтів.

1.3. Вибір архітектури веб-орієнтованої системи

Розробка веб-застосунку для оренди автомобілів потребує ретельного вибору архітектури, яка забезпечить високу продуктивність, масштабованість і зручність для подальшого розвитку проєкту. Архітектура визначає, як компоненти системи взаємодіють між собою, як обробляються дані та як система адаптується до зростаючої кількості користувачів. Для забезпечення ефективної роботи системи обрано клієнт-серверну архітектуру, яка чітко розмежовує функціонал між серверною та клієнтською частинами, забезпечуючи гнучкість, безпеку та швидкодію. Така модель є оптимальною для веб-додатків, оскільки дозволяє обробляти запити користувачів на сервері, зберігаючи дані в централізованій базі, тоді як клієнтська частина відповідає за відображення інтерфейсу та взаємодію з користувачем.

Серверна частина системи побудована на основі фреймворку Laravel, який є одним із провідних інструментів для створення веб-додатків на PHP. Laravel вирізняється модульною структурою, що полегшує організацію коду та його підтримку. Завдяки Eloquent ORM (Object-Relational Mapping) забезпечується зручна робота з базою даних через об’єктно-орієнтований підхід. Eloquent спрощує створення складних запитів, наприклад, для фільтрації автомобілів за маркою, класом, ціною чи доступністю у певному місті, а також автоматизує зв’язки між таблицями, такими як користувачі, автомобілі та бронювання. Laravel також пропонує вбудовані інструменти для маршрутизації, аутентифікації, авторизації та обробки платежів, що підвищує безпеку системи та скорочує час розробки. Наприклад, вбудована система авторизації захищає доступ до профілів користувачів і історії бронювань, а валідація даних забезпечує коректність інформації, такої як дати оренди чи платіжні дані. Крім того, Laravel підтримує інтеграцію з платіжними системами, що дозволяє

безпечно обробляти транзакції в різних валютах, а також забезпечує мультимовність через локалізацію, що відповідає потребам глобальної аудиторії.

Для порівняння було розглянуто кілька архітектурних підходів: монолітну архітектуру, мікросервісну архітектуру та серверлес-архітектуру. Монолітна архітектура об’єднує всі компоненти системи в єдиному кодовому блоці, що спрощує розробку та розгортання, але може ускладнити масштабування при значному зростанні навантаження. Мікросервісна архітектура розбиває систему на незалежні сервіси, кожен із яких відповідає за окрему функцію, наприклад, управління бронюваннями чи обробку платежів. Вона забезпечує високу масштабованість, але потребує складнішого управління та більших ресурсів на старті. Серверлес-архітектура передає управління серверною інфраструктурою стороннім сервісам, таким як AWS, Lambda, але менш підходить для проєктів із чіткою структурою бази даних і складною логікою, як у системи для оренди автомобілів. З огляду на потреби проєкту — простоту розробки, помірне навантаження та необхідність швидкого розгортання — обрано монолітну архітектуру на базі Laravel. Вона дозволяє швидко реалізувати всі необхідні функції, забезпечує легкість у підтримці коду та дає можливість у майбутньому перейти до мікросервісів, якщо це стане необхідним.

Клієнтська частина системи реалізована з використанням фреймворку Bootstrap, який забезпечує створення адаптивного та інтуїтивно зрозумілого інтерфейсу. Bootstrap дозволяє розробляти інтерфейс, що коректно відображається на пристроях із різними розмірами екранів — від смартфонів до десктопів. Готові компоненти Bootstrap, такі як форми для бронювання, фільтри пошуку чи модальні вікна для вибору додаткових послуг, прискорюють розробку та забезпечують сучасний вигляд системи. Наприклад, модальні вікна використовуються для підтвердження бронювання чи звернень до служби підтримки. Для адміністративної панелі застосовано інструмент Craftable, який інтегрується з Laravel і автоматично генерує шаблони та компоненти для управління користувачами, автомобілями, бронюваннями та платежами. Craftable значно скорочує час розробки адмінпанелі, надаючи зручний інтерфейс

для виконання CRUD-операцій (Create, Read, Update, Delete) і забезпечуючи гнучке налаштування, наприклад, для керування доступністю авто чи обробки запитів клієнтів.

Обрана клієнт-серверна архітектура на базі Laravel із Eloquent ORM та Craftable для адмінпанелі створює надійну, гнучку та масштабовану систему. Ця архітектура забезпечує оптимальну взаємодію між компонентами, високу продуктивність і можливість швидкого впровадження нових функцій, таких як підтримка мультимовності та мультивалютності, що відповідає потребам проєкту «Система управління орендою автомобілів».

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для створення веб-системи оренди автомобілів підібрано технологічний стек, який поєднує сучасні інструменти для забезпечення швидкості, надійності, безпеки та адаптивності до потреб глобальної аудиторії. Вибір технологій зосереджено на створенні зручного, інтуїтивного сервісу, який підтримує мультимовність, мультивалютність і відповідає вимогам масштабованості та простоти підтримки.

Серверна частина системи розробляється на основі фреймворку Laravel, визнаного інструменту для створення веб-додатків на PHP. Його модульна структура полегшує організацію коду та забезпечує зручність у подальшій підтримці. Завдяки Eloquent ORM забезпечується ефективна взаємодія з базою даних через об’єктно-орієнтований підхід. Це дозволяє легко створювати запити, наприклад, для пошуку автомобілів за маркою, ціною чи містом, а також керувати зв’язками між таблицями, такими як користувачі, бронювання та автомобілі. Laravel пропонує вбудовані інструменти для аутентифікації, авторизації, обробки платежів і захисту даних, що скорочує час розробки та підвищує безпеку. Наприклад, валідація даних гарантує коректність інформації про бронювання чи платежі, а вбудована підтримка локалізації забезпечує мультимовність для користувачів із різних країн. Інтеграція з платіжними системами дозволяє безпечно обробляти транзакції в різних валютах, що є

важливим для міжнародного сервісу.

Клієнтська частина системи створюється з використанням шаблонізатора Blade, який є частиною Laravel, у поєднанні з препроцесором SASS для стилізації. Blade забезпечує створення гнучких і повторно використовуваних шаблонів для сторінок, таких як каталоги автомобілів, форми бронювання чи особисті кабінети, що робить код більш структурованим і прискорює розробку. SASS додає гнучкості у дизайні завдяки змінним, вкладеним стилям і міксинам, дозволяючи створювати адаптивний інтерфейс, який виглядає сучасно та працює на різних пристроях — від смартфонів до комп'ютерів. Наприклад, форми для бронювання чи фільтри пошуку стилізуються через SASS, щоб забезпечити зручність і привабливий вигляд.

Адміністративна панель системи створюється за допомогою інструменту Craftable, який інтегрується з Laravel і дозволяє швидко генерувати інтерфейс для управління сутностями, такими як користувачі, автомобілі, бронювання чи платіжні операції. Craftable автоматизує створення CRUD-операцій (створення, читання, оновлення, видалення), що значно економить час на розробку адмінпанелі. Наприклад, адміністратори можуть легко керувати доступністю автомобілів, обробляти запити клієнтів чи редагувати дані через зручний інтерфейс. Craftable також дозволяє гнучко налаштовувати панель під специфічні потреби проєкту.

Для зберігання даних використовується реляційна база даних MySQL, яка ідеально інтегрується з Laravel через Eloquent ORM. MySQL забезпечує швидку обробку запитів і надійне управління даними, що необхідно для роботи з великою кількістю автомобілів, бронювань і користувачів. Наприклад, зв'язки між таблицями дозволяють швидко отримувати інформацію про доступні автомобілі в певному місті чи історію бронювань користувача. MySQL гарантує стабільність і можливість масштабування, що є важливим для системи з потенційно великою аудиторією.

Обраний технологічний стек — Laravel з Eloquent ORM, MySQL, Blade, SASS і Craftable — створює надійну основу для розробки ефективної, безпечної

та зручної системи управління оренди автомобілів. Ці інструменти дозволяють реалізувати ключові функції, такі як пошук і бронювання автомобілів, підтримка мультимовності й мультивалютності, безпечна обробка платежів і зручне адміністрування. Такий підхід забезпечує швидку розробку, простоту підтримки та можливість додавання нових функцій у майбутньому, що відповідає цілям проєкту.

Висновки до першого розділу

У першому розділі визначено ключові аспекти розробки веб-системи для оренди автомобілів, а також сформульовано основні технічні вимоги до проєкту. Проведено аналіз аналогічних сервісів, таких як K-Rent, Renty та Ukr-Prokat, що дозволило оцінити їхні функціональні можливості, виявити сильні сторони, такі як зручна фільтрація та широкий вибір автомобілів, а також недоліки, зокрема обмежену географічну доступність чи високу вартість оренди. На основі цього аналізу сформовано вимоги до системи, які включають підтримку мультимовності, мультивалютності, зручний інтерфейс, безпечні транзакції та адаптивність для різних пристроїв.

Для реалізації проєкту обрано клієнт-серверну архітектуру, яка чітко розділяє функціонал між серверною та клієнтською частинами. Серверна частина, побудована на фреймворку Laravel, відповідає за обробку бронювань, управління даними користувачів і автомобілів, а також інтеграцію з платіжними системами для безпечної обробки транзакцій у різних валютах. Клієнтська частина взаємодіє із сервером через REST API, забезпечуючи швидку та ефективну передачу даних. Для зберігання інформації обрано реляційну базу даних MySQL, яка, завдяки інтеграції з Eloquent ORM, забезпечує швидку обробку складних запитів і надійне управління даними про автомобілі, бронювання та користувачів. MySQL вирізняється високою продуктивністю та масштабованістю, що є критично важливим для системи з потенційно великою кількістю користувачів.

Для створення системи використовуються сучасні технології: Laravel для

серверної частини, що забезпечує модульну організацію коду, безпеку та підтримку мультимовності через вбудовану локалізацію. Клієнтська частина розробляється з використанням шаблонізатора Blade у поєднанні з SASS для створення адаптивного та сучасного інтерфейсу, який коректно відображається на різних пристроях. Для адміністративної панелі застосовується інструмент Craftable, який інтегрується з Laravel і автоматизує створення інтерфейсу для управління користувачами, автомобілями та бронюваннями, спрощуючи виконання CRUD-операцій. Такий технологічний стек дозволяє створити стабільну, безпечну та гнучку систему для оренди автомобілів, яка відповідає сучасним стандартам ринку та забезпечує зручність для користувачів і адміністраторів.

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ

2.1. Визначення варіантів використання та об’єктно-орієнтованої структури системи

Веб-застосунок для оренди автомобілів створена для того, щоб користувачі могли легко знаходити, бронювати та оплачувати автомобілі для оренди. Користувачі поділяються на звичайних, які можуть лише переглядати доступні автомобілі, підтверджених, які мають можливість бронювати автомобілі та отримувати доступ до контактної інформації власників, а також адміністраторів, які керують системою, підтверджують користувачів і забезпечують її стабільну роботу.

Основна мета розробки системи полягає у створенні зручного веб-середовища, яке спрощує процес оренди автомобілів. Система забезпечує швидкий пошук автомобілів, гнучку фільтрацію за параметрами, такими як марка, клас чи місто, а також безпечне бронювання та оплату з підтримкою мультимовності та мультивалютності. Це дозволяє користувачам з усього світу комфортно взаємодіяти з сервісом, роблячи процес оренди простим і доступним для всіх.

Для незареєстрованих користувачів передбачено можливість переглядати списки автомобілів, переходити за категоріями, брендом, кузовом чи містом, а також ознайомлюватися з деталями про автомобілі та їхніх власників. Це дозволяє відвідувачам сайту швидко знаходити потрібну інформацію без необхідності реєстрації.

Зареєстровані користувачі можуть редагувати свій профіль, створювати запити на бронювання автомобілів, переглядати історію своїх бронювань, а також звертатися до служби підтримки чи подавати скарги через спеціальні форми.

Підтверджені користувачі отримують додаткову перевагу — доступ до контактної інформації власників автомобілів, що полегшує організацію оренди та комунікацію між сторонами.

Адміністратори системи мають повний контроль над системою. Вони відповідають управління списком автомобілів, бронюваннями, платіжними операціями, а також керують містами, регіонами та статичними сторінками сайту, такими як «Питання та відповіді» чи «Умови використання».

Функціональні вимоги до системи включають підтримку процесів реєстрації та авторизації користувачів. Система дозволяє створювати, редагувати та скасовувати бронювання, а також переглядати історію бронювань із їхніми статусами. Передбачено фільтрацію автомобілів за маркою, класом, ціною, містом чи іншими параметрами. Користувачі можуть відправляти запити на бронювання, при цьому контактна інформація залишається прихованою для всіх, окрім автора запиту, власника автомобіля та адміністратора. Підтверджені користувачі отримують доступ до контактів власників. Для зручності подачі скарг чи звернень до підтримки використовуються модальні форми.

Нефункціональні вимоги стосуються зручності, продуктивності та масштабованості системи. Інтерфейс системи розроблено так, щоб нові користувачі могли розібратися з ним за 3–5 хвилин, а досвідчені — за менше ніж 2 хвилини. Час відповіді системи на дії користувача не перевищує 3 секунд. Дизайн інтерфейсу є інтуїтивно зрозумілим, з чітким поділом на розділи пошуку, бронювання та профілю, адаптивним для різних пристроїв і мінімалістичним за стилем. Система розрахована на зростання кількості користувачів і бронювань, забезпечуючи надійне збереження даних.

Системні вимоги до середовища виконання передбачають наявність щонайменше 4 ГБ оперативної пам’яті, процесора з частотою від 1.6 GHz, стабільного інтернет-з’єднання та сучасних веббраузерів. На серверній стороні використовується реляційна база даних MySQL для зберігання даних про користувачів, автомобілі, бронювання, регіони, міста та статичні сторінки. Для реалізації серверної логіки застосовується фреймворк Laravel.

Для створення бронювання користувачам необхідно зареєструватися, якщо вони ще не мають облікового запису, або авторизуватися, якщо акаунт уже створено.

Таблиця 2.1

Таблиця прецеденту “Реєстрація/Авторизація”

Назва варіанту використання	Реєстрація/Авторизація
Основні актори	Користувач
Короткий опис	Користувачі мають зареєструватися, якщо ще не мають акаунту, або авторизуватися для створення бронювання автомобіля.

Керування бронюванням дозволяє користувачам створювати або скасовувати свої замовлення на оренду автомобілів.

Таблиця 2.2

Таблиця прецеденту “Керування бронюваннями”

Назва варіанту використання	Керування бронюваннями
Основні актори	Користувач
Короткий опис	Користувач може створювати або скасовувати бронювання автомобіля.

Перегляд і сортування автомобілів забезпечують зручний пошук транспортних засобів за потрібними параметрами.

Таблиця 2.3

Таблиця прецеденту “Перегляд і сортування автомобілів”

Назва варіанту використання	Перегляд і сортування автомобілів
Основні актори	Користувач
Короткий опис	Перегляд і сортування автомобілів забезпечують зручний пошук транспортних засобів за потрібними параметрами.

Процес подання запиту на бронювання детально розкриває, як користувач взаємодіє з автомобілями, доступними для оренди через систему. Таблиця чітко

демонструє, як створюється запит із включенням контактної інформації, зберігаючи при цьому її конфіденційність для всіх, окрім власника авто, адміністратора та того, хто подав запит.

Таблиця 2.4

Таблиця прецеденту “Подання запиту на бронювання ”

Назва варіанту використання	Подання запиту на бронювання
Основні актори	Користувач
Короткий опис	Користувач може подати запит на бронювання, заповнивши форму з ім’ям та контактами. Запит відображається в системі, але контактні дані залишаються недоступними для сторонніх.

Функціонал перегляду профілю та історії оренди користувача забезпечує зручний доступ до особистих даних, при цьому інші користувачі не мають права переглядати цей профіль. Таблиця детально пояснює, як користувач може переглянути власний профіль або історію своїх оренд, отримуючи ключову інформацію для особистого використання

Таблиця 2.5

Таблиця прецеденту “ Перегляд профілю та історії оренди користувача”

Назва варіанту використання	Перегляд профілю та історії оренди користувача
Основні актори	Користувач
Короткий опис	Користувач може переглянути власний профіль та історію своїх оренд автомобілів, тоді як доступ для інших користувачів обмежено.

Адміністративне керування охоплює різноманітні інструменти та функції, призначені для злагодженого контролю системи адміністратором. Таблиця

детально описує повноваження адміністратора, включаючи підтвердження користувачів, управління автомобілями, бронюваннями, запитами, регіонами, містами та статичними сторінками, що гарантує стабільність і ефективність системи.

Таблиця 2.6

Таблиця прецеденту “Адміністративне керування”

Назва варіанту використання	Адміністративне керування
Основні актори	Адміністратор
Короткий опис	Адміністратор може скасовувати замовлення, робити crud операції з автівками, брендами, кузовами та іншими параметрами системи.

На основі аналізу функціональних потреб визначено основні сутності, які підтримують функціонування системи. Діаграма класів, що відображає їхні взаємозв’язки, наведена на рисунку 2.1.

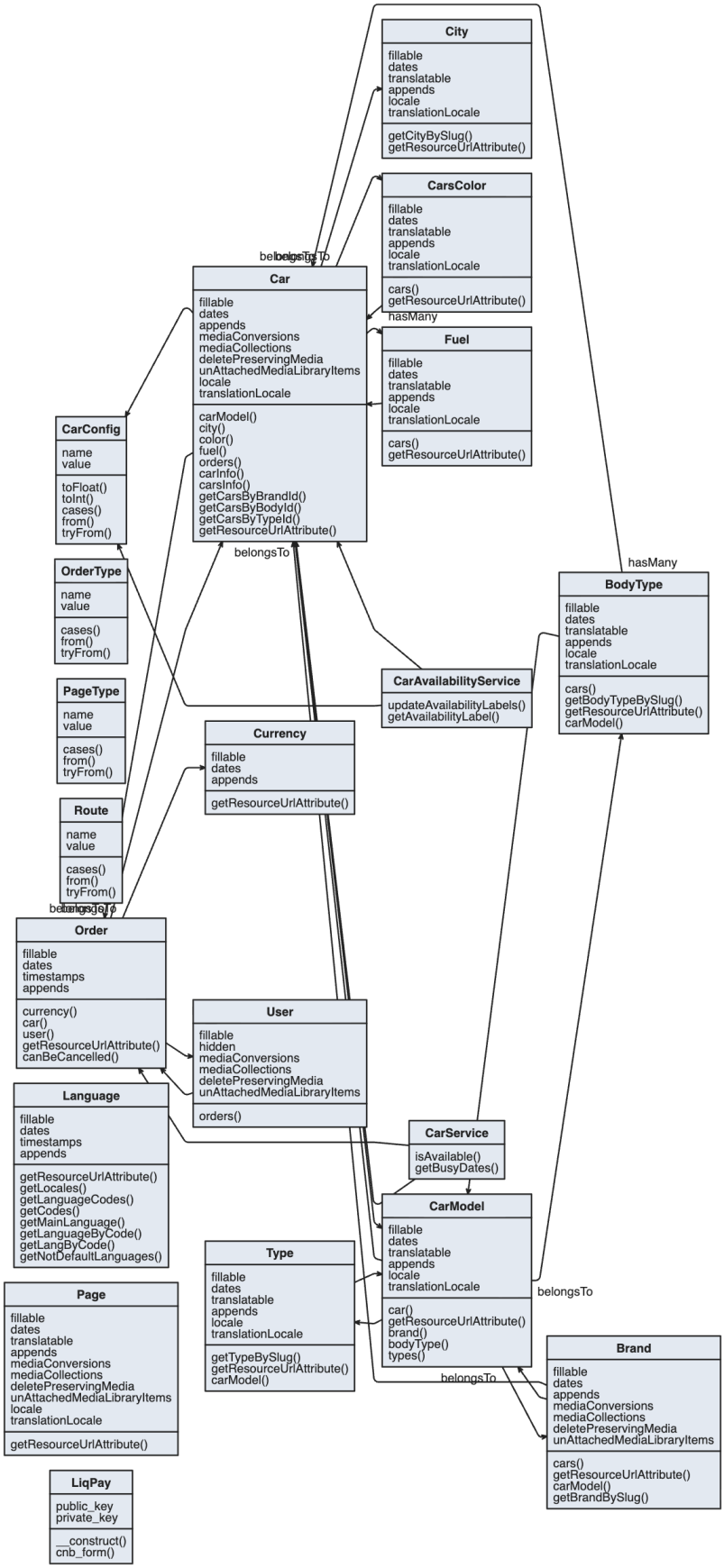


Рис. 2.1. Діаграма класів

Клас User відповідає за користувача системи. Він управляє інформацією про профіль, включаючи персональні дані, такі як ім’я, електронна адреса, пароль, номер телефону, а також статус верифікації. Метод orders() повертає список замовлень користувача. Метод isCarService() перевіряє, чи є користувач постачальником послуг оренди автомобілів. Метод getBusyDates() повертає дати, коли користувач зайнятий (наприклад, через активні оренди). Атрибути fillable дозволяють заповнювати поля як name, email, phone; dates керують часовими мітками; hidden приховує чутливі дані як password; mediaCollections обробляє медіафайли, наприклад, фото профілю; appends додає обчислювані атрибути; deletePreservingMedia видаляє користувача без видалення медіа.

Клас BodyType відповідає за типи кузовів автомобілів, такі як седан, хетчбек чи позашляховик. Метод getBodyTypeBySlug() повертає тип кузову за унікальним ідентифікатором (slug). Атрибути fillable дозволяють заповнювати дати; translatable та appends керують перекладами та обчислювальними полями; locale та translationLocale підтримують багатомовність.

Клас Car відповідає за автомобілі, доступні для оренди. Він є центральним елементом, зберігаючи технічні характеристики, геолокацію, ціни для різних періодів, депозит та інші параметри. Метод carModel() повертає пов’язану модель автомобіля. Метод carInfo() розширює об’єкт автомобіля, додаючи фотографії, деталі з пов’язаних таблиць і обчислені ціни на основі конфігурацій. Атрибути fillable дозволяють заповнювати дати та параметри; appends додає обчислювані поля; mediaConversions керує перетвореннями медіа; deletePreservingMedia видаляє без втрати медіа; locale та translationLocale підтримують переклади.

Клас CarModel відповідає за моделі автомобілів. Він пов’язує моделі з марками та типами кузовів, зберігаючи ключові технічні характеристики. Метод car() повертає пов’язані автомобілі. Метод brand() повертає пов’язану марку. Метод types() повертає пов’язані типи. Атрибути fillable дозволяють заповнювати дати; translatable та appends керують перекладами; locale та translationLocale підтримують багатомовність.

Клас City відповідає за міста, де доступні автомобілі. Він забезпечує географічну прив’язку та фільтрацію. Метод getCityBySlug() повертає місто за slug. Атрибути fillable дозволяють заповнювати дати; translatable та appends керують перекладами; locale та translationLocale підтримують багатомовність.

Клас CarsColor відповідає за кольори автомобілів. Він зберігає коди кольорів та назви для фільтрації. Метод hasMany() вказує на відношення "має багато" (наприклад, автомобілів). Атрибути fillable дозволяють заповнювати дати; translatable та appends керують перекладами; locale та translationLocale підтримують багатомовність.

Клас Fuel відповідає за типи палива автомобілів, такі як бензин, дизель чи електрика. Атрибути fillable дозволяють заповнювати дати; translatable та appends керують перекладами; locale та translationLocale підтримують багатомовність.

Клас Order відповідає за замовлення оренди. Він відстежує дати, адреси, статус, оплату та валюту. Метод currency() повертає пов’язану валюту. Метод car() повертає пов’язаний автомобіль. Метод canBeCancelled() перевіряє, чи можна скасувати замовлення. Метод generateForm() генерує форму оплати через LiqPay. Метод cancel() скасовує замовлення, якщо дозволено. Атрибути dates та timestamps керують часовими мітками; appends додає обчислювані поля.

Клас Brand відповідає за марки автомобілів. Він зберігає назви, іконки та кількість пов’язаних автомобілів для пошуку. Метод getBrandBySlug() повертає марку за slug. Метод belongsTo() вказує на відношення належності. Атрибути fillable дозволяють заповнювати дати; appends додає обчислювані поля; mediaConversions керує перетвореннями медіа; deletePreservingMedia видаляє без втрати медіа.

Клас Type відповідає за типи автомобілів, такі як економ-клас чи преміум. Метод getTypeBySlug() повертає тип за slug. Метод carModel() повертає пов’язані моделі. Атрибути fillable дозволяють заповнювати дати; translatable та locale підтримують переклади.

Клас Page відповідає за статичні сторінки платформи, такі як умови використання. Метод getPage() повертає вміст сторінки за її ідентифікатором або slug. Атрибути fillable дозволяють заповнювати дати; translatable та appends керують перекладами; deletePreservingMedia видаляє без втрати медіа; translationLocale підтримує локалізацію.

Клас Currency відповідає за валюти для оплати. Він зберігає курси обміну та символи для адаптації цін. Атрибути fillable дозволяють заповнювати дати; appends додає обчислювані поля.

Клас CarAvailabilityService відповідає за сервіс доступності автомобілів. Метод updateAvailabilityLabels() оновлює мітки доступності. Метод getAvailabilityLabel() повертає мітку статусу. Метод getBodyTypeBySlug() повертає тип кузову за slug (можливо, помилкове розміщення в діаграмі). Метод carModel() повертає модель автомобіля.

Клас LiqPay відповідає за інтеграцію з платіжною системою LiqPay. Метод construct() ініціалізує об’єкт з публічним та приватним ключами. Метод cnb_form() генерує форму для оплати з параметрами, такими як сума, валюта, опис.

Клас CarConfig відповідає за конфігурацію системи, наприклад, множники цін. Метод city() повертає пов’язане місто. Метод fuel() повертає тип палива. Метод orders() повертає замовлення. Метод cars() повертає автомобілі. Метод getCarsByBrand() повертає автомобілі за маркою. Метод belongsTo() вказує на відношення належності. Атрибути name та value зберігають конфігураційні значення.

Клас OrderType відповідає за типи замовлень (наприклад, статус оплати). Метод name та value зберігають назву та значення. Метод cases() повертає можливі випадки. Метод from() створює з значення. Метод tryFrom() намагається створити з значення.

Клас PageType відповідає за типи сторінок. Метод name та value зберігають назву та значення. Метод cases() повертає можливі випадки. Метод from() створює з значення. Метод tryFrom() намагається створити з значення.

2.2. Розробка бази даних системи

Для зберігання даних у додатку "Система управління орендою автомобілів" застосовується MySQL – реляційна система управління базами даних, яка гарантує надійне зберігання та обробку інформації про користувачів, пропозиції оренди, запити, звернення до служби підтримки, категорії, регіони, міста та статичні сторінки.

Опис усіх таблиць бази даних представлено в таблицях 2.7 – 2.16.

Таблиця User відповідає за управління інформацією про користувачів системи оренди автомобілів. Ця таблиця представляє профіль кожного учасника системи, включаючи персональні дані, такі як ім’я, електронна адреса, пароль, номер телефону, а також додаткову інформацію, як-от регіон і місто проживання чи місця активності. Він відображає статус користувача, зокрема чи був він підтверджений адміністратором, що впливає на доступ до певних функцій системи. Клас також слугує основою для ідентифікації користувачів у системі, забезпечуючи унікальність їхніх даних і контроль доступу до послуг оренди.

Таблиця 2.7

Таблиця Users

Поле	Тип	Призначення
id	bigint	Унікальний ідентифікатор користувача
name	string	Ім’я користувача
email	string	Електронна адреса
phone	string	Номер телефону
email_verified_at	datetime	Дата верифікації email
password	string	Зашифрований пароль
remember_token	string	Токен для бекенда
created_at	datetime	Дата створення запису
updated_at	datetime	Дата останнього оновлення

Таблиця BodyType відіграє ключову роль у класифікації автомобілів за типами кузовів, такими як седан, хетчбек чи позашляховик. Вона використовується для організації каталогів автомобілів і допомагає користувачам швидко знаходити транспортні засоби за їхньою конструктивною особливістю. Дані в цій таблиці підтримують багатомовність, дозволяючи зберігати назви типів кузовів у різних мовах, а також відстежують кількість пов’язаних автомобілів і їхній статус через часові мітки створення та оновлення.

Таблиця 2.8

Таблиця BodyType

Поле	Тип	Призначення
id	integer	Унікальний ідентифікатор типу кузову
slug	string	Унікальний ідентифікатор для URL-адрес
name	json	Назва типу кузову з підтримкою кількох мов
cars_count	integer	Кількість автомобілів, що належать цьому типу
icon	string	Шлях до іконки, яка представляє тип кузову

Таблиця Car є центральним елементом системи, призначеним для зберігання детальної інформації про автомобілі, доступні для оренди. Вона включає технічні характеристики, географічні координати, цінові пропозиції для різних періодів оренди, а також додаткові параметри, такі як кількість кілометрів, депозит і мінімальний термін оренди. Ця таблиця дозволяє адміністраторам та користувачам ефективно управляти пропозиціями, відстежувати статус автомобілів і забезпечувати їхню доступність на мапі

завдяки широті та довготі.

Таблиця 2.9

Таблиця Car

Поле	Тип	Призначення
id	integer	Унікальний ідентифікатор автомобіля
car_model_id	integer	Ідентифікатор моделі автомобіля
car_brand_id	integer	Ідентифікатор марки автомобіля
car_body_type_id	integer	Ідентифікатор типу кузову
car_slug	string	Унікальний ідентифікатор для URL-адрес
city_id	integer	Ідентифікатор міста, де розташований автомобіль
availability_label	string	Текстова мітка статусу доступності
longitude	string	Географічна довгота розташування
latitude	string	Географічна широта розташування
price_1	float	Вартість оренди за 1 день
price_7	float	Вартість оренди за 7 днів
price_30	float	Вартість оренди за 30 днів
price_31_more	float	Вартість оренди за 31+ днів
deposit	integer	Розмір депозиту для оренди
km_included_per_day	integer	Кількість включених кілометрів за день

Продовження таблиці 2.9

overlimit_charge_per_km	float	Плата за кожен кілометр понад ліміт
min_day_reservation	integer	Мінімальна кількість днів для бронювання
free_delivery	integer	Прапорець для безкоштовної доставки (0/1)
registration_number	string	Реєстраційний номер автомобіля
color_id	integer	Ідентифікатор кольору автомобіля
fuel_id	integer	Ідентифікатор типу палива
attribute_year	integer	Рік випуску автомобіля
attribute_seats	integer	Кількість пасажирських місць
attribute_1_to_100	float	Характеристика в діапазоні 0-100 (наприклад, економія палива)
attribute_max_speed	integer	Максимальна швидкість автомобіля
attribute_horsepower	integer	Потужність двигуна в кінських силах
attribute_transmission	string	Тип трансмісії (наприклад, автомат/механіка)
attribute_doors	integer	Кількість дверей
attribute_engine	string	Тип двигуна (наприклад, бензиновий/електричний)
attribute_baggage	integer	Кількість багажних відділень
status	boolean	Статус активності автомобіля (true/false)

Таблиця CarModel призначена для зберігання детальної інформації про моделі автомобілів, які представлені на системі. Вона пов’язує моделі з марками та типами кузовів, а також зберігає ключові технічні характеристики, такі як кількість місць, максимальна швидкість і тип трансмісії. Ця таблиця підтримує багатомовність назв і слугує основою для фільтрації пропозицій автомобілів за їхніми моделями.

Таблиця 2.10

Таблиця CarModel

Поле	Тип	Призначення
id	integer	Унікальний ідентифікатор моделі
slug	string	Унікальний ідентифікатор для URL-адрес
name	json	Назва моделі з підтримкою кількох мов
brand_id	integer	Ідентифікатор пов’язаної марки
body_type_id	integer	Ідентифікатор типу кузову
attribute_seats	boolean	Прапорець наявності сидінь (true/false)
attribute_1_to_100	float	Характеристика в діапазоні 0-100
attribute_max_speed	integer	Максимальна швидкість моделі
attribute_horsepower	integer	Потужність двигуна в кінських силах
attribute_transmission	string	Тип трансмісії (наприклад, автомат/механіка)
status	integer	Статус активності моделі (0/1)

Таблиця City призначена для зберігання даних про міста, де доступні автомобілі для оренди. Вона забезпечує географічну прив’язку транспортних засобів і користувачів, підтримує багатомовні назви та дозволяє відстежувати зміни в інформації про міста.

Таблиця 2.11

Таблиця City

Поле	Тип	Призначення
id	bigint	Унікальний ідентифікатор міста
slug	string	Унікальний ідентифікатор для URL-адрес
name	json	Назва міста з підтримкою кількох мов

Таблиця CarsColor призначена для зберігання інформації про доступні кольори автомобілів, включаючи їхні коди та назви в різних мовах. Вона допомагає користувачам фільтрувати автомобілі за кольором і забезпечує візуальну ідентифікацію через кольорові коди.

Таблиця 2.12

Таблиця CarsColor

Поле	Тип	Призначення
id	integer	Унікальний ідентифікатор кольору
slug	string	Унікальний ідентифікатор для URL-адрес
name	json	Назва кольору з підтримкою кількох мов
color_code	string	Код кольору в форматі (наприклад, HEX)

Таблиця Fuel призначена для зберігання даних про типи палива, які використовуються автомобілями на системі. Вона підтримує багатомовні назви та дозволяє користувачам фільтрувати пропозиції за типом палива, наприклад, бензин, дизель чи електрика.

Таблиця 2.13

Таблиця Fuel

Поле	Тип	Призначення
id	integer	Унікальний ідентифікатор типу палива
slug	string	Унікальний ідентифікатор для URL-адрес
name	json	Назва типу палива з підтримкою кількох мов

Таблиця Order призначена для зберігання детальної інформації про замовлення оренди автомобілів. Вона відстежує терміни оренди, адреси початку та завершення, статус оплати та загальну вартість, включаючи підтримку різних валют і інтеграцію з платіжними системами, такими як LiqPay.

Таблиця 2.14

Таблиця Order

Поле	Тип	Призначення
id	bigint	Унікальний ідентифікатор замовлення
user_id	bigint	Ідентифікатор користувача, який створив замовлення
car_id	integer	Ідентифікатор орендованого автомобіля
date_from	date	Дата початку оренди
date_to	date	Дата завершення оренди

Продовження таблиці 2.14

address_from	string	Адреса, звідки починається оренда
address_to	string	Адреса, куди повертається автомобіль
status	string	Поточний статус замовлення (наприклад, "active")
total_price	integer	Загальна сума оренди в обраній валюті
currency_id	integer	Ідентифікатор валюти
liqpay_payment_id	string	Ідентифікатор транзакції в LiqPay
payment_status	string	Статус оплати (наприклад, "paid"/"pending")

Таблиця Brand використовується для зберігання даних про марки автомобілів, доступних на системі. Вона включає назви марок, їхні іконки та кількість пов’язаних автомобілів, що полегшує пошук і класифікацію пропозицій.

Таблиця 2.15

Таблиця Brand

Поле	Тип	Призначення
id	integer	Унікальний ідентифікатор марки
slug	string	Унікальний ідентифікатор для URL-адрес
name	string	Назва марки автомобіля

Продовження таблиці 2.15

cars_count	Integer	Кількість автомобілів цієї марки
icon	string	Шлях до іконки марки

Таблиця Type призначена для зберігання даних про типи автомобілів, такі як клас (економ, преміум) або призначення (для міста, для подорожей). Вона підтримує багатомовність і слугує для розширеної фільтрації пропозицій.

Таблиця 2.16

Таблиця Type

Поле	Тип	Призначення
id	integer	Унікальний ідентифікатор типу
slug	string	Унікальний ідентифікатор для URL-адрес
name	json	Назва типу з підтримкою кількох мов

Таблиця Page використовується для зберігання даних про статичні сторінки платформи, такі як інформаційні розділи чи умови використання. Вона підтримує багатомовний вміст, включаючи заголовки, описи та ЧаПи, і дозволяє управляти видимістю сторінок.

Таблиця 2.17

Таблиця Page

Поле	Тип	Призначення
id	integer	Унікальний ідентифікатор типу
slug	string	Унікальний ідентифікатор для URL-адрес

Продовження таблиці 2.17

name	Json	Назва типу з підтримкою кількох мов

Таблиця Currency призначена для зберігання даних про валюти, які підтримуються для оплати на платформі. Вона включає обмінні курси та символи валют, що дозволяє адаптувати ціни для різних ринків і забезпечує гнучкість у транзакціях.

Таблиця 2.18

Таблиця Page

Поле	**Тип**	**Призначення**
id	integer	Унікальний ідентифікатор валюти
slug	string	Унікальний ідентифікатор для URL-адрес
sign	string	Символ валюти (наприклад, "$" або "€")
exchange_rate	float	Курс обміну відносно базової валюти

У процесі розробки системи оренди автомобілів важливим є належне моделювання зв’язків між користувачами, автомобілями та замовленнями. При створенні нового замовлення на оренду автомобіля, система зберігає відповідну інформацію в таблицю "orders" бази даних. До основних полів цієї таблиці належать: ідентифікатор користувача (user_id), ідентифікатор автомобіля (car_id), дата початку оренди (start_date), дата завершення (end_date) та статус замовлення (status).

Ці поля пов’язані з іншими таблицями за допомогою зовнішніх ключів.

Зокрема, поле user_id вказує на запис у таблиці "users", яка містить реєстраційні та контактні дані клієнтів, а поле car_id — на запис у таблиці "cars", що зберігає інформацію про наявні автомобілі, включно з їхніми технічними характеристиками, маркою, моделлю, роком випуску, кольором та типом пального.

Після успішного створення замовлення, воно стає доступним для перегляду через особистий кабінет користувача, який його створив. Адміністратор також має доступ до замовлень усіх користувачів. Водночас система реалізує механізми обмеження доступу до конфіденційної інформації. Наприклад, такі поля, як контактні дані користувача (users.phone), можуть переглядати лише автор замовлення або адміністратор. Подібна логіка реалізована на рівні програмного коду через механізми авторизації, що гарантує захист персональних даних.

2.3. Проєктування та реалізація алгоритмів роботи системи

Додаток "Система управління орендою автомобілів" вирізняється зручним інтерфейсом і пропонує весь необхідний набір функцій, які полегшують взаємодію між користувачами, пошук варіантів оренди та їхнє ефективне впорядкування. Роботу з системою детально роз’яснено в наступних пунктах.

Після авторизації адміністратор отримує доступ до спеціального інтерфейсу управління, який є основним інструментом для контролю над усіма аспектами платформи. Цей інтерфейс, реалізований на базі Craftable — інструменту для швидкого створення CRUD-інтерфейсів, інтегрованих з фреймворком Laravel, забезпечує повний доступ до адміністративної панелі. Адміністратор може не тільки переглядати, але й модифікувати або видаляти пропозиції оренди автомобілів, що включає скасування замовлень у разі порушень правил або за запитом користувача. Наприклад, при скасуванні замовлення система автоматично перевіряє статус (наприклад, чи є воно оплаченим чи ні) і оновлює базу даних, забезпечуючи синхронізацію з профілями користувачів. Це реалізовано через спеціальні методи в коді, такі як cancel в контролері Order, де перевіряється авторизація через middleware Laravel,

і статус змінюється на PAYMENT_CANCELLED, з подальшим сповіщенням користувача. Крім того, адміністратор уповноважений створювати, оновлювати чи видаляти категорії автомобілів (наприклад, седан, кросовер чи позашляховик), регіони (області України) та міста (з вказівкою координат на карті за допомогою Google Maps API), що дозволяє динамічно адаптувати платформу до нових ринкових потреб. Наприклад, при створенні нової категорії автомобілів адміністратор може вказати її slug для URL, назву на кількох мовах (враховуючи мультимовність системи) та пов’язати з брендами чи типами кузовів, забезпечуючи логічну структуру бази даних MySQL. Аналогічно, керування статичними сторінками включає редагування вмісту через адміністративну панель, де можна змінювати текст, додавати зображення чи видаляти сторінки за необхідності, наприклад, оновлювати FAQ чи правила оренди, з автоматичним збереженням змін у базі даних. Окрім цього, адміністратор має можливість переглядати звернення до служби підтримки, включаючи скарги на інших користувачів (наприклад, про невідповідність автомобіля опису чи порушення умов оренди), і реагувати на них залежно від ситуації — від надсилання попереджень до блокування акаунтів. Це інтегровано з системою сповіщень, де адміністратор може використовувати дошку замовлень (dashboard) для моніторингу статусів: червоні картки для скасованих, зелені для активних, з детальною інформацією про користувача, автомобіль та дати оренди. Інтерфейс адміністративної панелі починається з головної сторінки, де відображається дошка з трьома колонками етапів замовлень (створено, оплачено, завершено), що дозволяє швидко реагувати на проблеми. Натискаючи на картку замовлення, адміністратор переходить до детальної сторінки, де доступна вся інформація: характеристики автомобіля (марка, модель, рік, колір, паливо, кілометраж), дані користувача (контакти, історія оренд) та опції скасування без участі клієнта. Система налічує 12 ключових сутностей для редагування, таких як автомобілі, бренди, кузови, кольори, паливо, міста, регіони, валюти, користувачі, замовлення, запити та статичні сторінки. При редагуванні автомобіля, наприклад, адміністратор може вибирати модель, доступність,

прогресивні ціни (з коефіцієнтами 0.85 для 7+ днів і 0.7 для місяця, якщо поля не заповнені), заставу (яка автоматично додається до суми та повертається в кінці), обмеження кілометражу (стандартно 250 км/день), доплату за перевищення, мінімальний термін оренди, доставку, номерний знак, рік випуску, кількість сидінь та необов'язкові атрибути з моделі (якщо не вказані). Крім того, можна завантажувати фото через медіа-менеджер, з автоматичним стисненням у WebP для оптимізації, і вказувати координати на мапі для точного розташування. Безпека забезпечується CSRF-токенами в формах, автентифікацією через Laravel Auth та обмеженням доступу: контактні дані (наприклад, phone з таблиці users) лише адміністратору або автору. Це гарантує захист персональних даних відповідно до принципів GDPR-подібних стандартів. Загалом, роль адміністратора є центральною для підтримки стабільності системи, дозволяючи масштабування (наприклад, додавання нових автомобілів чи регіонів) і швидке реагування на інциденти, що робить платформу надійною та ефективною для всіх користувачів.

Незареєстрований користувач, або гість, має обмежений функціонал системи. Він може переглядати активні пропозиції оренди автомобілів і використовувати навігацію за категоріями, містами, типом кузову та брендами. Гість також може детально вивчати автомобілі та їхню доступність по датам, однак спроба орендувати автомобіль призводить до перенаправлення на сторінку авторизації, де потрібно ввійти в систему для продовження.

2.4. Реалізація функціоналу системи управління орендою автомобілів

Метод generateForm рис. 2.2 генерує форму оплати LiqPay для обробки замовлення, використовуючи дані з таблиці Order, включаючи суму, валюту та унікальний ідентифікатор. Ця функція інтегрується з платіжною системою, забезпечуючи безпечну транзакцію, і перенаправляє користувача на сторінку оплати з таймером на 10 хвилин. Успішна оплата фіксує замовлення в профілі користувача, тоді як невдача призводить до його скасування.

```
public function generateForm(Order $order)
```

```
{
    $liqpay = new LiqPay(env('LIQPAY_PUBLIC_KEY'), env('LIQPAY_PRIVATE_KEY'));
    $order->load('currency');
    $form = $liqpay->cnb_form([
        'action'       => 'pay',
        'amount'       => $order->total_price,
        'currency'     => strtoupper($order->currency->slug),
        'description'  => 'Оплата замовлення №' . $order->id,
        'order_id'     => $order->id,
        'version'      => '3',
        'server_url'   => str_replace('http://', 'https://', route('liqpay.callback')),
        'result_url'   => str_replace('http://', 'https://', route('orders.show', $order)),
    ]);
    return view('front.template-parts.liqpay_form', compact('form'));
}
```

Рис. 2.2. Метод generateForm

Метод cancel рис. 2.3 дозволяє користувачу скасувати замовлення, якщо він є його автором, перевіряючи права доступу та поточний статус замовлення. Функція включає захист від несанкціонованих дій через помилку 403 і забезпечує оновлення статусу на PAYMENT_CANCELLED, якщо скасування можливе, після чого перенаправляє користувача з відповідним повідомленням.

```
public function cancel(Request $request, Order $order)
{
    $auth_user = auth()->user();
    if ($order->user_id !== $auth_user->id) {
        abort(403, 'Unauthorized action.');
    }
    if (!$order->canBeCancelled()) {
        return redirect()->back()->withErrors(['cancel' => trans('front.order.cannot_cancel')]);
    }
    $order->status = \App\Enums\OrderType::PAYMENT_CANCELLED->value;
    $order->save();
    return redirect()->route('orders.show', $order)->with('success',
trans('front.order.cancelled_success'));
}
```

Рис. 2.3. Метод cancel

Метод carInfo рис. 2.4 розширює об’єкт автомобіля з таблиці Cars, додаючи фотографії та деталізовані дані, отримані через JOIN із таблицями CarModel, Brands, CarsColors і Fuels. Функція обробляє медіафайли, обчислює ціни на основі констант CarConfig і повертає оновлений об’єкт із назвами моделі, марки, кольору та типу палива, адаптованими для відображення.

```
public function carInfo()
{
```

```
        $media_photos = $this->getMedia('cars');
        $main_photo = null;
        $photos = [];
        if (isset($media_photos[0])) {
            $path_parts = pathinfo($media_photos[0]->getUrl('minifiedWebp'));
            $main_photo = str_replace(['-minifiedWebp', 'conversions/'], '', $path_parts['dirname'] .
'/' . $path_parts['basename']);
            foreach ($media_photos as $photo) {
                $path_parts = pathinfo($photo->getUrl('minifiedWebp'));
                $photos[] = str_replace(['-minifiedWebp', 'conversions/'], '', $path_parts['dirname'] . '/'
. $path_parts['basename']);
            }
        }
        $car = self::query()
            ->join('car_models', 'cars.car_model_id', '=', 'car_models.id')
            ->join('brands', 'car_models.brand_id', '=', 'brands.id')
            ->join('cars_colors', 'cars.color_id', '=', 'cars_colors.id')
            ->join('fuels', 'cars.fuel_id', '=', 'fuels.id')
            ->select(
                'cars.*',
                'car_models.name as car_model_name',
                'brands.slug as brand_slug',
                'cars_colors.slug as color_slug',
                'cars_colors.name as color_name',
                'cars_colors.color_code as color',
                'fuels.name as fuel_name'
            )
            ->where('cars.id', $this->id)
            ->first();
        $car['main_photo'] = $main_photo;
        $car['photos'] = $photos;
        $car['price_1'] = (int)$car['price_1'];
        $car['price_7'] = (int)($car['price_7'] > 0 ? $car['price_7'] : $car['price_1'] *
CarConfig::PRICE7_MULTIPLIER->toFloat());
        $car['price_30'] = (int)($car['price_30'] > 0 ? $car['price_30'] : $car['price_1'] *
CarConfig::PRICE30_MULTIPLIER->toFloat());
        $car['price_31_more'] = (int)($car['price_31_more'] > 0 ? $car['price_31_more'] :
$car['price_1'] * CarConfig::PRICE31MORE_MULTIPLIER->toFloat());
        $car['fuel_name'] = json_decode($car['fuel_name'])->en;
        return $car;
    }
```

Рис. 2.4. Метод carInfo

Висновки до другого розділу

У ході створення проекту "Система управління орендою автомобілів" здійснено ретельний аналіз потреб і розроблено об’єктно-орієнтовану структуру системи з урахуванням зв’язків між її компонентами та моделями. Встановлено реляційну базу даних на системі MySQL для безпечного збереження даних про

користувачів, варіанти оренди, запити та пов’язані об’єкти. Окреслено порядок дій для різних груп користувачів і ґрунтовно розглянуто впровадження ключових можливостей: обробка замовлень, сортування пропозицій, захист особистих даних у запитах та зручне адміністрування системою.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ УПРАВЛІННЯ ОРЕНДОЮ АВТОМОБІЛІВ

3.1. Порядок встановлення та налаштування параметрів платформи

Встановлення та налаштування системи оренди автомобілів передбачає поетапне розгортання її компонентів та конфігурацію для забезпечення стабільної роботи. До основних етапів належить підготовка серверного середовища, розгортання програмного коду, налаштування бази даних і конфігурація прав доступу.

Архітектура системи включає веб-додаток на основі фреймворку Laravel, систему керування адмініструванням на базі Craftable, базу даних MySQL для зберігання інформації про користувачів, автомобілі та замовлення, а також клієнтську частину, реалізовану із використанням Blade-шаблонів і стилізовану за допомогою препроцесора Sass. Усі компоненти розміщуються на серверному середовищі, розгорнутому на системі OpenServer або сумісному веб-сервері.

Для запуску системи необхідно встановити OpenServer або подібне середовище з підтримкою PHP (версії не нижче 8.1), MySQL (версія 8.0+) та Node.js. Після клонування репозиторію з вихідним кодом із GitLab, встановлюються залежності через Composer для серверної частини та npm для збирання фронтенду. Далі налаштовується файл оточення .env, у якому вказуються параметри підключення до бази даних (DB_HOST, DB_DATABASE, DB_USERNAME, DB_PASSWORD), а також генерується ключ додатка за допомогою команди:

php artisan key:generate

Для ініціалізації структури бази даних виконується команда:

php artisan migrate

Вона створює всі необхідні таблиці, зокрема: users, cars, orders, а також допоміжні довідники.

Також потрібно запустити jobs та cron, для успішних відкладених дій та статичних оновленнях даних з певними інтервалами (наприклад валюта)

php artisan queue:work

php artisan schedule:work

Клієнтська частина базується на Blade-шаблонах Laravel і використовує Sass для написання стилів, які компілюються через Laravel Mix. Адміністративна панель генерується за допомогою Craftable — інструменту для швидкого створення CRUD-інтерфейсів, інтегрованих із Laravel. Вона надає адміністраторам змогу управляти користувачами, замовленнями, автопарком тощо.

Система доступу реалізована за допомогою middleware Laravel. Звичайні користувачі можуть оформлювати замовлення, переглядати інформацію про автомобілі та власну історію оренди. Адміністратори мають розширені права на модерацію та управління всіма сутностями системи. Безпека системи гарантується використанням CSRF-токенів у формах, автентифікацією через Laravel Auth та можливістю підключення HTTPS для захисту даних при передачі.

Таким чином, система оренди автомобілів побудована на сучасному технологічному стеку Laravel + MySQL + Craftable + Sass, що забезпечує гнучкість, масштабованість та відповідність вимогам дипломного проєкту.

3.2. Структура інтерфейсу веб застосунку «Система управління орендою автомобілів»

Інтерфейс користувача в системі оренди автомобілів реалізовано у вигляді адаптивного веб-додатку, що забезпечує зручну взаємодію для різних ролей користувачів — від звичайних клієнтів до адміністраторів. Візуальна частина побудована на базі Blade-шаблонів у поєднанні зі стилями, написаними за допомогою препроцесора Sass. Такий підхід дозволяє досягти гнучкого дизайну з підтримкою різних типів пристроїв — настільних ПК, планшетів і мобільних телефонів. Це створює позитивний досвід користування і забезпечує швидкий доступ до основних функцій — перегляду автопарку, створення та керування замовленнями.

Для реалізації адміністративної панелі використано систему Craftable, яка тісно інтегрується з Laravel і дозволяє швидко сформувати повноцінний бекофіс.

Через неї адміністратори можуть керувати даними користувачів, автомобілів, замовлень і довідників, що значно спрощує процес супроводу та модерування системи.

Перше що бачить користувач на сайті, це головна сторінка (рис. 3.1). На ній розташований основний заголовок, список брендів, типів кузовів та лістинг з 16 автомобілів.

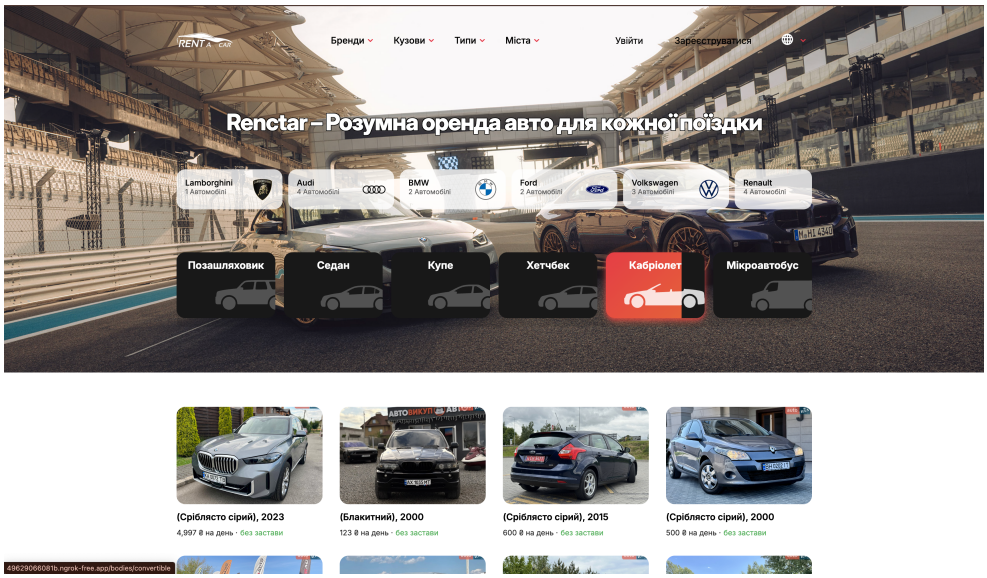


Рис. 3.1. Головна сторінка клієнської частини

Далі йде опис, з якого користувач може дізнатися про цю компанію, а також швидкі питання відповіді (рис. 3.2).

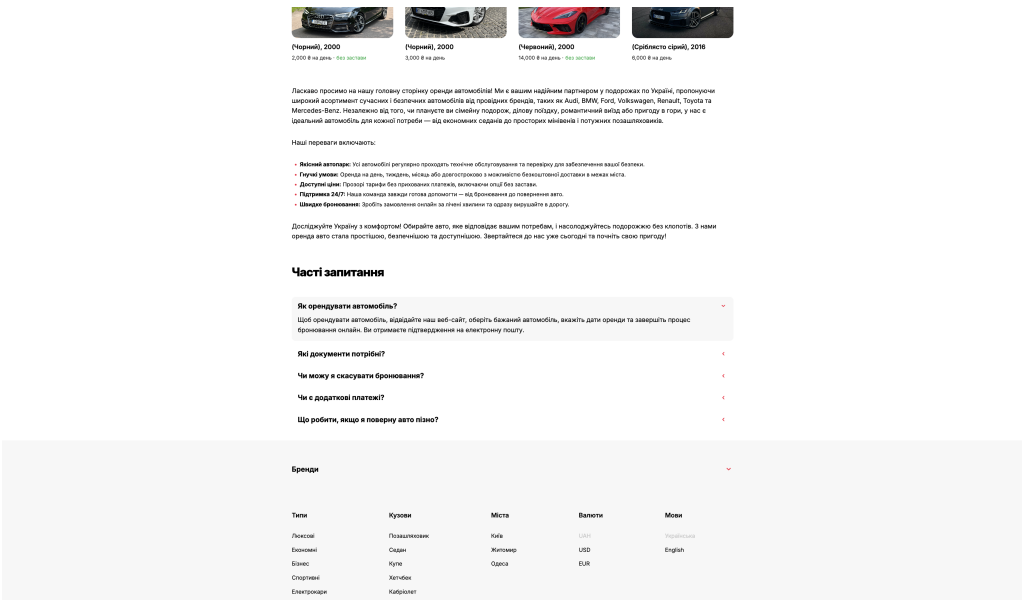


Рис. 3.2. Головна сторінка клієнської частини

З самого верху у нас є хедер (рис. 3.3), який є головною навігацією по сайту і включає в себе такі пункти:

1. Логотип компанії – посилання на головну сторінку
2. Бренди сайту
3. Автомобільні кузови
4. Типи автомобілів
5. Міста в яких можна орендувати автомобілі
6. Кнопки для реєстрації та авторизації, а також після авторизації на їх місці з’являється посилання на профіль

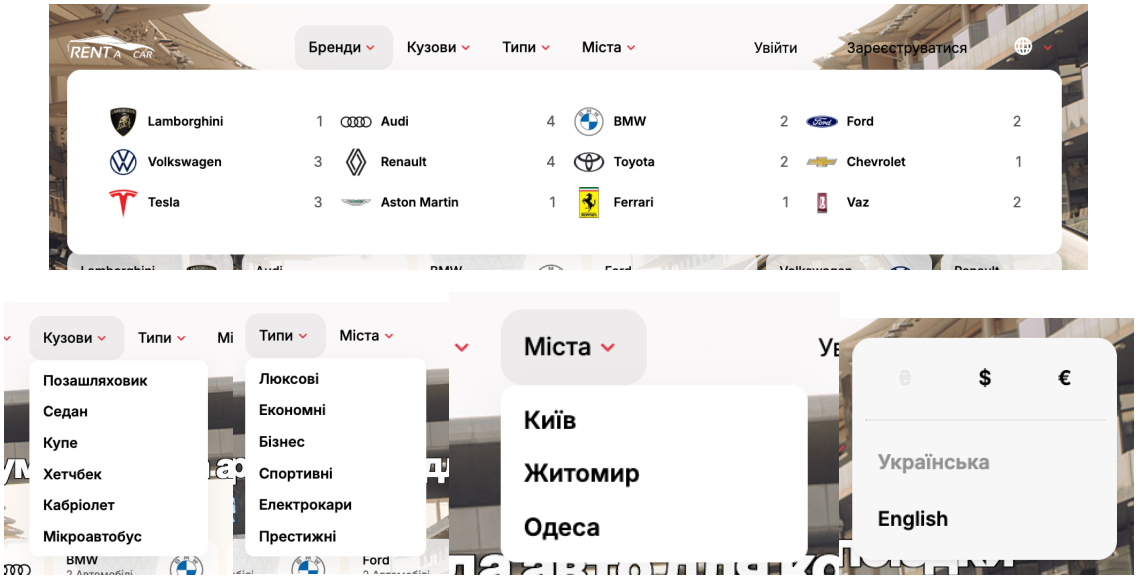


Рис. 3.3. Хедер, з усіма випадаючими меню

Також знизу присутній футер(рис. 3.4), який дублює більшість функціоналу з хедера.

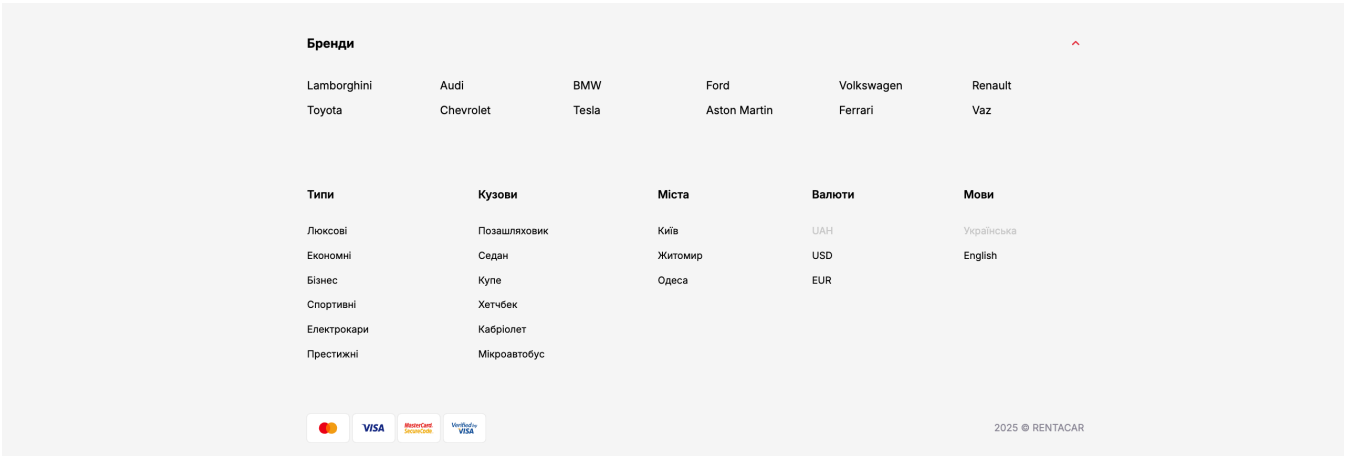


Рис. 3.4. Футер

Користувач може перейти по одному з посилань, наприклад по бренду Audi, і він опиниться на лістингу цього бренду (рис. 3.5). Де присутні картки автомобілів, сортування по ціні, додаткові авто(які можуть сподобатись) та інший текстовий контент для брендів.

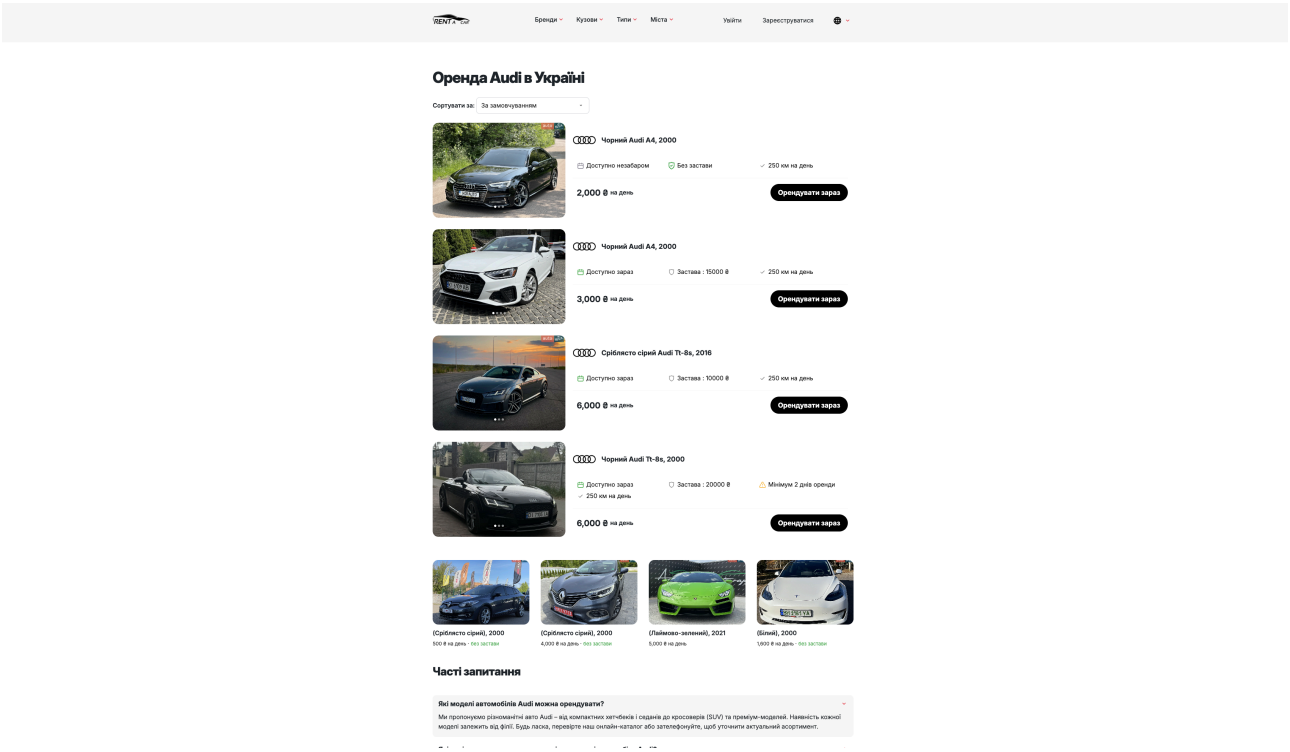


Рис. 3.5. Лістинг бренду Audi

На картці автівки вказується базова інформація її оренди (рис. 3.6). Присутні параметри доступності автівки, наявність застави, мінімальний період оренди, максимальний кілометраж на день, ціна та кнопка оренди.



Рис. 3.6 Картка автівки

Якщо користувач натисне на назву автівки, чи її фото, то він перейде на сторінку самої автівки (рис. 3.7). На цій сторінці, присутня вся інформація про автомобіль, його характеристики оптові ціни та де автомобіль знаходиться на мапі.

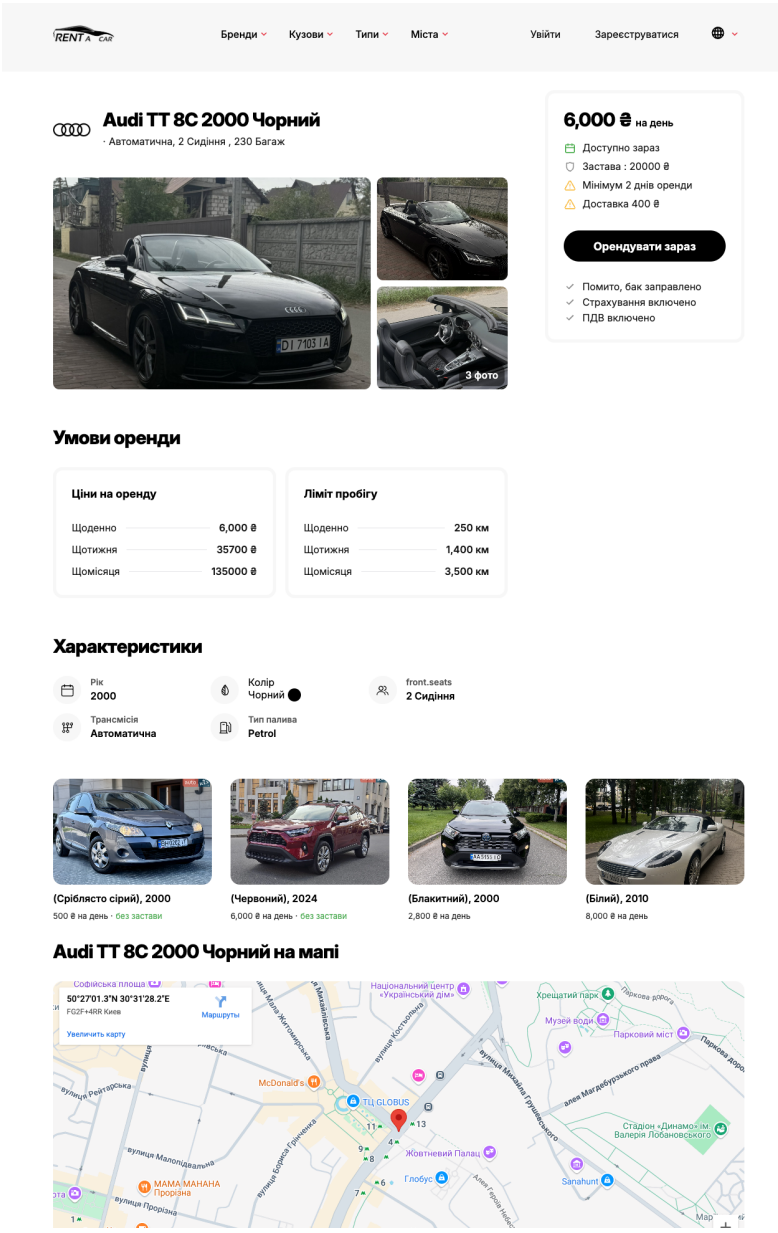


Рис. 3.7 Сторінка автівки

Також на даному сайті є мультимовність та мультивалютність. І якщо ми вибиремо англійську мову та валюту у євро, то інформація на сайті буде виглядати так(рис. 3.8).

Інтерфейс системи управління орендою автомобілів розроблений з урахуванням простоти та зручності використання. Адаптивний дизайн, сортування автомобілів, безпечна оплата та гнучке адміністрування забезпечують комфортну взаємодію для всіх користувачів, незалежно від їхнього статусу чи пристрою.

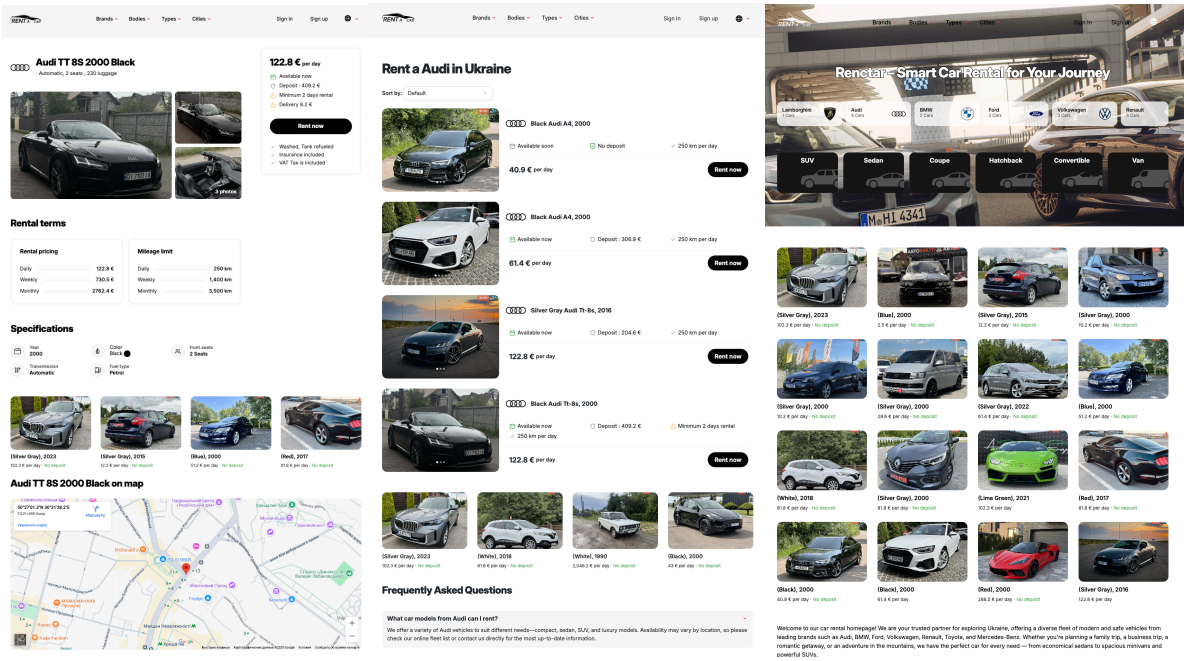


Рис. 3.8 Сторінки з іншою мовою та валютою

Якщо Користувач натисне на орендувати авто, то його перекине на сторінку авторизації, або він може зарєструватися, якщо ще не має акаунту на даній системі(рис. 3.9).

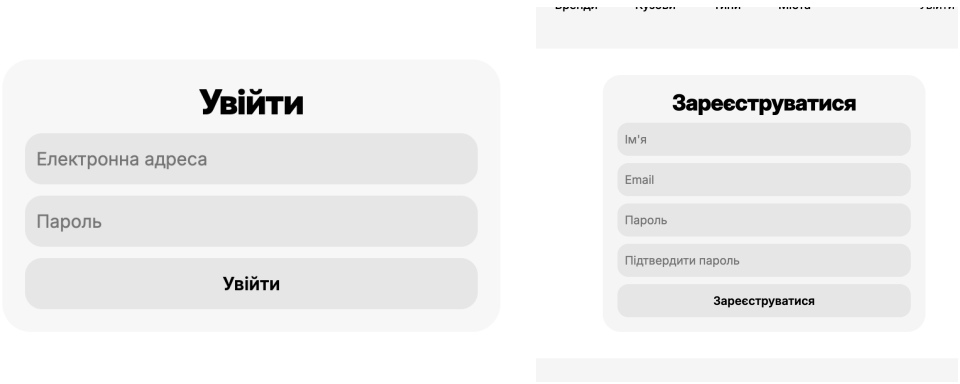


Рис. 3.9 Сторінки авторизації та реєстрації

Далі перейдемо до інтерфейсу адміністратора, де на головній сторінці нас зустрічає дошка з замовленнями (рис. 3.10). Де вказані 3 колонки етапів виконання замовлень. Червоний колір, значить що замовлення було створено, оплачено, але потім відмінено по певним причинам.

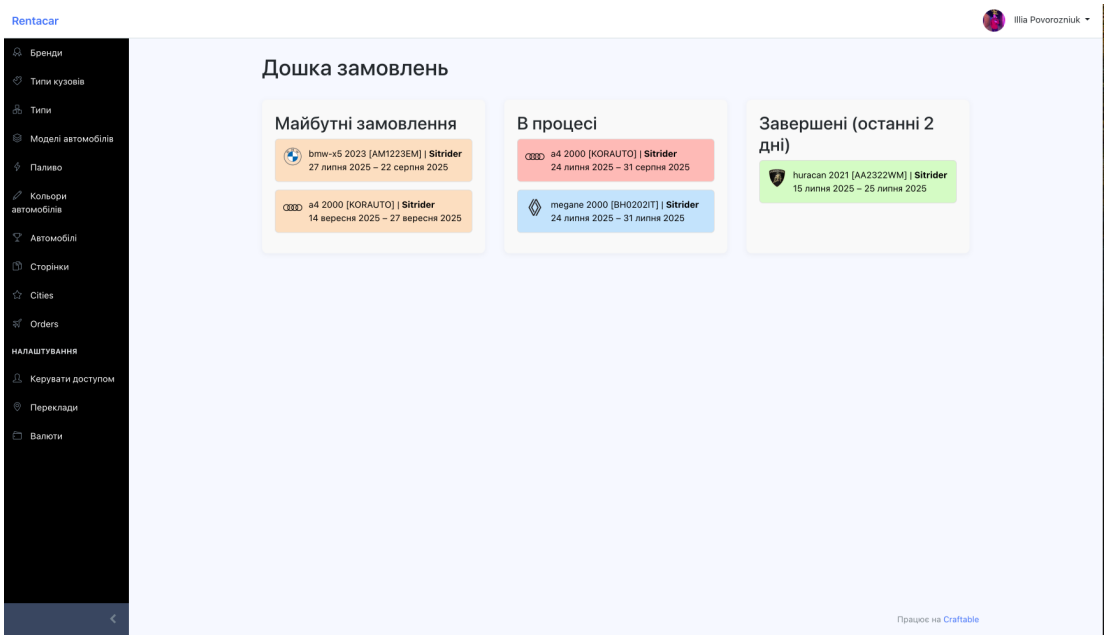


Рис. 3.10 Сторінки авторизації та реєстрації

Якщо натиснути на одну з карток, автомобілів, то адміністратор перейде на це замовлення(рис. 3.11), де буде детальна інформація про автомобіль, замовлення, та користувача. Також тут є можливість скасувати замовлення без участі користувача, якщо той порушив, або попросив закрити його.

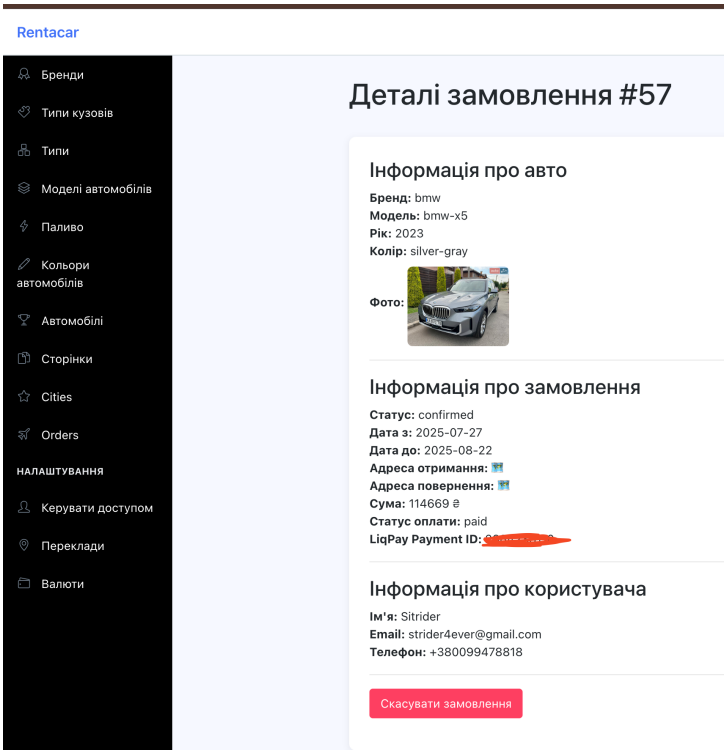


Рис. 3.11 Сторінки авторизації та реєстрації

У адміністратора, є 12 сутностей, які можна створювати, або редагувати.

Для наочного прикладу візьмемо сутнійсть автомобіля та його редагування (рис.3.12). У автомобілі ми можемо вибираим певну модель, доступність. Також тут діють прогресивні ціни, якщо поле ціна 7 не заповнене, то ціна за 7+ днів буде рахуватися за коефіцієнтом 0,85 за день від першої ціни, за місяць буде 0,7. Також можна вибирати заставу, якщо авто дороге і це автоматично включається в ціну та в кінці календарного місяця повертається на рахунок користувача.

Далі йде кілометраж на день, якщо він 0, тоді береться стандартні 250 кілометрів на день. Потім йде доплата за лишні кілометри, міні мальний термін оренди, доставка, номер, колір, паливо, рік, кількість сидінь. Також йдуть необов’язкові атрибути, які беруться з сутності моделі, якщо в авто вони не заповнені. Потім кінці треба вибрати місто, та є можливість за допомогою Google Maps API вибирати координати, де знаходиться авто. І в кінці можна вибрати завантажити, або замінити фото автомобіля

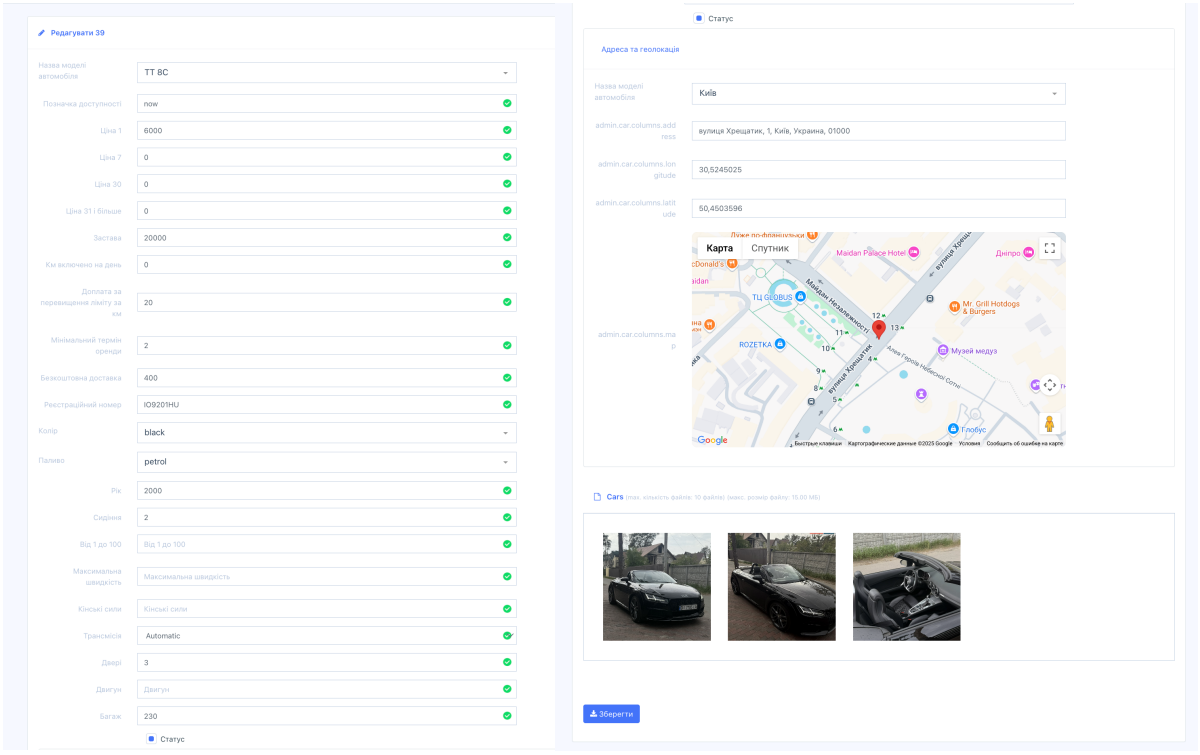


Рис. 3.12 Редагування автомобіля

3.3. Тестування роботи системи

Тестування веб-застосунки "Система управління орендою автомобілів" проведено за допомогою типового сценарію використання, який охоплює дії користувача від входу на сайт до взаємодії з пропозицією оренди. Окремо

перевірено адміністративні функції для оцінки повноти реалізації системи. Основна мета полягала в перевірці правильності роботи ключових можливостей, зручності інтерфейсу та відповідності встановленим вимогам.

Користувач заходить на головну сторінку(рис. 3.13). Там відображаються автомобілі, бренди, та типи кузовів.

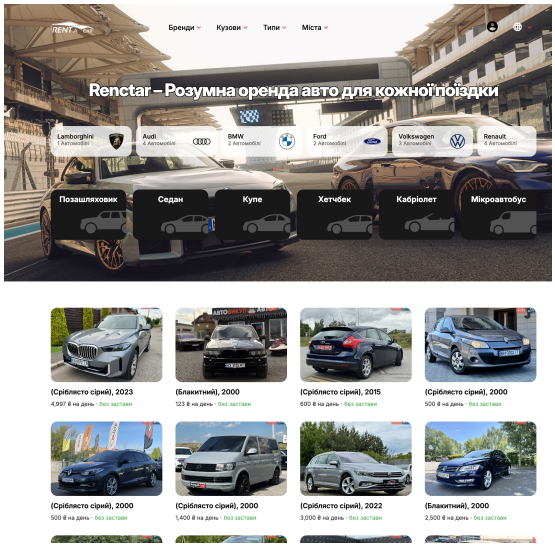


Рис. 3.13 Головна сторінка

Користувач логіниться та переходить на сторінку лістигу автомобілів Audi, і натискає на кнопку орендувати автомобіль(рис. 3.14).

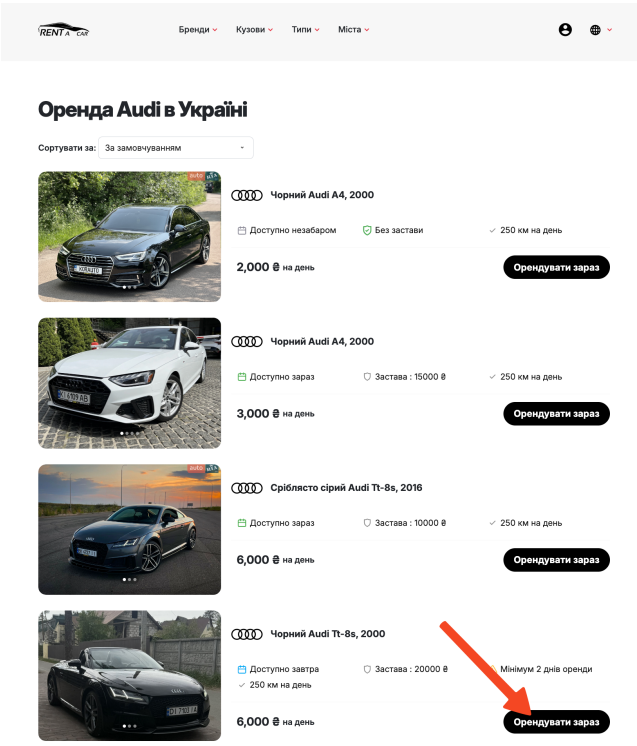


Рис. 3.14 Лістинг автомобілів Audi

Далі користувача перекидує на сторінку оренди автомобіля, де він може обрати вільні дати для оренди(рис. 3.15). В даному варіанті можна вибрати мінімум 2 дні, також йде застава 20,000 грн і доставка авто 400 грн(якщо вказати адресу). Тому обираємо поточні параметри і отримуємо ціну: 32,400 грн.

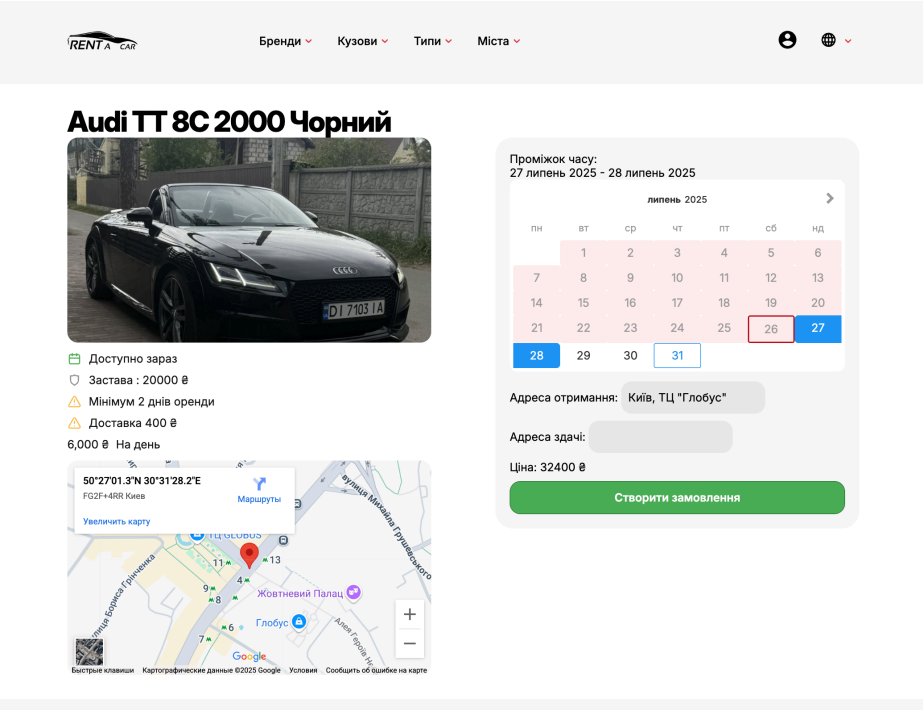


Рис. 3.15 Сторінка створення оренди авто

Коли користувач підтверджує замовлення, його перекидує на сторінку замовлення(рис. 3.16), де в нього є 10 хв, щоб оплатити його.

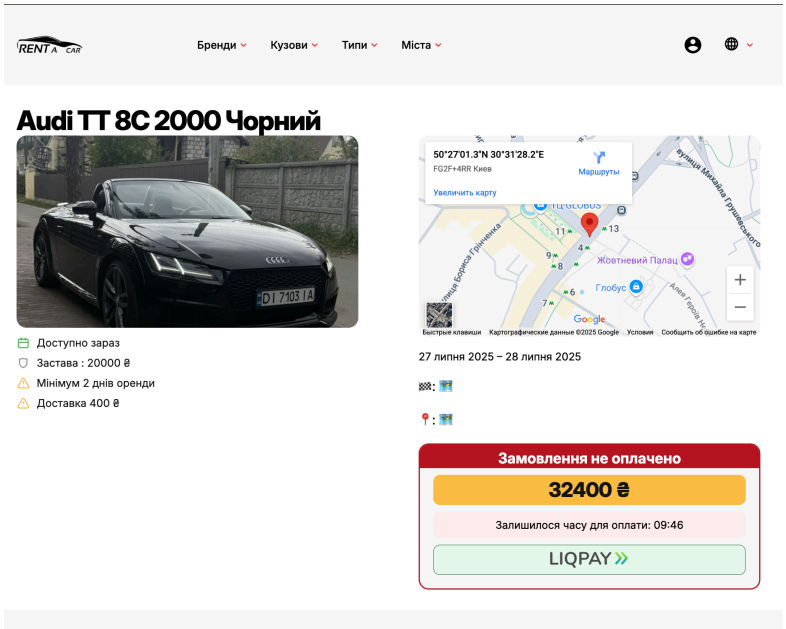


Рис. 3.16 Сторінка замовлення

Потім користувач натискає на кнопку LiqPay, після чого його перекидує на сторінку оплати, де він успішно оплачує замовлення(рис. 3.17).

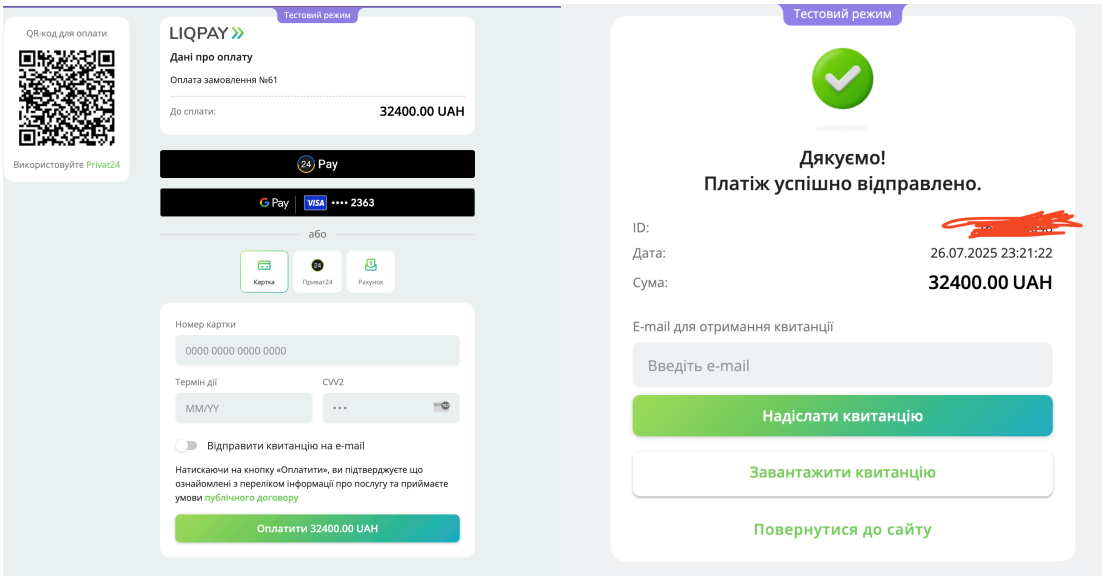


Рис. 3.17 Оплата LiqPay

Після оплати користувач переходить в свій профіль, де може побачити нове замовлення в своєму списку. (рис. 3.18).

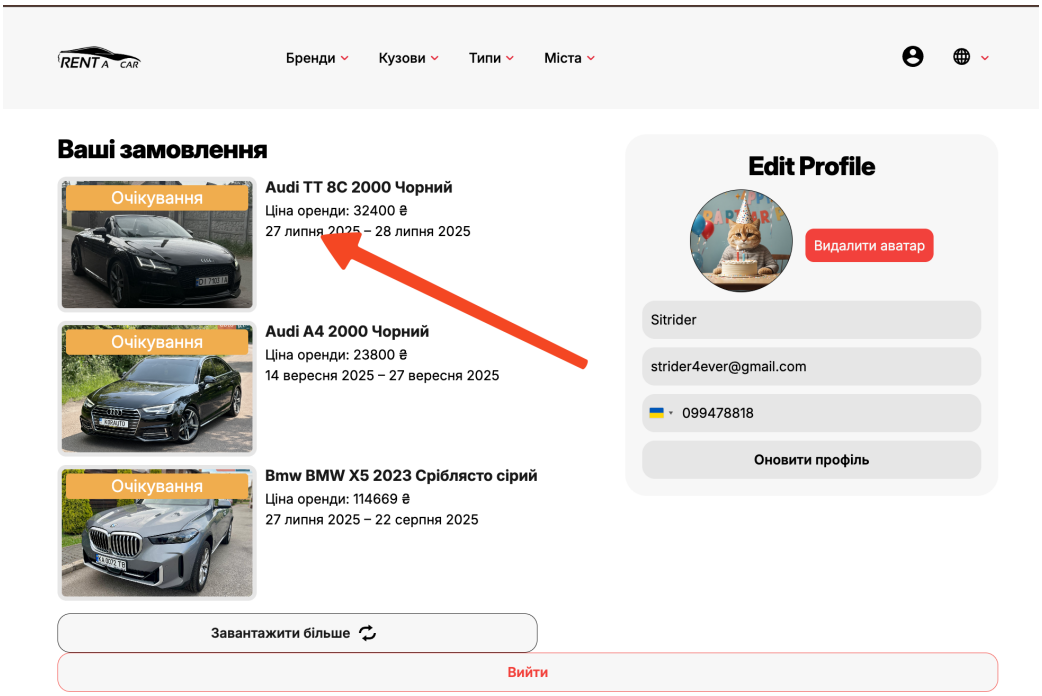


Рис. 3.18 Профіль користувача з новим замовленням

Далі користувач натискає на дане замовлення і його перекидає на сторінку замовлення, де можна подивитися детальну інформацію (рис. 3.19).

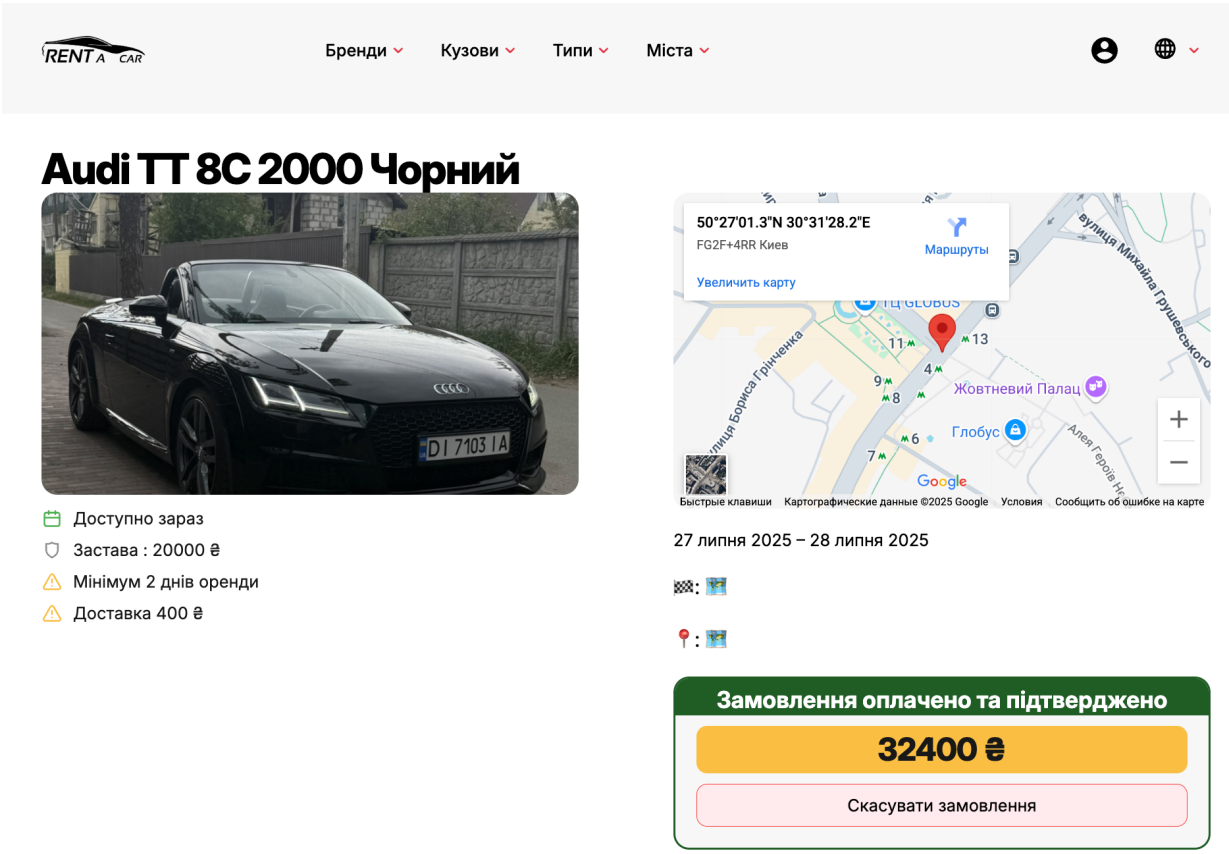


Рис. 3.19 Сторінка нового замовлення

Також слід переконатись, чи додались зміни в адмінці і чи оновилась інформація по замовлення. Тому під адміном заходимо на головну сторінку, де можемо побачити оновлену дошку з новим замовленням(рис. 3.20).

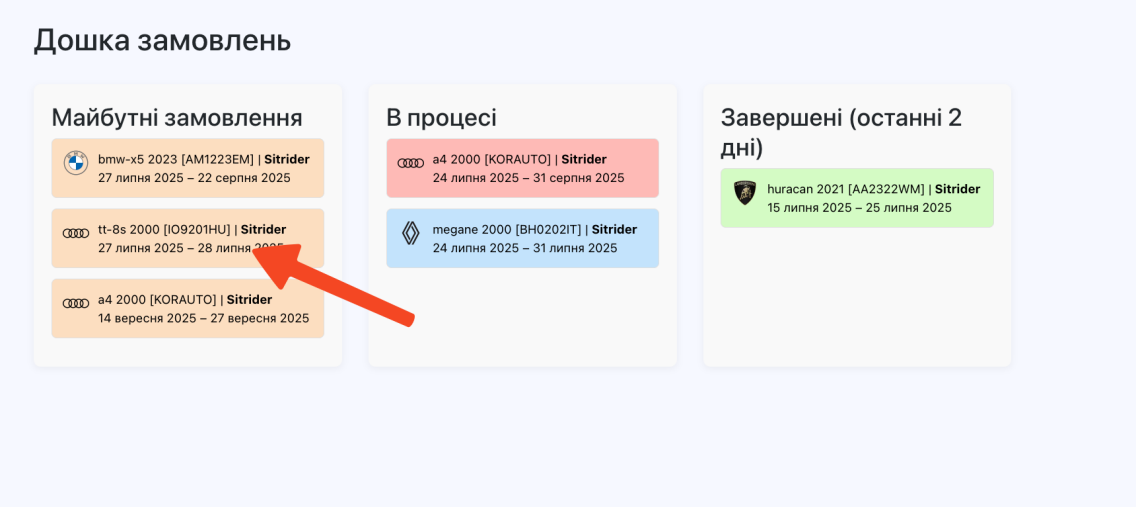


Рис. 3.20 Оновлена дошка замовлень в адмінці

Натискаємо на це замовлення і переходимо на її детальну інформацію (рис. 3.21).

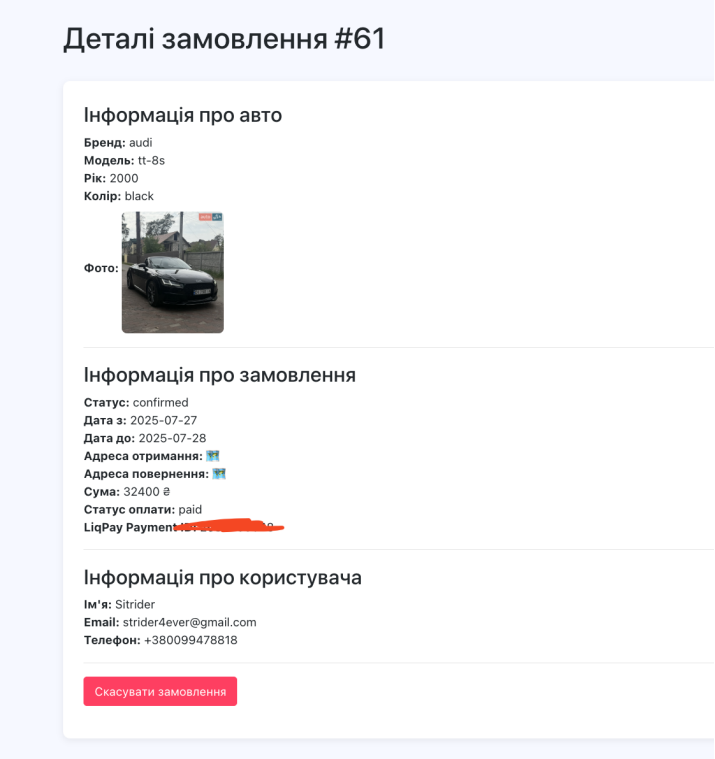


Рис. 3.21 Детальна інформація про замовлення

Висновки до третього розділу

У цьому розділі викладено детальну схему впровадження системи "Система управління орендою автомобілів", яка базується на веб-сервері з використанням Laravel, базі даних MySQL, клієнтській частині, створеній із застосуванням Craftable, SaaS і jQuery, а також адмінпанелі, розробленій через Craftable SaaS. Подано опис процесу встановлення та конфігурації складових, а також їхньої взаємодії, що дало змогу розібратися в структурі системи та зв’язках між її модулями. Налаштування доступу за допомогою middleware Laravel і захист через CSRF-токени та HTTPS підкреслили високу надійність і безпеку системи.

Далі представлено аналіз інтерфейсу розробленого програмного забезпечення, де розкрито функціональні можливості залежно від ролі користувача. Звичайні користувачі мають змогу публікувати, оновлювати пропозиції оренди та відправляти запити, верифіковані учасники отримують можливість переглядати контактні дані напряму, а адміністратори володіють повним набором прав, зокрема управлінням категоріями, внесенням змін і

видаленням вмісту. Цей підхід сприяє зручній навігації для кожного користувача та адаптації системи до їхніх завдань.

Окрім того, продемонстровано результати тестування системи за допомогою сценаріїв використання – від входу на стартову сторінку до взаємодії з пропозиціями, а також адміністрування та реагування на звернення. Перевірка коректності основних операцій (реєстрація, пошук, сортування, спілкування) засвідчила, що система відповідає поставленим вимогам і готова до реального застосування.

ВИСНОВКИ

У межах кваліфікаційної роботи бакалавра розроблено веб-застосунок "Система управління орендою автомобілів". Проведено аналіз подібних сервісів, таких як K-Rent, Renty та Ukr-Prokat, що дозволило сформувати основні вимоги до системи: інтуїтивно зрозумілий дизайн, верифікація користувачів, захищений обмін інформацією та підтримка послуг оренди автомобілів.

Створено об’єктно-орієнтовану структуру системи та реляційну базу даних на MySQL для зберігання даних про користувачів, пропозиції оренди, запити та категорії. Реалізовано механізми для створення замовлень, обробки запитів та адміністрування. Контактні дані захищено для неавторизованих користувачів. Система розгорнута з використанням Laravel, MySQL, Craftable SaaS і jQuery, із забезпеченням безпеки через CSRF-токени та middleware. Інтерфейс адаптовано для різних ролей: звичайні користувачі можуть публікувати пропозиції, верифіковані учасники отримують доступ до контактів, а адміністратори керують усіма аспектами. Тестування засвідчило стабільну роботу функцій: реєстрація, пошук та взаємодія.

Архітектура, побудована на Laravel і MySQL, забезпечує гнучкість і безпеку. Ергономічний інтерфейс сприяє зручному використанню. Практична цінність полягає в наданні ефективного інструменту для оренди автомобілів. Майбутні покращення включають інтеграцію аналітики для адміністраторів, розширення можливостей сортування автомобілів за обраними параметрами та впровадження системи рейтингу користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційний сайт “K-rent” [Електронний ресурс] – Режим доступу до ресурсу: https://k-rent.com.ua/

2. Офіційний сайт “Renty” [Електронний ресурс] – Режим доступу до ресурсу: https://renty.ae/

3. Офіційний сайт “Ukr-procat” [Електронний ресурс] – Режим доступу до ресурсу: https://ukr-prokat.com/

4. Документація Craftable [Електронний ресурс] – Режим доступу до ресурсу: https://getcraftable.com/

5. Документація Laravel [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/12.x

6. Документація PHP [Електронний ресурс] – Режим доступу до ресурсу: https://www.php.net/docs.php

7. Документація SASS [Електронний ресурс] – Режим доступу до ресурсу: https://sass-lang.com/documentation/

8. Документація MySQL [Електронний ресурс] – Режим доступу до ресурсу: https://dev.mysql.com/doc/

9. Документація jQuery [Електронний ресурс] – Режим доступу до ресурсу: https://jquery.com/

10. Документація LiqPay [Електронний ресурс] – Режим доступу до ресурсу: https://www.liqpay.ua/doc

11. Документація Laravel Middleware [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/middleware

12. Документація Google Maps [Електронний ресурс] – Режим доступу до ресурсу: https://developers.google.com/maps/documentation

13. Документація Laravel CSRF Protection [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/csrf

14. Документація Composer [Електронний ресурс] – Режим доступу до ресурсу: https://getcomposer.org/doc/

15. Документація Openserver [Електронний ресурс] – Режим доступу до ресурсу: https://ospanel.io/docs/

16. Документація GitLab [Електронний ресурс] – Режим доступу до ресурсу: https://docs.gitlab.com/

17. Керівництво з розробки веб-додатків на PHP [Електронний ресурс] – Режим доступу до ресурсу: https://phptherightway.com/

18. Посібник з оптимізації баз даних MySQL [Електронний ресурс] – Режим доступу до ресурсу: https://www.percona.com/blog/mysql-optimization/

19. Документація по інтеграції платежів із LiqPay [Електронний ресурс] – Режим доступу до ресурсу: https://liqpay.readme.io/docs

20. Керівництво з безпеки веб-додатків Laravel [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/security

21. Посібник з використання jQuery у веб-розробці [Електронний ресурс] – Режим доступу до ресурсу: https://learn.jquery.com/

22. Документація по адмініструванню серверів із Openserver [Електронний ресурс] – Режим доступу до ресурсу: https://ospanel.io/support/

23. Керівництво з налаштування REST API у Laravel [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/api

24. Матеріали з управління проектами у GitLab [Електронний ресурс] – Режим доступу до ресурсу: https://about.gitlab.com/topics/version-control/

25. Документація по оптимізації продуктивності Laravel [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/performance

26. Керівництво з тестування веб-додатків [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/software_testing/

27. Посібник з інтеграції Craftable із Laravel [Електронний ресурс] – Режим доступу до ресурсу: https://getcraftable.com/docs/integration

28. Документація по безпеці MySQL [Електронний ресурс] – Режим доступу до ресурсу: https://dev.mysql.com/doc/refman/8.0/en/security.html

29. Керівництво з роботи з Composer для PHP-проектів [Електронний ресурс] – Режим доступу до ресурсу: https://getcomposer.org/doc/usage

30. Документація по створенню адаптивного дизайну із SASS [Електронний ресурс] – Режим доступу до ресурсу: https://sass-lang.com/guide

ДОДАТКИ

Додаток А

Контролер Автомобіля

```
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Http\Requests\Admin\Car\BulkDestroyCar;
use App\Http\Requests\Admin\Car\DestroyCar;
use App\Http\Requests\Admin\Car\IndexCar;
use App\Http\Requests\Admin\Car\StoreCar;
use App\Http\Requests\Admin\Car\UpdateCar;
use App\Models\Car;
use App\Models\CarModel;
use App\Models\CarsColor;
use App\Models\City;
use App\Models\Fuel;
use Brackets\AdminListing\Facades\AdminListing;
use Exception;
use Illuminate\Auth\Access\AuthorizationException;
use Illuminate\Contracts\Routing\ResponseFactory;
use Illuminate\Contracts\View\Factory;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Response;
use Illuminate\Routing\Redirector;
use Illuminate\Support\Facades\DB;
use Illuminate\View\View;
use Illuminate\Pagination\LengthAwarePaginator;

class CarsController extends Controller
{

    public function index(IndexCar $request)
    {
        // Якщо не задано сортування, встановлюємо за замовчуванням -id
        if (!$request->has('orderBy') && !$request->has('orderDirection')) {
            $request->merge([
                'orderBy' => 'id',
                'orderDirection' => 'desc',
            ]);
        }
        // create and AdminListing instance for a specific model and
        $data = AdminListing::create(Car::class)->processRequestAndGet(
            // pass the request with params
            $request,

            // set columns to query
            ['id', 'car_model_id', 'car_slug','availability_label', 'price_1', 'price_7', 'price_30',
'price_31_more', 'deposit', 'km_included_per_day', 'overlimit_charge_per_km',
'min_day_reservation', 'free_delivery', 'registration_number', 'color_id', 'fuel_id', 'attribute_year',
'attribute_seats', 'attribute_1_to_100', 'attribute_max_speed', 'attribute_horsepower',
```

```
'attribute_transmission', 'attribute_doors', 'attribute_engine', 'attribute_baggage', 'status'],

                // set columns to searchIn
                ['id',  'availability_label',  'car_slug',  'registration_number',  'attribute_transmission',
'attribute_engine']
            );

            $new_data = Car::carsInfo($data->toArray());

            $old_order = array_column($data->items(), 'id');
            $new_data = $new_data->sortBy(function ($item) use ($old_order) {
                return array_search($item->id, $old_order);
            })->values();

            $data = new LengthAwarePaginator(
                $new_data,
                $data->total(),
                $data->perPage(),
                $data->currentPage(),
                ['path' => $data->path()]
            );
            if ($request->ajax()) {
                if ($request->has('bulk')) {
                    return [
                        'bulkItems' => $data->pluck('id')
                    ];
                }
                return ['data' => $data];
            }

            return view('admin.car.index', ['data' => $data]);
        }

        public function create()
        {
            $this->authorize('admin.car.create');
            return view('admin.car.create',[
                'car_models' => CarModel::all(),
                'cities' => City::all(),
                'cars_colors' => CarsColor::all(),
                'fuels' => Fuel::all(),
                'mode' => 'create'
            ]);
        }

        public function store(StoreCar $request)
        {
            // Sanitize input
            $sanitized = $request->getSanitized();
            $sanitized['car_brand_id'] = $request['car_model']['brand_id'];
            $sanitized['car_body_type_id'] = $request['car_model']['body_type_id'];
            $sanitized['car_model_id'] = $request->getCarModelId();
```

```
            $sanitized['car_slug'] = $request['car_model']['slug'];
            $sanitized['color_id'] = $request->getCarsColorId();
            $sanitized['fuel_id'] = $request->getFuelId();
            $sanitized['city_id'] = $request->getCityID();

            // Store the Car
            $car = Car::create($sanitized);
        if ($request->ajax()) {
                return ['redirect' => url('admin/cars'), 'message' => trans('brackets/admin-
ui::admin.operation.succeeded')];
            }

            return redirect('admin/cars');
        }

        public function show(Car $car)
        {
            $this->authorize('admin.car.show', $car);

            // TODO your code goes here
        }

        public function edit(Car $car)
        {
            $this->authorize('admin.car.edit', $car);
            $car->load('carModel');
            $car->load('color');
            $car->load('fuel');
            $car->load('city');
            return view('admin.car.edit', [
                'car' => $car,
                'car_models' => CarModel::all(),
                'cities' => City::all(),
                'cars_colors' => CarsColor::all(),
                'fuels' => Fuel::all(),
                'mode' => 'edit'
            ]);
        }

        public function update(UpdateCar $request, Car $car)
        {
            // Sanitize input
            $sanitized = $request->getSanitized();

            $sanitized['car_model_id'] = $request->getCarModelId();
            $sanitized['car_brand_id'] = $request['car_model']['brand_id'];
            $sanitized['car_body_type_id'] = $request['car_model']['body_type_id'];
            $sanitized['car_slug'] = $request['car_model']['slug'];
            $sanitized['color_id'] = $request->getCarsColorId();
            $sanitized['fuel_id'] = $request->getFuelId();
            $sanitized['city_id'] = $request->getCityID();
```

```
            // Update changed values Car
            $car->update($sanitized);

            if ($request->ajax()) {
                return [
                    'redirect' => url()->previous(),
                    'message' => trans('brackets/admin-ui::admin.operation.succeeded'),
                ];
            }

            return redirect()->back();
        }

        public function destroy(DestroyCar $request, Car $car)
        {
            $car->delete();

            if ($request->ajax()) {
                return response(['message' => trans('brackets/admin-
ui::admin.operation.succeeded')]);
            }

            return redirect()->back();
        }

        public function bulkDestroy(BulkDestroyCar $request) : Response
        {
            DB::transaction(static function () use ($request) {
                collect($request->data['ids'])
                    ->chunk(1000)
                    ->each(static function ($bulkChunk) {
                        Car::whereIn('id', $bulkChunk)->delete();
                    });
            });

            return response(['message' => trans('brackets/admin-ui::admin.operation.succeeded')]);
        }
```

Додаток Б

Контролер замовлень

```
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\Order;
use Illuminate\Http\Request;
use Brackets\AdminListing\AdminListing;

class OrdersController extends Controller
{
    public function index(Request $request)
    {
        // Default sorting
        if (!$request->has('orderBy') && !$request->has('orderDirection')) {
            $request->merge([
                'orderBy' => 'id',
                'orderDirection' => 'desc',
            ]);
        }
        // AdminListing for Order
        $data = AdminListing::create(Order::class)->processRequestAndGet(
            $request,
            ['id', 'user_id', 'car_id', 'status', 'total_price',
'currency_id','date_from','date_to','created_at'],
            ['id', 'status']
        );
        // Eager load relations for display
        $data->load(['user', 'car.media', 'currency']);
        // Assign carInfo to each car in orders
        foreach ($data as $order) {
            if ($order->car) {
                $order->car_info = $order->car->carInfo();
            }
        }
        if ($request->ajax()) {
            if ($request->has('bulk')) {
                return [
                    'bulkItems' => $data->pluck('id')
                ];
            }

            return response()->json(['data' => $data]);
        }
//          dd($data[0]);
        return view('admin.orders.index', compact('data'));
    }
    /**
     * Show board with orders split by status: upcoming, in progress, finished (not older than 2
```

```
days)
     */
    public function board(Request $request)
    {
        $now = now();
        $twoDaysAgo = $now->copy()->subDays(2)->startOfDay();
        // Upcoming: date_from > now
        $upcoming = \App\Models\Order::with(['car', 'user', 'currency'])
            ->where('date_from', '>', $now)
            ->whereNotNull('liqpay_payment_id')
            ->orderBy('date_from')
            ->get();
        // In progress: date_from <= now && date_to >= now
        $inProgress = \App\Models\Order::with(['car', 'user', 'currency'])
            ->where('date_from', '<=', $now)
            ->where('date_to', '>=', $now)
            ->whereNotNull('liqpay_payment_id')
            ->orderBy('date_from')
            ->get();
        // Finished: date_to < now && date_to >= twoDaysAgo
        $finished = \App\Models\Order::with(['car', 'user', 'currency'])
            ->where('date_to', '<', $now)
            ->where('date_to', '>=', $twoDaysAgo)
            ->whereNotNull('liqpay_payment_id')
            ->orderBy('date_to', 'desc')
            ->get();
        // Assign carInfo to each car in orders
        foreach([$upcoming, $inProgress, $finished] as $orders) {
            foreach ($orders as $order) {
                if ($order->car) {
                    $order->car_info = $order->car->carInfo();
                }
            }
        }
        return view('admin.orders.board', compact('upcoming', 'inProgress', 'finished'));
    }
    /**
     * Show a single order with all details and allow admin to cancel.
     */
    public function show(Request $request, $id)
    {
        $order = \App\Models\Order::with(['car', 'user', 'currency'])->findOrFail($id);
        if ($order->car) {
            $order->car_info = $order->car->carInfo();
        }
        return view('admin.orders.show', compact('order'));
    }

    /**
     * Cancel an order as admin.
     */
    public function cancel(Request $request, $id)
```

```
        {
            $order = \App\Models\Order::findOrFail($id);
            if ($order->status === \App\Enums\OrderType::PAYMENT_CANCELLED->value) {
                return redirect()->back()->withErrors(['cancel' => 'Order already cancelled.']);
            }
            $order->status = \App\Enums\OrderType::PAYMENT_CANCELLED->value;
            $order->save();
            return redirect()->route('admin/orders/show', $order->id)->with('success', 'Order cancelled successfully.');
        }
    }
```

Додаток В

Налаштування роутів для користувачів

```
if (Schema::hasTable('languages')) {
        Route::middleware(['locale'])->group(function () {
            Route::prefix(LocaleMiddleware::getLocale())
                ->group(function () {
                    Route::namespace('App\Http\Controllers\Front')->group(function () {
                        Route::get('/', 'SiteController@index')->name('home');
                        Route::get('/' . RouteEnum::BODIES->value, 'SiteController@bodyTypes')-
>name('body-types');
                        Route::get('/' . RouteEnum::BODIES->value . '/{slug}',
'SiteController@bodyType')->name('body-type');
                        Route::get('/' . RouteEnum::TYPES->value, 'SiteController@types')-
>name('types');
                        Route::get('/' . RouteEnum::TYPES->value . '/{slug}', 'SiteController@type')-
>name('type');
                        Route::get('/' . RouteEnum::BRANDS->value, 'SiteController@brands')-
>name('brands');
                        Route::get('/' . RouteEnum::BRANDS->value . '/{slug}',
'SiteController@brand')->name('brand');
                        Route::get('/' . RouteEnum::CITIES->value . '/{slug}', 'SiteController@city')-
>name('city');
                        Route::get('/' . RouteEnum::CAR->value . '/{id}', 'SiteController@car')-
>name('car');
                        Route::get('/login', 'AuthController@showLoginForm')->name('login');
                        Route::post('/login', 'AuthController@login')->name('login.post');
                        Route::get('/register', 'AuthController@showRegistrationForm')-
>name('register');
                        Route::post('/register', 'AuthController@register')->name('front.register.post');
                        Route::post('/logout', 'AuthController@logout')->name('front.logout');

                        Route::middleware('auth')->group(function () {
                            Route::get('/profile', 'ProfileController@editProfile')->name('front.profile');
                            Route::post('/profile', 'ProfileController@updateProfile')-
>name('front.profile.update');
                            Route::get('/profile/password', 'ProfileController@editPassword')-
>name('front.profile.password');
                            Route::post('/profile/password', 'ProfileController@updatePassword')-
>name('front.profile.password.update');

                            Route::get('/create-order/{car}', [OrderController::class, 'createOrder'])-
>name('orders.create');
                            Route::post('/create-order/{car}', [OrderController::class, 'store'])-
>name('orders.create.post');

                            Route::prefix('orders')->group(function () {

                                Route::get('{order}', [OrderController::class, 'show'])-
>name('orders.show');
```

```
                    Route::get('{order}/pay', [OrderController::class, 'showPaymentForm'])->name('orders.pay');
                    Route::post('{order}/liqpay', [LiqPayController::class, 'generateForm'])->name('liqpay.form');
                    // Cancel order route
                    Route::post('/orders/{order}/cancel', [OrderController::class, 'cancel'])->name('order.cancel');
                });
            });

            Route::post('/set-currency', [CurrencyController::class, 'set'])->name('set.currency');
        });
    });
});
}
```