

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

КВАЛІФІКАЦІЙНА РОБОТА
на тему: «Розробка web-застосунку «DriveRent» для сервісу
оренди авто»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

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

(підпис) Дар'я ПОЛЯНСЬКА

Виконала: здобувачка вищої освіти групи ПД-42

Дар'я ПОЛЯНСЬКА

Керівник: _____
Наталія ТРІНТІНА
канд.тех.наук., доц.

Рецензент: _____

Київ 2026

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**
Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Інженерії програмного забезпечення

_____ Ірина ЗАМРІЙ

« ____ » _____ 2026 р.

**ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Польянській Дар'ї Сергіївні

1. Тема кваліфікаційної роботи: «Розробка web-застосунку «DriveRent» для сервісу оренди авто»

керівник кваліфікаційної роботи канд.тех.наук, доц. Наталія ТРИНТИНА,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» лютого 2026 року № 51.

2. Строк подання кваліфікаційної роботи «29» травня 2026 року.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про організацію процесів оренди автомобілів, методи автоматизації бронювання транспортних засобів, опис підходів до керування автопарком, тарифами та статусами замовлень, технічна документація щодо використання технологій розробки.

4. Зміст розрахунково-пояснювальної записки:

1. Огляд та аналіз предметної галузі оренди автомобілів і процесів керування автопарком.
2. Аналіз існуючих програмних засобів для автоматизації оренди автомобілів, бронювання транспортних засобів і контролю замовлень.
3. Огляд засобів реалізації web-застосунку DriveRent для оренди автомобілів.
4. Проектування структури web-застосунку, бази даних та основних модулів системи.
5. Програмна реалізація та опис функціонування web-застосунку DriveRent.
6. Тестування web-застосунку.

5. Вимоги до застосунку:

1. Аналіз предметної галузі оренди автомобілів.
2. Аналіз аналогів програмного забезпечення.
3. Вимоги до web-застосунку DriveRent.
4. Архітектура програмної реалізації застосунку.
5. Взаємодія React, Spring Boot та PostgreSQL.
6. Алгоритм бронювання автомобіля.
7. Діаграма класів або структура основних сутностей системи.
8. Екранні форми web-застосунку.
9. Апробація результатів дослідження.

6. Дата видачі завдання «23» лютого 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	23.02–28.02.2026	
2	Аналіз та дослідження існуючих аналогів програмного забезпечення для оренди автомобілів	01.03–07.03.2026	
3	Аналіз предметної галузі оренди автомобілів та визначення вимог до web-застосунку DriveRent	09.03–15.03.2026	
4	Огляд засобів реалізації web-застосунку DriveRent	16.03–22.03.2026	
5	Проектування web-застосунку DriveRent, логічної структури системи та UML-діаграм	23.03–05.04.2026	
6	Програмна реалізація серверної та клієнтської частин web-застосунку	06.04–26.04.2026	
7	Тестування web-застосунку DriveRent та оформлення результатів тестування	27.04–03.05.2026	
8	Оформлення роботи: вступ, висновки, реферат, список джерел та додатки	04.05–10.05.2026	
9	Розробка демонстраційних матеріалів	11.05–17.05.2026	
10	Попередній захист роботи	18.05–22.05.2026	

Здобувачка вищої освіти

_____ (підпис)

Дар'я ПОЛЯНСЬКА

Керівник
кваліфікаційної роботи

_____ (підпис)

Наталія ТРИНТИНА

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 50 стор., 3 табл., 31 рис., 22 джерел.

Мета роботи – спрощення процесу оренди автомобілів шляхом розробки web-застосунку DriveRent, що забезпечує перегляд каталогу авто, оформлення оренди та керування автопарком.

Об'єкт дослідження – процес організації оренди автомобілів, що включає вибір транспортного засобу, оформлення оренди, облік клієнтів і контроль стану автопарку.

Предмет дослідження – засоби, методи та технології автоматизації процесу оренди автомобілів у web-застосунку DriveRent.

Короткий зміст роботи: У роботі проаналізовано предметну галузь оренди автомобілів та визначено основні проблеми. Проаналізовано існуючі програмні засоби для автоматизації оренди автомобілів: RentSyst, HQ Rental Software, Rent Centric та Turo.

Розроблено структуру web-застосунку DriveRent та визначено його ключові функціональні можливості, зокрема: перегляд каталогу автомобілів, фільтрацію транспортних засобів за категорією, характеристиками та доступністю, перегляд детальної інформації про автомобіль, створення бронювання, перевірку доступності автомобіля на обраний період, керування тарифами, статусами замовлень та автопарком.

У роботі використано Java та Spring Boot для реалізації серверної частини, PostgreSQL для збереження даних, React і TypeScript для створення клієнтського інтерфейсу, а також Docker для організації середовища розгортання.

Сферою використання застосунку є автоматизація роботи сервісів оренди автомобілів, які потребують зручного інструменту для керування автопарком, бронюваннями, тарифами та замовленнями клієнтів.

КЛЮЧОВІ СЛОВА: ОRENDA АВТОМОБІЛІВ, WEB-ЗАСТОСУНОК, DRIVERENT, БРОНЮВАННЯ, АВТОПАРК, JAVA, SPRING BOOT, POSTGRESQL, REACT, TYPESCRIPT.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	9
ВСТУП	11
1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ	12
1.1 Особливості процесу оренди автомобілів	12
1.2 Визначення вимог до веб-застосунку DriveRent	15
1.3 Інформаційна модель предметної галузі	16
2 АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ ОРЕНДИ АВТОМОБІЛІВ	17
2.1 RENTSYST	17
2.2 HQ RENTAL SOFTWARE	19
2.3 RENT CENTRIC	20
2.4 TURO	21
2.5 Узагальнення результатів аналізу	22
3 ОГЛЯД ЗАСОБІВ РЕАЛІЗАЦІЇ ДОДАТКУ ДЛЯ ОРЕНДИ АВТОМОБІЛІВ	25
3.1 Мова програмування Java	26
3.2 Фреймворк Spring Boot	26
3.3 Система управління базами даних PostgreSQL	26
3.4 React	27
3.5 TypeScript	28
3.6 Docker	28
3.7 Середовище розробки — IntelliJ IDEA	28
3.8 Система контролю версій — Git та GitHub	29
4 ПРОЄКТУВАННЯ WEB-ЗАСТОСУНКУ DRIVERENT	31
4.1 Логічна структура системи	31
4.2 Діаграма прецедентів веб-застосунку DriveRent	32
4.3 Діаграма предметної галузі веб-застосунку DriveRent	33
4.4 Діаграми діяльності	34
4.5 Діаграма класів веб-застосунку DriveRent	36
4.6 Схема компонентів веб-застосунку DriveRent	38
4.7 Схема розгортання веб-застосунку DriveRent	39
4.8 Діаграма станів меню веб-застосунку DriveRent	41
5 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ WEB-ЗАСТОСУНКУ DRIVERENT	42
5.1 Реалізація серверної частини	43
5.2 Реалізація клієнтської частини	45
5.3 Екранні форми веб-застосунку	47
5.4 Тестування програмного забезпечення	52
5.5 Результати тестування	54
ВИСНОВКИ	58
ПЕРЕЛІК ПОСИЛАНЬ	60
ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ	62
ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ	71

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення

БД – база даних

СУБД – система управління базами даних

API – Application Programming Interface

REST – Representational State Transfer

REST API – програмний інтерфейс взаємодії між клієнтською та серверною частиною за принципами REST

SPA – Single Page Application

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

JS – JavaScript

TS – TypeScript

JSON – JavaScript Object Notation

SQL – Structured Query Language

ORM – Object-Relational Mapping

JPA – Java Persistence API

JWT – JSON Web Token

HTTP – HyperText Transfer Protocol

HTTPS – HyperText Transfer Protocol Secure

UI – User Interface

UX – User Experience

IDE – Integrated Development Environment

UML – Unified Modeling Language

CRUD – Create, Read, Update, Delete

DTO – Data Transfer Object

MVC – Model-View-Controller

ВСТУП

Актуальність теми. Оренда автомобілів відіграє дуже важливу роль в житті багатьох людей. Багато фізичних осіб, підприємств, туристів, доставок та інших діючих осіб використовують транспортні засоби без придбання їх у власність. Сучасні сервіси з отриманням автомобілів для оренди вже досить часто створюються у вебсередовищах.

У багатьох вебсередовищах оренди автомобілів автоматизація процесів ведення обліку автопарку, тарифів, бронювань та замовлень ведеться досить розрізненими інструментами. Ці спрощені інструменти спричиняють труднощі у веденні справ, що стосуються автопарку, оренди автомобілів, тарифів та замовлень. Такий підхід все більше викликає проблему із збільшенням кількості автомобілів, власників та замовлень.

Вебзастосунок для оренди автомобілів може вирішити вказану проблему. Клієнт отримує можливості переглядати автомобілі, їх вибрати та зробити бронювання. Адміністратор отримує можливості керувати автопарком, встановлювати тарифи, встановлювати статуси автомобілів та замовлення від клієнтів. Враховуючи наведені думки, можна вважати, що розробка web-застосунку DriveRent для оренди автомобілів є актуальною задачею.

Мета роботи - спрощення процесу оренди автомобілів шляхом розробки web-застосунку DriveRent, що забезпечує перегляд каталогу авто, оформлення оренди та керування автопарком.

Відповідно до вказаної мети необхідно виконати такі завдання для досягнення мети бакалаврської роботи: дослідження предметної галузі та встановлення її проблем, аналіз вже існуючих вебзастосунків, встановлення вимог до вебзастосунку, вибору засобів його реалізації, проектування вебзастосунку,

його реалізація на сервері, веб-інтерфейсі, встановлення механізму збереження даних та тестування.

Об'єкт дослідження - процес організації оренди автомобілів, що включає вибір транспортного засобу, оформлення оренди, облік клієнтів і контроль стану автопарку.

Предмет дослідження - засоби, методи та технології автоматизації процесу оренди автомобілів у web-застосунку DriveRent.

Методи дослідження. У бакалаврській роботі використовуються такі методи: аналіз предметної галузі, порівняльний аналіз вже існуючих вебзастосунків, розробка об'єктно-орієнтованого проекту, веб-моделювання, проектування вебзастосунку, тестування.

Наукова новизна одержаних результатів. У ході виконання роботи одержано нове вирішення задачі вебзастосунку для оренди автомобілів - структуру web-застосунку DriveRent.

Практична значущість одержаних результатів. Розроблений вебзастосунок може бути використаний для оренди автомобілів. Застосунок DriveRent можна розширити додаванням можливості створення замовлення з онлайн-оплатою, встановленням рейтингів автомобілів та встановленням системи знижок або встановлення акцій.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Особливості процесу оренди автомобілів

Предметною галуззю, обраною для виконання даної кваліфікаційної роботи, є процес організації та автоматизації оренди автомобілів за допомогою вебзастосунку. Оренда автомобілів є однією з найбільш поширених послуг. Такий вид оренди використовується фізичними особами та підприємствами для короткострокового або довгострокового використання автомобілів, які не вимагають придбання їх у власність. Така модель оренди виправдана для великої кількості клієнтів, включаючи туристів, представників компаній, служб доставки, малого бізнесу та інших осіб, яким іноді випадає потреба в короткостроковому використанні автомобіля.

Клієнти сучасних часів вимагають від процесу оренди автомобіля багато інноваційних рішень, що дозволяють їм виконати весь процес онлайн. Ідеальний вигляд процесу вимагає від користувача можливості перегляду доступних автомобілів, вибору тарифу, вказання періоду оренди та оформлення бронювання без контакту із адміністратором. Сервіс оренди автомобілів має мати можливість контролювати автопарк, статуси автомобілів, їхні тарифи та заявки від клієнтів.

Традиційно процес оренди автомобілів веде адміністратор. Він самостійно веде список автомобілів, приймає заявки від клієнтів та відновлює їх статуси. Такий спосіб дії є незручним і при збільшенні об'ємів автопарку і клієнтів.

Цільовою аудиторією є дві групи людей: клієнти та адміністратори. Клієнтами можна вважати тих, хто бажає орендувати автомобілі, а також тих, хто виконує це як частину виконання інших обов'язків. Адміністраторами можна вважати тих, хто відповідає за додавання автомобілів до відділення, їх опис та перегляд.

У випадку з клієнтами найважливішими для них є зручність перегляду автомобілів, фільтрація, актуальні ціни та мінімальна кількість дій, що

відбуваються до відправки замовлення. Адміністраторам найбільш важливо те, що вся інформація щодо автомобілів є надійною, що вона має логічне написання та швидко оновлюється.

Типовий процес оренди:

1. Адміністратор додає автомобілі до відділення оренди.
2. Клієнт переглядає відділення, вибирає автомобіль та створює замовлення.
3. Адміністратор перевіряє доступності автомобіля.
4. У разі, якщо автомобіль доступний, замовлення підтверджується.
5. Після дії періоду замовлення, статус автомобіля оновлюється, таким чином він може бути вибраним іншими клієнтами.

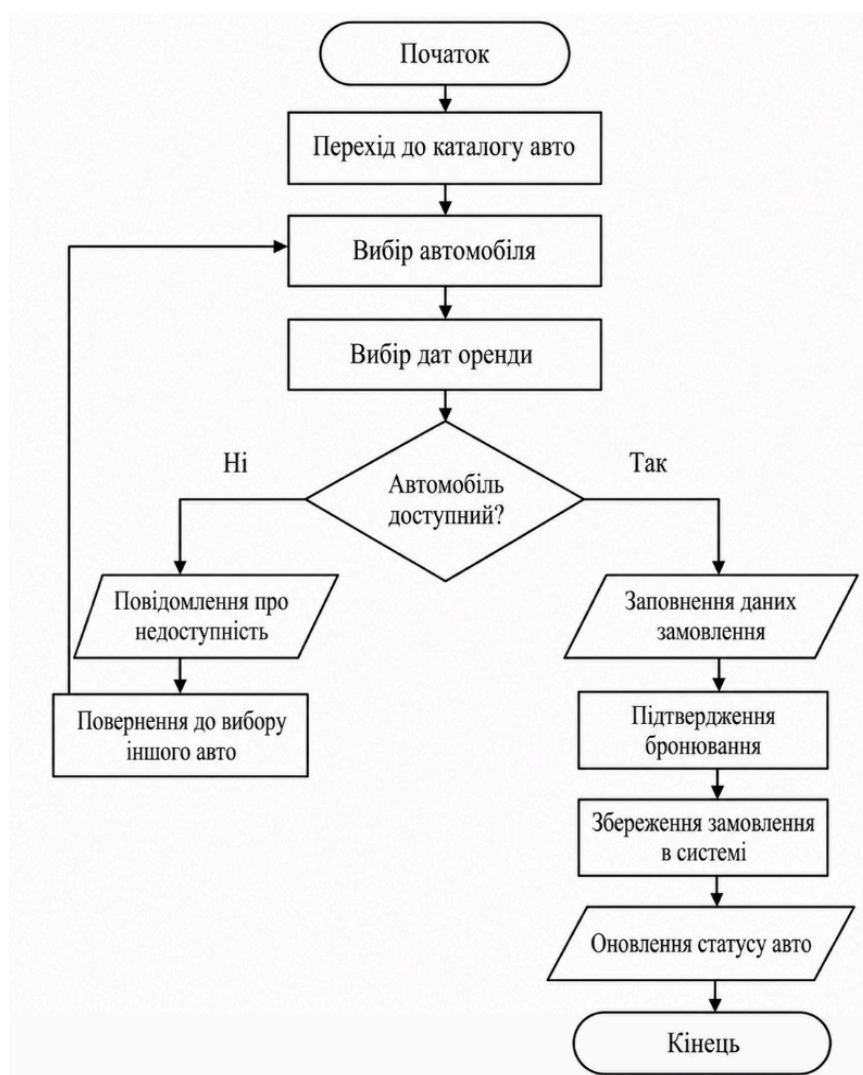


Рис. 1.1 Послідовність дій користувача під час бронювання автомобіля

Однією з важливих проблем у видачі автомобілів на оренду є проблема виключення конфліктів у бронюванні. Один і той самий автомобіль не може бути виданим кільком клієнтам на однаковий період часу. Тому потрібно виконувати перевірку на сервері перед відмовою замовлення на видачу автомобіля. Також до задачі належить розробка тарифної системи, яка забезпечить встановлення вартості оренди автомобіля на відрізок часу. Тарифна система може бути простою у випадку виконання початкової версії вебзастосунків, але майбутній розвиток програмного продукту дозволить передбачити тарифну систему зі знижками тощо.

Вирішення задачі предметної області можна реалізувати за допомогою різних інформаційних засобів. Один з можливих варіантів - вибрати ведення в електронних таблицях. Проте цей засіб забезпечує автоматизації та зручності для клієнтів. Використання CRM програмного забезпечення також можливе. Проте вони можуть не враховувати всі особливості оренди автомобілів. Найкращим варіантом для рішення таких проблем є створення вебзастосунку, спрямованого на різні процеси автопарку.

Особливістю вибраного способу є можливість доступу до вебзастосунку з різних пристроїв. Він буде зручним для клієнтів та адміністраторів. У випадку якщо він стане доступним у смартфоні клієнтів, вони зможуть переглядати автомобілі, адміністратор може оновлювати автопарк у будь-якому місці.

Ручний облік		Автоматизована система Driverent	
	Ведення даних у таблицях		Централізована база даних
	Ручна перевірка доступності авто		Автоматична перевірка доступності
	Високий ризик помилок		Зниження кількості помилок
	Можливе дублювання бронювань		Уникнення конфліктів бронювання
	Повільне оновлення інформації		Актуальні статуси в реальному часі
	Складність контролю замовлень		Зручне керування замовленнями
		Результат: підвищення швидкості, точності та зручності роботи сервісу	

Рис. 1.2 Порівняння традиційного та автоматизованого підходу до керування орендою автомобілів

1.2 Визначення вимог до web-застосунку DriveRent

На основі аналізу вимог до оренди автомобілів було здійснено визначення функціональних та нефункціональних вимог до розробки web-застосунку DriveRent. Функціональні вимоги визначають, які дії повинен виконувати застосунок для забезпечення коректної оренди автомобілів. Вимоги включають наступні функції:

- реєстрація та авторизація користувачів (ролі клієнт, адміністратор);
- перегляд категорій автомобілів;
- фільтрація автомобілів;
- перегляд інформації про вибраний автомобіль;
- створення бронювання;
- перевірка доступності автомобіля;
- зберігання інформації замовників автомобілів;
- перегляд власних бронювань;
- керування відділом автомобілів (додавання, видалення, модифікація);
- керування відділом замовлень (замовлення, видалення, модифікація).

До нефункціональних вимог віднесено:

- час завантаження сторінок не повинен перевищувати 3 секунди;
- API виконує запити в межах 1-2 секунди;
- середовище повинне забезпечувати роботу не менше 50 користувачів;
- інтерфейс повинен правильно відображатися на персональному комп'ютері, планшеті та смартфоні;
- доступ до адмін-панелі повинен бути тільки для користувачів із роллю адміністратора;
- дані повинні зберігатися у базі даних;
- код застосунку повинен мати модульну структуру.

1.3 Інформаційна модель предметної галузі

Інформаційна модель предметної галузі відображає основні сутності, що використовуються в web-застосунку DriveRent, та взаємозв'язки між ними. Відображення цих компонентів дозволяє краще розуміти структуру web-застосунку, взаємодію користувачів із ним, а також відразу встановити напрямок проектування бази даних для компонентів.

В web-застосунку DriveRent відображаються наступні сутності: користувач, автомобіль, категорія автомобіля, тариф, бронювання, замовлення, статус. Користувачі можуть мати одну із двох ролей: клієнт або адміністратор. Клієнт відіграє роль перегляду автомобілів і створення бронювань. Адміністратори відіграють роль керування автомобілями, тарифами, замовленнями.

Сутність „Автомобіль“ містить такі інформаційні елементи, які відображають опис автомобіля: марка, модель, рік випуску, тип кузова, тип пального, передач, кількість місць, вартість оренди, статус. Автомобіль належить до категорії автомобіля, які відділяються наступними типами: економ, комфорт, бізнес та позашляховик.

Сутність „Тариф“ відображає такі параметри, як вартість оренди автомобіля за добу, додаткові умови використання автомобіля, опис тарифу. Тариф відноситься до автомобіля.

Сутність „Бронювання“ містить інформацію про клієнта, автомобіль, дату початку та завершення бронювання, статус бронювання. При створенні бронювання необхідно перевірити доступність автомобіля.

Сутність „Замовлення“ містить інформацію про інформаційні параметри замовлення, що відображає статус замовлення. Використання статусів дозволяє адміністратору відстежувати статус замовлення.

В результаті відображення основних сутностей та їх взаємозв'язків у предметній галузі встановлено основні компоненти web-застосунку, що дозволяє відразу встановити напрямок створення доменної моделі, діаграми класів або структури бази даних для системи.

2 АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ ОРЕНДИ АВТОМОБІЛІВ

Програмне забезпечення для автоматизації оренди автомобілів призначене для спрощення різних процесів, що відносяться до бронювання транспортних засобів, керування автопарком та взаємодії із клієнтами. Вони можуть виконуватися у вигляді web-застосунків, мобільних додатків, CRM-систем або маркетплейсів, які сполучають власників автомобілів зі сприймати їх в оренду.

Програмні продукти для оренди автомобілів можна відсортувати наступним чином:

- спеціалізовані CRM-системи для компаній, що займаються орендою автомобілів;
- вебзастосунки для бронювання транспортних засобів;
- мобільні додатки для каршерингу;
- маркетплейси для оренди автомобілів від користувача до користувача;
- універсальні таблиці та CRM-системи, які можуть бути адаптовані під діагностування автопарку.

В межах наведеного розділу розглядаються існуючі програмні рішення, що вирішують відображені задачі, подібні до потрібної реалізації web-застосунку.

2.1 RentSyst

RentSyst - це система, що взаємодіє з компаніями, які хочуть керувати своїм автопарком. На сайті вказується, що RentSyst дозволяє компаніям контролювати процес оренди, відстежувати завантаження автомобілів та керувати бронюваннями віддалено через смартфон або веб-сайт.

Система виглядає як підхід до рішення задач конкретної галузі. Вона вже містить функції, що стосуються транспортних засобів. Майбутній застосунок

DriveRent може бути використаний як приклад системи, яка відразу вміщає і клієнтські, і адміністративні частини.

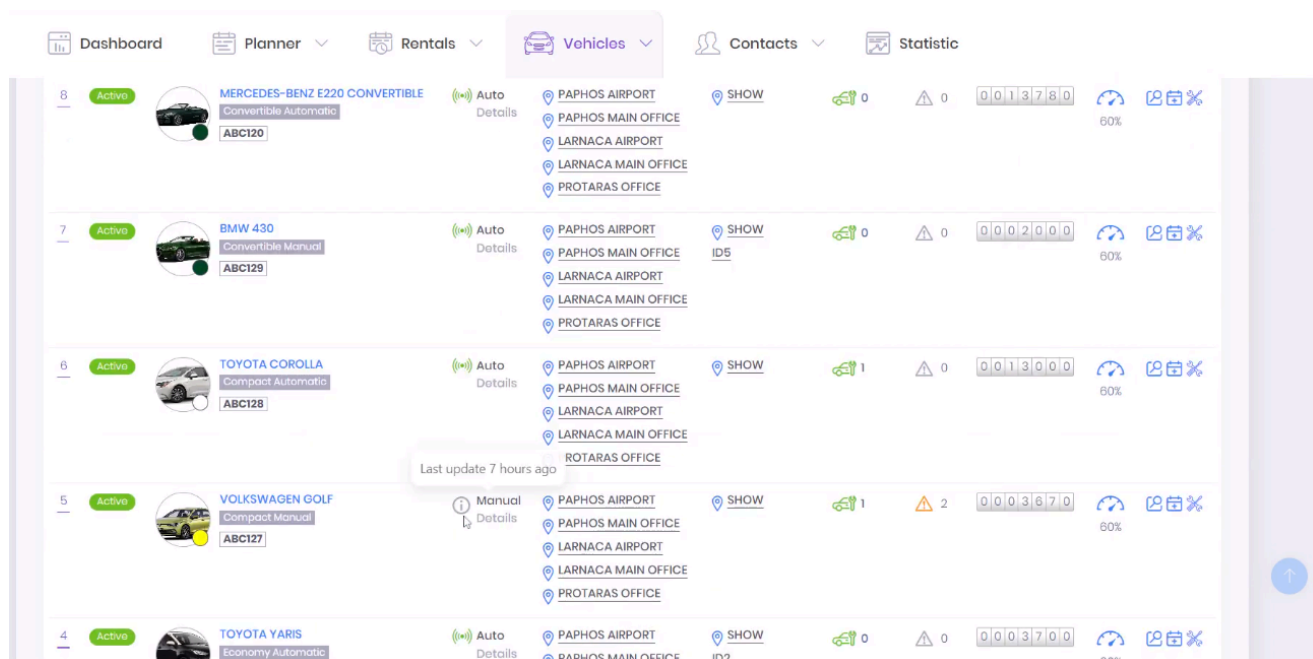


Рис. 2.1 Приклад інтерфейсу системи RentSyst для керування орендою автомобілів

Переваги RentSyst:

- наявність функцій керування автопарком;
- можливість роботи з бронюваннями та клієнтами;
- підтримка web-версії та мобільного доступу;
- можливість контролю платежів і витрат;
- орієнтація саме на сферу оренди транспортних засобів.

Недоліки RentSyst:

- система є комерційною, тому її використання може потребувати регулярної оплати;
- функціонал може бути надлишковим для невеликого сервісу оренди;
- інтерфейс і логіка роботи залежать від готової платформи;
- для повного використання можливостей системи потрібне налаштування під конкретну компанію.

2.2 HQ Rental Software

HQ Rental Software є програмним забезпеченням для компаній, що займаються орендою автомобілів. Software включає наступні функції: керування автопарком, онлайн-бронювання, обробка платежів, робота з клієнтами, створення звітів та відстеження доступності автомобілів. Компанія розглядає цей продукт як комплексне рішення для розвитку бізнесу з оренди автомобілів.

Особливу увагу в HQ Rental Software приділено онлайн-резервуванню та керуванню автопарком. Такі функції вважаються важливими не тільки для клієнтів DriveRent, але і для адміністратора.

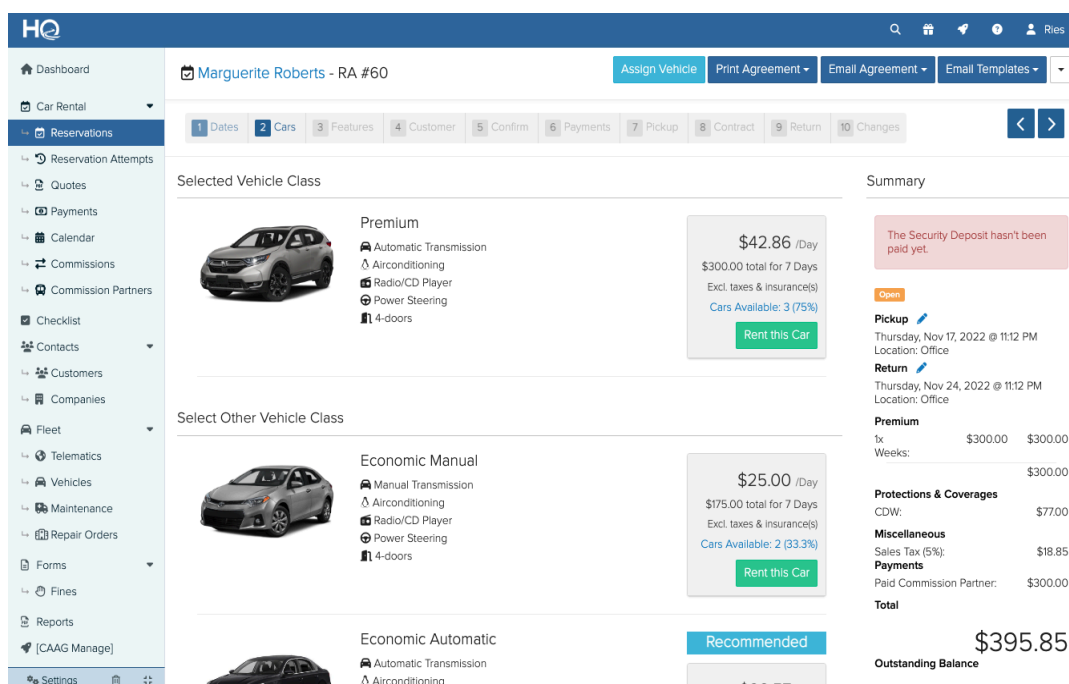


Рис. 2.2 Приклад інтерфейсу HQ Rental Software для керування автопарком і бронюваннями

Переваги HQ Rental Software:

- підтримка онлайн-бронювання;
- наявність функцій керування автопарком;
- можливість контролю клієнтів і замовлень;

- наявність аналітичних панелей для діагностування різних аспектів бізнесу;
- підтримка мобільного доступу.

Недоліки HQ Rental Software:

- визначена специфіка програмного забезпечення для ведення готових бізнесів, а не для невеликих проєктів;
- складність впровадження деяких функцій;
- забезпечення додаткового налаштування для адаптації системи під модель роботи підприємства;
- неможливість системи відповідати індивідуальним вимогам до програмного забезпечення замовника.

2.3 Rent Centric

Rent Centric - це програмне забезпечення для керування прокатом автомобілів. Програма дозволяє керувати парком автомобілів, створювати сайт для бронювання автомобілів онлайн, встановлювати тарифи, керувати клієнтами, фінансами та автопарком.

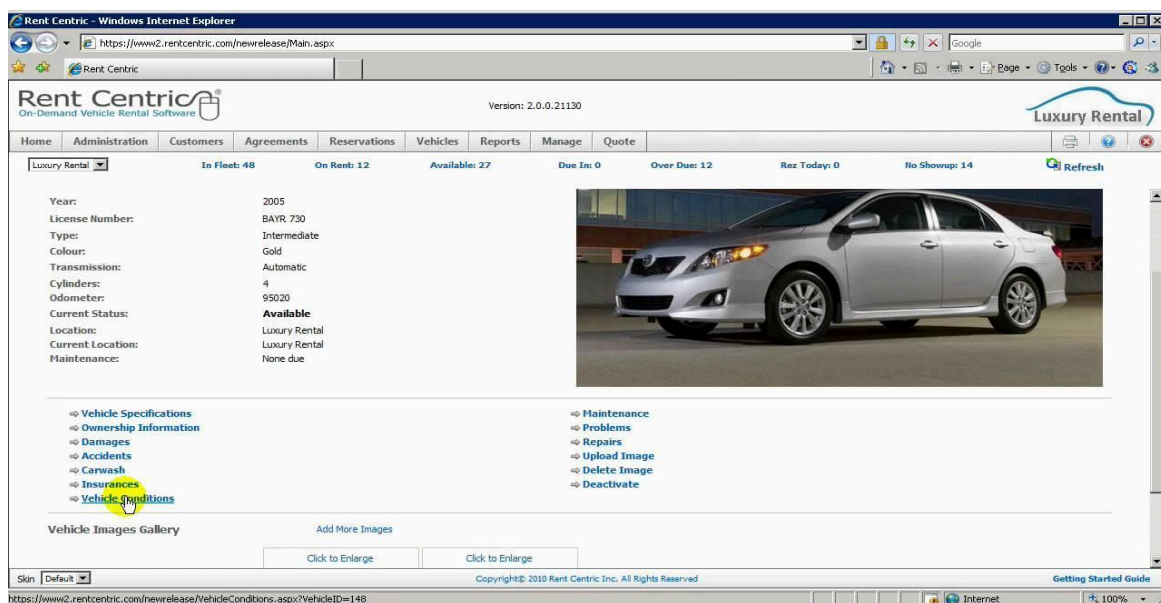


Рис. 2.3 Приклад інтерфейсу Rent Centric для керування орендою автомобілів

Переваги Rent Centric:

- підтримка онлайн-бронювання автомобілів;
- наявність функцій керування тарифами та клієнтами;
- можливість контролю автопарку і замовлень;
- наявність звітності та фінансового аналізу;
- можливість використання системи компаніями різного масштабу.

Недоліки Rent Centric:

- система може бути складною для невеликого сервісу оренди автомобілів;
- для адаптації під конкретну модель роботи може знадобитися додаткове налаштування;
- велика кількість функцій ускладнює швидке освоєння системи;
- комерційна основа використання може бути економічно недоцільною для малого проекту;
- обмежена гнучкість при необхідності реалізації індивідуальних вимог замовника.

2.4 Turo

Turo - це маркетплейс, на якому користувачам можна орендувати автомобілі від інших користувачів. Відмінним від автопрокатних компаній є те, що всі автомобілі виставляються власниками, а орендується користувачами. На сайті Turo зазначено, що користувачам можна знайти потрібний автомобіль і орендувати його без відвідування пункту прокату.

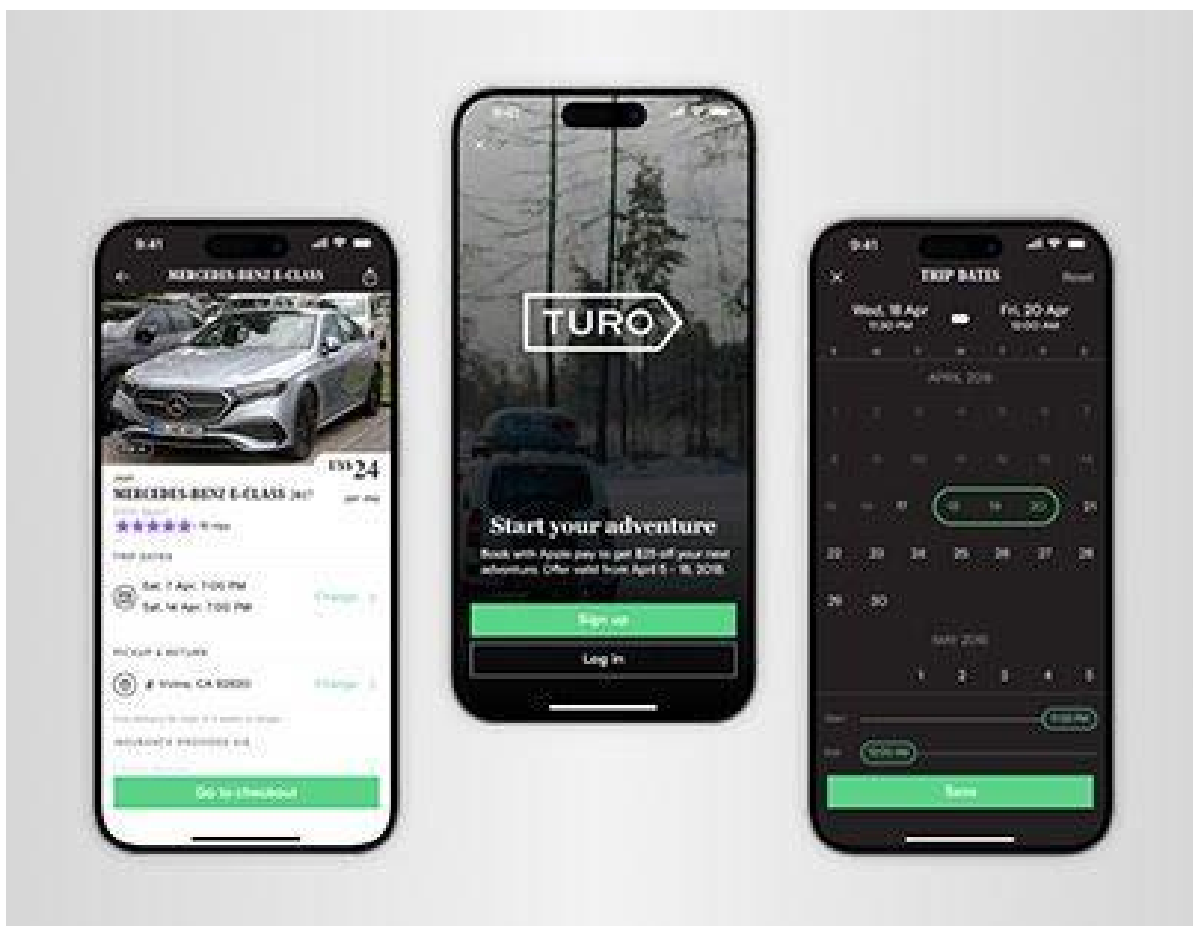


Рис. 2.4 Приклад інтерфейсу платформи Туго для пошуку та бронювання

Переваги Туго:

- Зручний каталог автомобілів;
- Зрозуміла логіка вибору автомобіля та його бронювання;
- Можливість вибору автомобіля за місцем, датами та характеристиками;
- Орієнтація на спрощену взаємодію користувача з платформою;
- Сучасний клієнтський інтерфейс.

Недоліки Туго:

- Туго не є CRM компанії;
- Адміністрування авто відбувається за правилами платформи;
- Відсутність функцій для відділу автопарку щодо тарифів, статусів, замовлень.

2.5 Узагальнення результатів аналізу

Розглянуті програмні засоби мають спільну мету — спростити процес оренди автомобілів, зробити бронювання більш зручним для клієнта, а керування автопарком більш структурованим і контрольованим для компанії. Вони дозволяють зменшити кількість ручних операцій, швидше обробляти заявки користувачів, відстежувати доступність автомобілів і підтримувати актуальність інформації про стан автопарку.

Водночас кожне з розглянутих рішень має власну специфіку та орієнтоване на різні моделі використання. RentSyst, HQ Rental Software і Rent Centric більше підходять для бізнесу, який уже має сформований автопарк і потребує комплексної системи внутрішнього керування прокатною компанією. Turo, навпаки, більше орієнтований на клієнтський досвід, швидкий пошук автомобіля, перегляд доступних варіантів і оформлення оренди через зручний інтерфейс, але не є повноцінною системою керування автопарком окремої компанії.

Таблиця 2.1

Порівняння існуючих програм аналогів

Показник	RentSyst	HQ Rental Software	Rent Centric	Turo	DriveRent
Основне призначення	CRM для оренди авто	Система керування прокатом	ПЗ для прокату та каршерингу	Маркетплейс оренди авто	Web-застосунок для оренди автомобілів
Адміністративна панель	+	+	+	лише базове керування акаунтом	+
Робота з тарифами	+	+	+	ціна задається власником авто	+
Каталог автомобілів	+	+	+	+	+
Керування автопарком	+	+	+	керування лише власними авто	+
Контроль оренд	+	+	+	контроль через заявки власників	+

Продовження таблиці 2.1

Порівняння існуючих програм аналогів

Показник	RentSyst	HQ Rental Software	Rent Centric	Turo	DriveRent
Керування клієнтами	+	+	+	перегляд даних лише в межах оренди	+
Профіль користувача	+	+	+	+	+
Локальне розгортання проєкту	-	-	-	-	+
Можливість адаптації під власну логіку	налаштування в межах готової CRM	налаштування під бізнес-процес и прокату	налаштування через параметри системи	відсутня зміна логіки платформи	повна зміна коду під вимоги проєкту
Відсутність залежності від комерційної підписки	-	-	-	-	+
Простота для невеликого сервісу	потребує налаштування CRM	потребує впровадження в бізнес	потребує освоєння модулів	проста для клієнта, не для сервісу	проста структура для малого сервісу
Орієнтація на бізнес автопрокату	+	+	+	-	+

За результатами порівняння існуючих програм-аналогів встановлено, що більшість розглянутих рішень мають базовий набір функцій для керування орендою автомобілів, зокрема роботу з тарифами, каталогом автомобілів, клієнтами, бронюваннями та контролем оренди. Водночас у багатьох системах окремі можливості реалізовані частково або обмежено, наприклад розмежування ролей користувачів, демонстраційний перегляд інтерфейсу, локальне розгортання проєкту та відсутність залежності від комерційно підписки.

3 ОГЛЯД ЗАСОБІВ РЕАЛІЗАЦІЇ ДОДАТКУ ДЛЯ ОРЕНДИ АВТОМОБІЛІВ

Для реалізації web-застосунку DriveRent необхідно обрати такі програмні засоби, які дозволять створити стабільну клієнт-серверну систему для перегляду автомобілів, бронювання транспортних засобів, керування автопарком, тарифами, користувачами та статусами замовлень. Оскільки застосунок передбачає роботу через браузер, доцільним є використання web-архітектури, у якій клієнтська частина відповідає за інтерфейс користувача, серверна частина — за бізнес-логіку, а база даних — за збереження інформації.

У загальному вигляді архітектуру web-застосунку DriveRent можна подати як взаємодію декількох основних компонентів: користувацького інтерфейсу, серверної частини, бази даних і середовища розгортання.

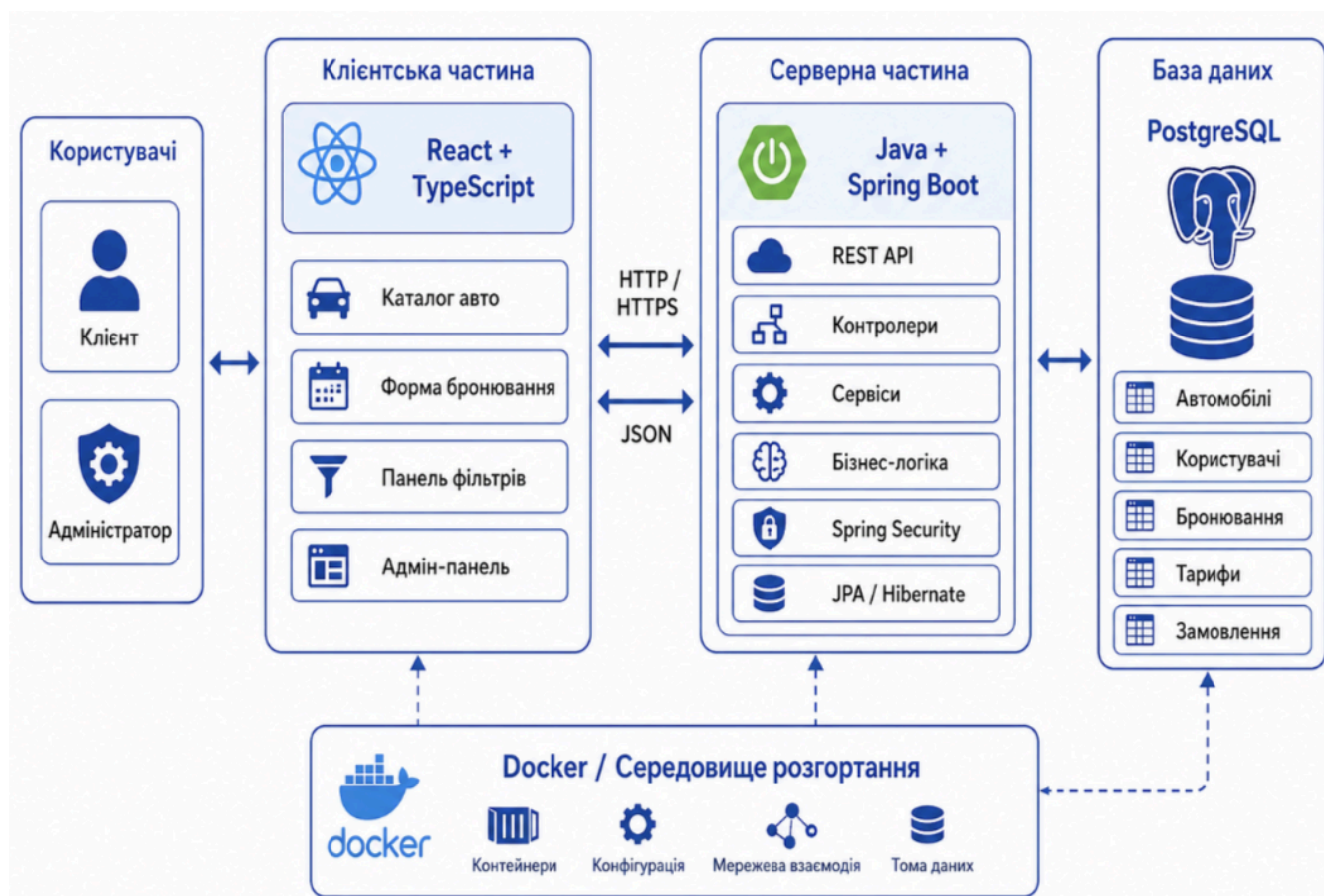


Рис. 3.1 Архітектура програмної реалізації web-застосунку DriveRent

3.1 Мова програмування Java

Мова програмування, яка використовується для реалізації серверної частини DriveRent, є Java. Вона є дуже зручною для створення клієнт-серверних застосунків. У Java використовується об'єктно-орієнтований підхід, який дозволяє створювати класи, об'єкти, інтерфейси, пакети, а також використовувати можливість спадкування. Краще структурування коду дозволяє виконувати роботу з різними сутностями, які є частиною системи оренди автомобілів.

Крім того, Java є дуже вдалим вибором для реалізації бізнес-логіки системи DriveRent. У даній мові можна легко створювати правила для бронювань, перевірки доступності автомобілів, зміни статусів замовлення або взаємодії з базою даних. Сполучення можливостей мови Java дозволяє не тільки виконати потрібну роботу, а й ефективно її розширити в майбутній період.

3.2 Фреймворк Spring Boot

Одним з найбільш доцільних варіантів фреймворку для створення серверної частини DriveRent є Spring Boot. Відомо, що Spring Boot призначений для створення автономних (stand-alone), виробничих (production-grade) Spring-застосунків, які можна швидко запустити. У випадку використання серверної частини DriveRent, відповідно до опису фреймворку, можна створити REST API, тоді клієнтська частина і серверна стануть здатними обмінюватись даними між собою.

3.3 Система управління базами даних PostgreSQL

Для збереження даних DriveRent використовуватиметься система управління базами даних PostgreSQL. Відомо, що це об'єктно-реляційна система керування базами даних з відкритим кодом, яка підтримує мову SQL та дозволяє безпечно

збереження та масштабування даних. У документації PostgreSQL зазначено, що система підтримує багато можливостей та стандартів SQL.

База даних PostgreSQL, що буде використовуватися для DriveRent, дозволить зберігати такі дані, які можна віднести до системи: інформація про автомобілі, про користувачів, про категорії автомобілів, тарифи для оренди, історія та статуси замовлень. Зокрема, окремі таблиці бази даних можна віднести до автомобілів, інші - до користувачів, інші - до замовлень. У цих таблицях можна зберігати інформацію про те, які саме автомобілі вибрали відвідувачі, а також який час вони обрали для оренди автомобілів.



Рис. 3.2 Послідовність у web-застосунку DriveRent

3.4 React

Для вирішення завдання клієнтської частини варто використовувати бібліотеку React. React - це бібліотека для створення інтерфейсів користувача, які здійснюються шляхом використання окремих компонентів. На основі документації React визначено, що React дозволяє створювати складні інтерфейси на основі окремих компонентів, які вирішуються окремо та відокремлюються від інших частин інтерфейсу.

У випадку створення DriveRent буде доцільно використовувати бібліотеку React. Компонентами вважається, наприклад, інтерфейси автомобіля, списки автомобілів, форму бронювання, форму фільтрів, інтерфейс адміністратора, таблицю бронювань та форму тарифів. Відокремлення окремих компонентів дозволить спростити процес створення інтерфейсу, оскільки можна зробити відокремлення окремої частини від інших та змінити її без впливу на інші частини інтерфейсу.

3.5 TypeScript

Разом із використанням бібліотеки React доцільно використовувати мову TypeScript. TypeScript - це мова, що розширює JavaScript та дозволяє використовувати типізацію для визначення типів змінних, об'єктів, функцій тощо. Відокремлення типізації дозволяє виявляти помилки написання коду. TypeScript виділяє типізацію від JavaScript, що дозволяє редакторам коду пропонувати підказки для введення змінних, методів, властивостей тощо, а також виявляти помилки написання коду.

Відокремлення мови TypeScript може бути використане для опису структур даних, що передаються між компонентами інтерфейсу. Використання мови TypeScript для визначення типів дозволяє виявляти помилки в даних, що передаються між компонентами інтерфейсу, а також відокремлює його від відкритого коду JavaScript.

3.6 Docker

Для розгортання створеного застосунку можна використовувати засіб контейнеризації Docker. Docker - це платформа, що дозволяє створювати та запускати застосунки у відокремлених контейнерах. Відокремлення контейнерів дозволяє створювати застосунки, які працюють відокремлено від інших застосунків та спрощує процес розробки застосунків.

У разі розробки DriveRent такий засіб буде корисним, оскільки Docker дозволяє відокремити базу даних (PostgreSQL), backend-сервер та frontend-частину. Розгортання застосунку у відокремлених контейнерів дозволить спростити процес налаштування середовища розробки, відокремити його від комп'ютера розробника, а також спростити процес розгортання застосунку.

3.7 Середовище розробки — IntelliJ IDEA

Для розробки web-застосунку DriveRent використовувалася інтегроване середовище розробки (IDE) IntelliJ IDEA. Ця IDE підходить для створення Java-проектів і підтримує Spring Boot, Maven/Gradle, Git, а також розгортання вебзастосунків.

IntelliJ IDEA використовувалася для розробки серверної частини web-застосунку DriveRent. Класи, моделі, сервіси та компоненти доступу до даних вебзастосунку були розроблені в цій IDE для проектування та реалізації бази даних за допомогою PostgreSQL.

IntelliJ IDEA дозволила проектувати відповідно до принципів об'єктно-орієнтованого програмування, що було необхідним для побудови web-застосунку DriveRent. Web-застосунок мав окремі модулі для користувачів, автомобілів, категорій, тарифів, оренди та замовлень.

3.8 Система контролю версій — Git та GitHub

Для контролю версій програмного проекту використовуватиметься система контролю версій Git. Ця система дозволяє відстежувати зміни, що вносяться до проекту та коду, даючи розробникам можливість повертатися до попередніх версій коду.

Git можна використовувати для контролю версій як серверної, так і клієнтської частини web-застосунку DriveRent. Це буде особливо корисно для розробки модулів, що керують каталогом доступних автомобілів, орендою автомобілів, тарифами та статусами замовлень.

Для зберігання репозиторію Git проекту можна використовувати такий сервіс, як GitHub. GitHub дозволяє зберігати проект віддалено та дає змогу переглядати історію змін проекту, створювати резервні копії проекту та організовувати файли проекту. Крім того, GitHub допоможе у демонстрації проекту та майбутній підтримці вебзастосунку.

Таблиця 3.1

Використані засоби реалізації

Засіб реалізації	Призначення у проєкті DriveRent
Java	Реалізація серверної логіки, моделей, сервісів і правил обробки бронювань
Spring Boot	Створення REST API, обробка запитів, організація backend-частини
PostgreSQL	Збереження даних про автомобілі, користувачів, тарифи, бронювання та замовлення
React	Створення клієнтського інтерфейсу web-застосунку
TypeScript	Типізація даних на frontend-частині та зменшення кількості помилок у коді
Docker	Контейнеризація застосунку та спрощення запуску середовища розробки
IntelliJ IDEA	Розробка серверної частини, робота з Java, Spring Boot, моделями, сервісами та репозиторіями
Git	Контроль версій програмного коду та збереження історії змін у проєкті
GitHub	Віддалене зберігання репозиторію, резервне копіювання коду та організація роботи з проєктом

4 ПРОЄКТУВАННЯ WEB-ЗАСТОСУНКУ DRIVERENT

4.1 Логічна структура системи

Розробка web-застосунку DriveRent передбачає поділ системи на логічні рівні. Такий поділ дозволить відокремити користувацький інтерфейс від бізнес-рівня та рівня даних, кожен з яких відповідатиме за певну частину програми.

Логічна структура web-застосунку DriveRent складається з трьох основних рівнів. Перший — це рівень представлення, який забезпечить взаємодію між користувачами системи та самою системою. Цей рівень складається з окремих інтерфейсів для клієнта та адміністратора системи. Клієнт має доступ до каталогу доступних транспортних засобів у додатку та зможе створювати бронювання оренди. Водночас адміністратор має доступ до інтерфейсу системи для управління автопарком, тарифами на оренду, а також статусами транспортних засобів та бронювань у системі.

Бізнес-рівень міститиме основні функціональні модулі вебзастосунку. До них входитимуть модулі для управління користувачами системи, транспортними засобами в автопарку, категоріями транспортних засобів, тарифами на оренду, бронюваннями, замовленнями на транспортні засоби, статусами автомобілів та бронювань, а також звітності для бізнесу.

Нарешті, рівень доступу до даних відповідатиме за взаємодію між додатком та програмним забезпеченням бази даних. Цей рівень дозволить здійснювати доступ, зберігання, оновлення та видалення даних щодо користувачів, транспортних засобів, тарифів та бронювань у системі. Завдяки відокремленню цього рівня від бізнес-рівня додатка логічна структура системи стане більш зрозумілою та зручною для розробників системи.

Логічна структура web-застосунку DriveRent представлена на рисунку 4.1.

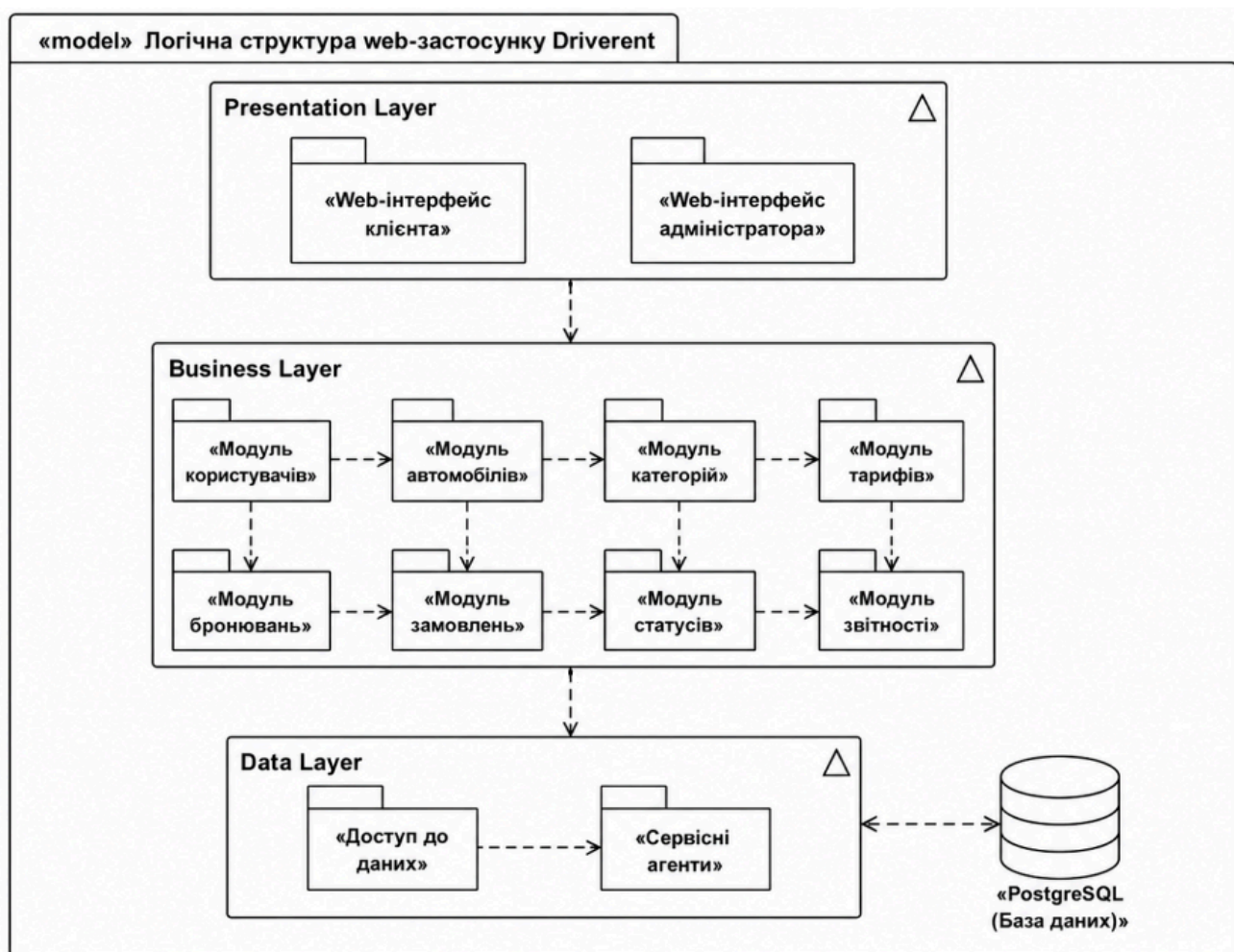


Рис. 4.1 Логічна структура web-застосунку DriveRent

4.2 Діаграма прецедентів web-застосунку DriveRent

У web-застосунку DriveRent передбачено два типи користувачів: клієнт та адміністратор. Клієнт може переглядати доступні автомобілі, обирати умови оренди, робити бронювання та переглядати свої замовлення. Адміністратор має доступ до додаткових функцій, що дозволяють керувати автопарком, тарифами на оренду, статусом автомобілів та замовленнями.

Основними функціями клієнта є реєстрація та вхід у систему, перегляд доступних автомобілів, фільтрування доступних автомобілів за вибраними параметрами, перегляд детальної інформації про автомобіль, оформлення бронювання оренди та перегляд своїх замовлень на оренду.

Адміністратор може виконувати всі функції клієнта, а також керувати автомобілями в автопарку, категоріями автомобілів, тарифами на оренду, статусом автомобілів та замовленнями на оренду. Крім того, адміністратор може переглядати бронювання оренди за їхнім статусом для управління орендою автомобілів.

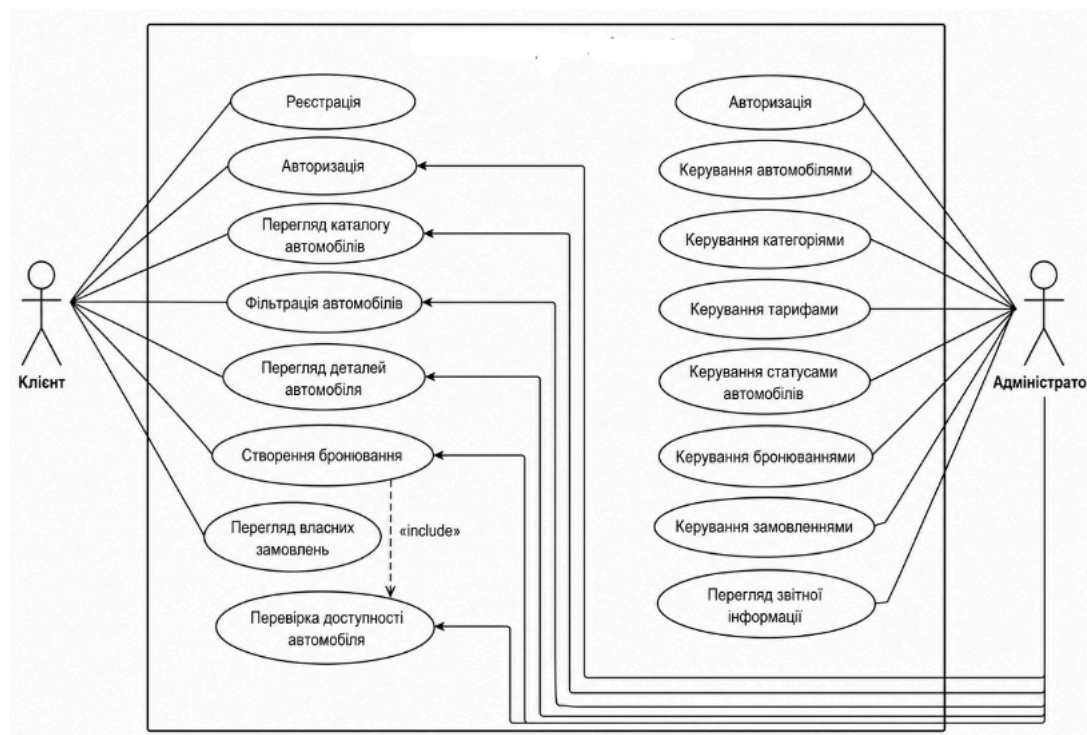


Рис. 4.2 Діаграма прецедентів web-застосунку DriveRent

4.3 Діаграма предметної галузі web-застосунку DriveRent

Діаграма предметної галузі використовується для відображення основних сутностей, що беруть участь у процесі оренди автомобілів, і взаємозалежностей між ними. Використання діаграми предметної галузі дозволяє формувати загальне уявлення про структуру даних майбутньої системи.

Користувачі можуть мати однією з двох ролей - роллю клієнта або адміністратора. Клієнти виконують дії стосовно перегляду автомобілів та створення вибраних їми бронювань, в той час як адміністратор виконує дії стосовно керування автомобілів, встановленням тарифів, статусів та замовлень.

Автомобілі є основними елементами системи. Кожен автомобіль має власною категорією, встановленим тарифом і статусом. Статус встановлюється для відображення стану автомобіля - чи є він доступним для оренди, чи вже знаходиться у відсутності.

Бронювання. Іншими словами, замовлення. Клієнт, що створює бронювання виконує вибір автомобіля та періода, після чого відбувається створення замовлення на вибраний автомобіль, здійснюваного після підтвердження бронювання. Замовлення також володіє власним статусом, який відображає його стан: створене, підтверджене, активне, завершене або скасоване.

Діаграму предметної галузі web-застосунку DriveRent наведено на рисунку 4.3.

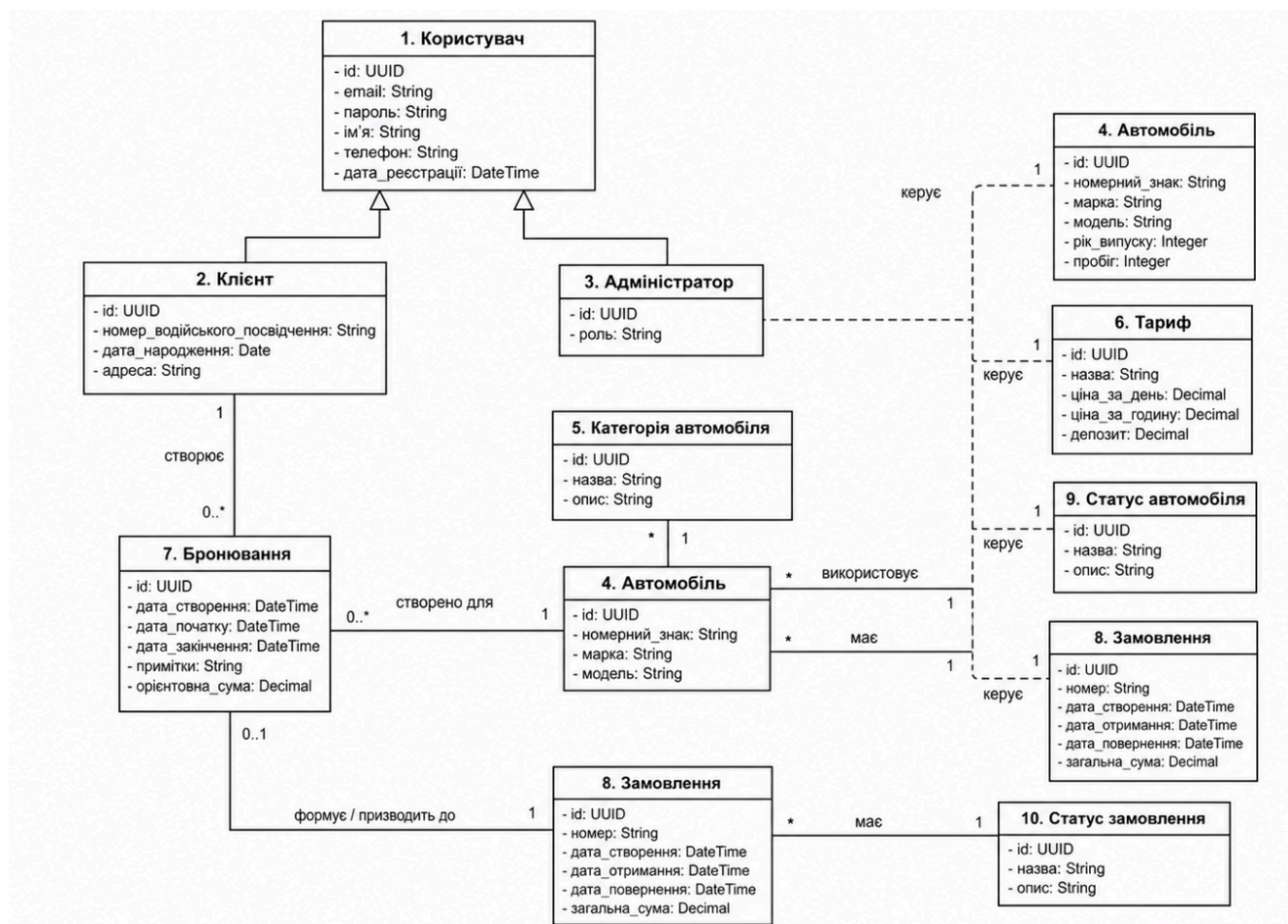


Рис. 4.3 Діаграма предметної галузі web-застосунку DriveRent

4.4 Діаграми діяльності

Діаграми діяльності використовуються для ілюстрації послідовності дій, що виконуються під час реалізації певного процесу в системі. Для web-застосунку DriveRent доцільно змодельовати два сценарії — процес оренди автомобіля клієнтом та управління замовленнями адміністратором.

Перший сценарій описує дії клієнта під час вибору та оренди автомобіля. Етапи включають в себе вхід клієнта в систему, перегляд каталогу автомобілів, застосування різних фільтрів, перегляд детальної інформації про автомобіль, що його цікавить, вибір дат оренди автомобіля та створення замовлення на оренду. Після створення замовлення на оренду система перевіряє, чи доступний автомобіль у вибрані дати оренди. Якщо автомобіль доступний у вибрані клієнтом дати оренди, замовлення на оренду зберігається в системі, а клієнт отримує підтвердження про створення замовлення на оренду. Якщо система визначає, що запитуваний автомобіль недоступний у вибрані клієнтом дати оренди, клієнт отримує відповідне повідомлення та пропозицію вибрати інший автомобіль або інші дати оренди.

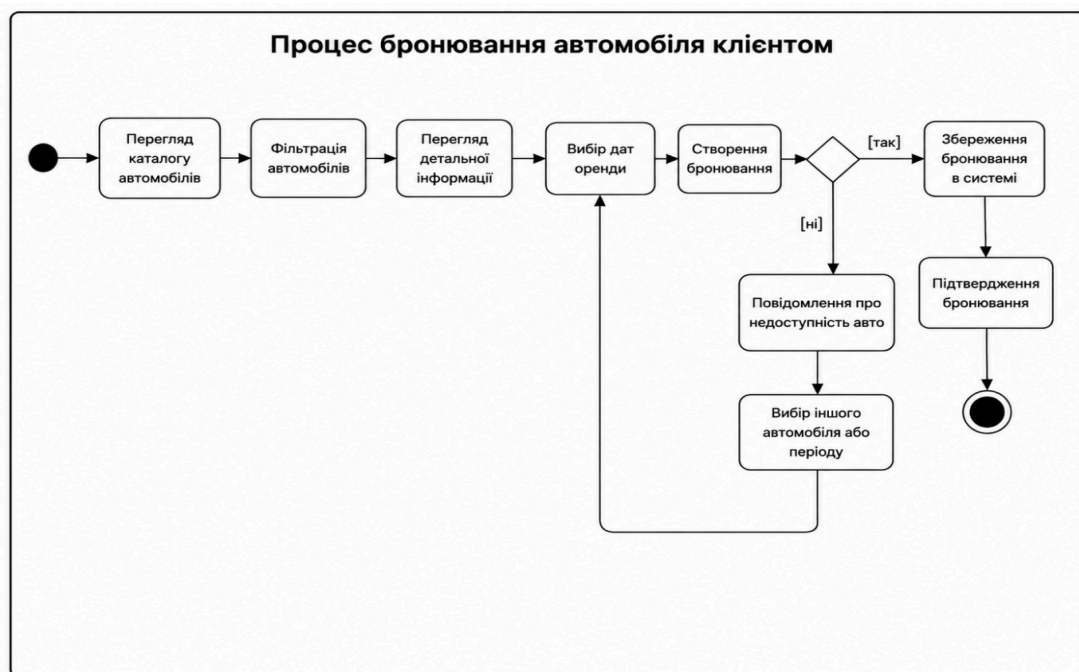


Рис. 4.4 Діаграма діяльності оренди автомобіля клієнтом

Другий сценарій, який потрібно змодельовати, — це дії адміністратора щодо замовлень у системі. Адміністратор повинен увійти в систему, перейти до панелі адміністратора, переглянути список замовлень, розміщених клієнтами, натиснути на конкретне замовлення на оренду, яке потрібно обробити, та змінити статус цього замовлення відповідно до поточного статусу оренди автомобіля клієнтом. Крім того, адміністратор також може оновлювати статус автомобілів у системі оренди, щоб відобразити їхню доступність для оренди клієнтами. Наприклад, статус автомобіля можна змінити з «доступний для оренди» на «орендований» або з «орендований» на «доступний для оренди».



Рис. 4.5 Діаграма діяльності адміністратора, який управляє замовленнями на оренду

4.5 Діаграма класів web-застосунку DriveRent

Діаграма класів використовується для опису структури програмного додатка з точки зору його класів, атрибутів класів, методів класів та взаємозв'язків між ними класами. Діаграма класів web-застосунку DriveRent дозволяє показати, які класи використовуються при розробці вебзастосунку.

Основні класи додатка включають: User, Client, Admin, Car, CarCategory, Tariff, Booking, Order, CarStatus та OrderStatus. Клас User зберігає дані користувача додатка, такі як його ID, адреса електронної пошти, пароль, ім'я, номер телефону та його роль. Класи користувачів Client та Admin є похідними від класу User.

Клас Car зберігає дані про автомобіль, який доступний для оренди. Атрибути цього класу включають марку автомобіля, модель автомобіля, рік випуску, номерний знак автомобіля, тип палива, яке використовує автомобіль, тип коробки передач автомобіля, кількість місць в автомобілі, пробіг автомобіля та поточний стан автомобіля. Клас CarCategory групує автомобілі за категоріями, до яких вони належать. Клас Tariff містить умови оренди автомобілів.

Клас Booking зберігає бронювання автомобіля клієнтом вебзастосунку. Атрибути класу включають дані клієнта, який бронює автомобіль, автомобіль, що бронюється, час початку та закінчення оренди, статус бронювання та орієнтовну вартість бронювання автомобіля. На основі підтверджених бронювань створюється клас Order. Клас «Order» використовується для управління орендою автомобіля клієнтом. Статус автомобілів та замовлень визначається відповідно у класах «CarStatus» та «OrderStatus».

Також важливо включити класи, пов'язані зі службами та контролерами вебзастосунку. Контролери, такі як «CarController», «BookingController» та «OrderController», приймають запити від клієнта. Сервіси додатка, такі як CarService, BookingService та OrderService, містять бізнес-логіку додатка, наприклад перевірку наявності автомобіля для оренди, бронювання автомобіля, зміну статусу заброньованого автомобіля та прийняття замовлень на оренду автомобілів. Репозиторії використовуються для доступу до даних, що зберігаються в базі даних PostgreSQL.

Діаграма класів web-застосунку представлена на рисунку 4.6.

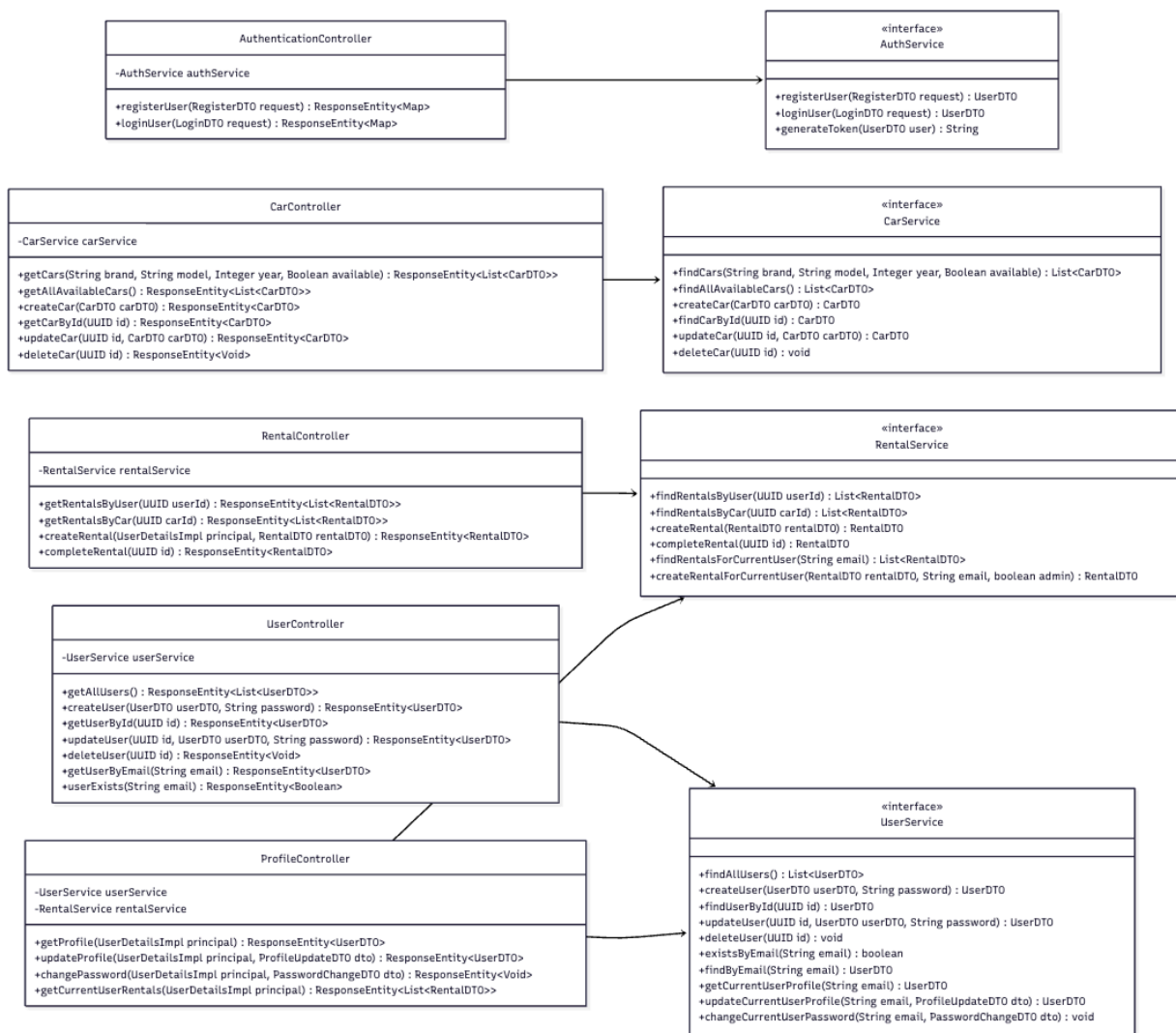


Рис. 4.6 Діаграма класів web-застосунку DriveRent

4.6 Схема компонентів web-застосунку DriveRent

Схема компонентів відображає структуру компонентів вебзастосунку.

Web-застосунок DriveRent має клієнт-серверну структуру. Клієнтська частина реалізована на базі React та TypeScript; її основними компонентами є категорії автомобілів, фільтри, деталі автомобілів, бронювання та панель адміністратора.

Серверні компоненти реалізовані на Java та Spring Boot. Серверні компоненти підключаються до клієнтських компонентів через REST API, виконують бізнес-логіку та звертаються до бази даних. Основними серверними

компонентами є: аутентифікація, клієнти, автомобілі, бронювання, замовлення, тарифи та статуси.

Серверні компоненти мають базу даних PostgreSQL. Серверні компоненти взаємодіють з базою даних через ORM (Java Persistence API (JPA) та Hibernate).

Взаємодія компонентів відбувається наступним чином: Клієнт працює у веб-інтерфейсі. Серверний компонент отримує запит від клієнтського компонента через REST API, звертається до бази даних та повертає результат у форматі JSON.

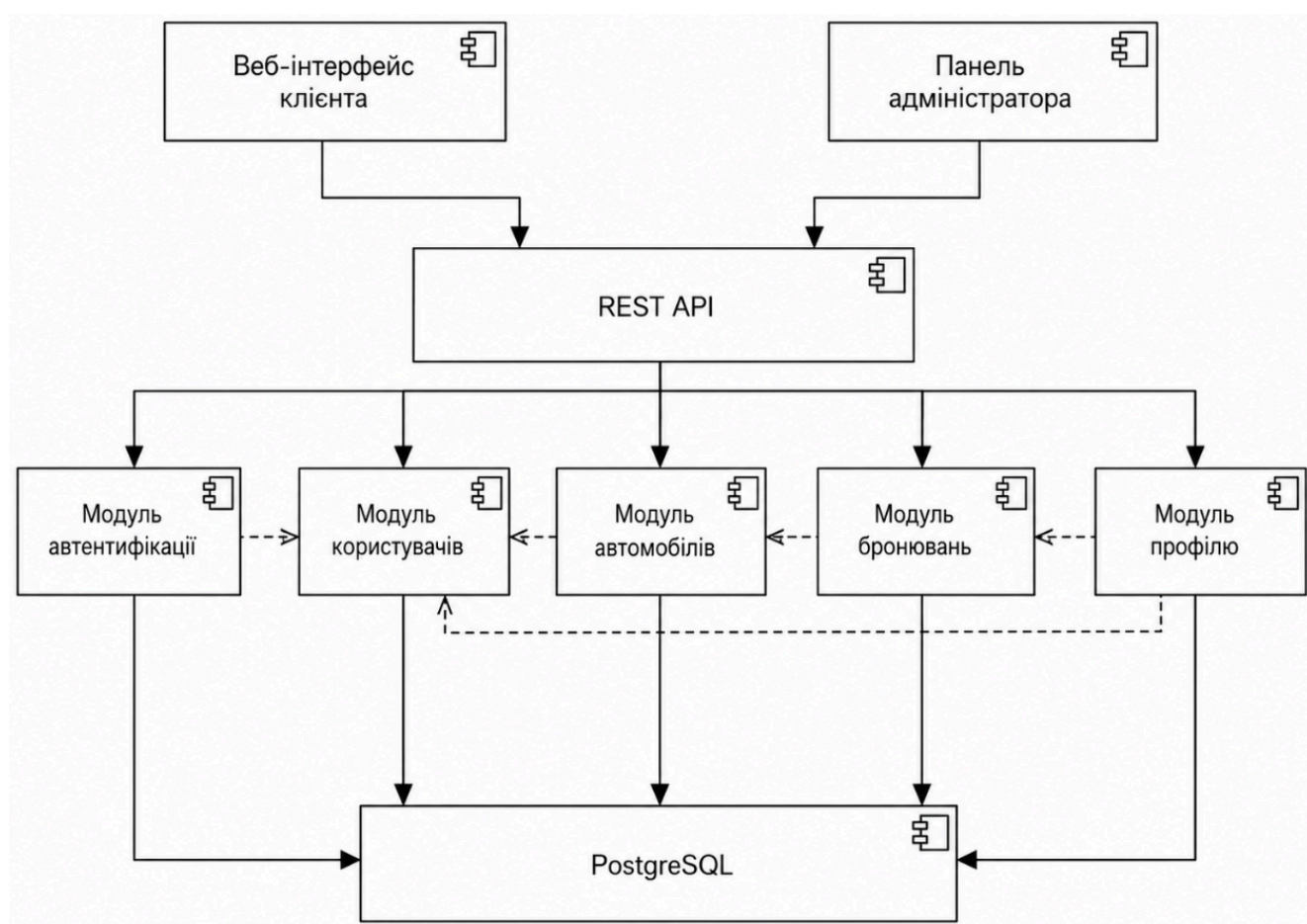


Рис. 4.7 Діаграма компонентів web-застосунку DriveRent

4.7 Схеми розгортання web-застосунку DriveRent

Схеми розгортання використовується для відображення фізичної структури розміщення компонентів програмного забезпечення. На цій схемі вказується, які компоненти виконуються на вузлах і як вони взаємодіють між собою.

Web-застосунку DriveRent відповідає структурі клієнт-сервер. Користувач взаємодіє з додатком у браузері, що виконується на персональному комп'ютері, планшеті чи смартфоні. У браузері відкривається клієнтська частина додатка.

Для розгортання сервісу використовується додаток Spring Boot. Він приймає запити від клієнтської частини та відповідає відповідно до виконання бізнес-логіки. Сервіс відповідає за авторизацію, отримання списку автомобілів, створення бронювань, відновлення статусів автомобілів.

Базою даних використовується система PostgreSQL. Діаграма вказує, що база даних взаємодіє з серверною частиною додатка в результаті виконання SQL-запитів.

Для спрощення процесу запуску середовища розробки може використовуватися Docker. За допомогою Docker виконується розгортання окремо сервісу, бази даних та окремо серверної частини. Це створює більш зрозумілу та зручну структуру розгортання.

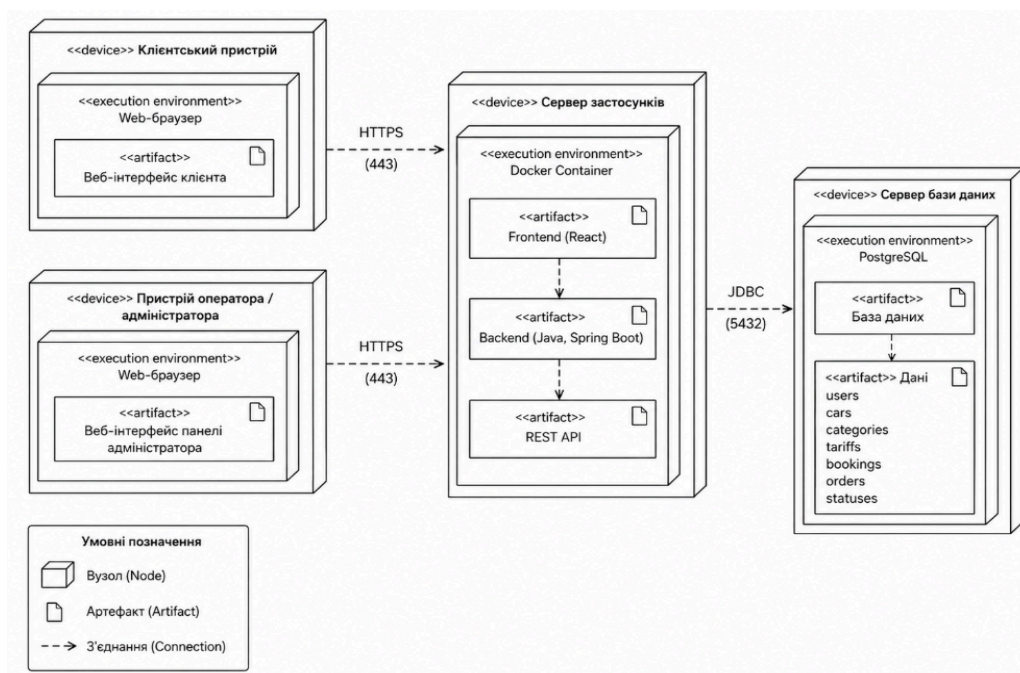


Рис. 4.8 Схема розгортання web-застосунку DriveRent

4.8 Діаграма станів меню web-застосунку DriveRent

Діаграма станів меню використовується для відображення можливих переходів між основними екранами web-застосунку DriveRent.

Діаграма станів меню дозволяє відобразити логіку руху користувача та доступні стани для клієнта та адміністратора.

Після відкриття вебзастосунку клієнт потрапляє на головну сторінку, з якої можна перейти до процесу авторизації або перегляду каталогу автомобілів. У випадку, якщо користувач не є авторизованим, він може переглядати інформацію про автомобілі, але для бронювання їх потрібна авторизація або реєстрація.

Основними станами меню вебзастосунку є наступні: головна сторінка, авторизація, реєстрація, каталог автомобілів, сторінка автомобіля, форма бронювання, особистий кабінет клієнта, список замовлень, адміністративна панель, керування автопарком, керування тарифами та керування замовленнями.

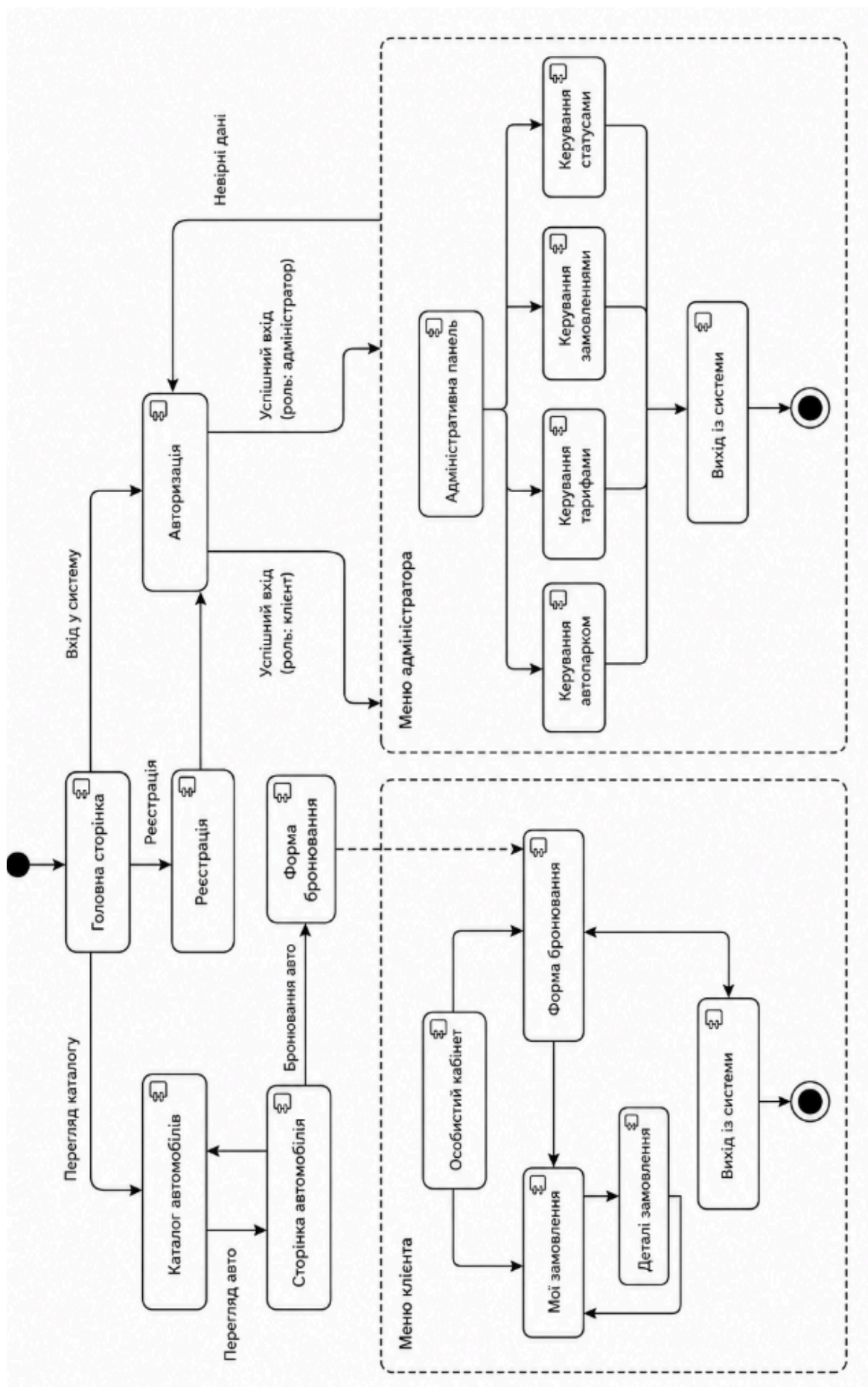


Рис. 4.9 Діаграма станів меню web-застосунку DriveRent

5 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ WEB-ЗАСТОСУНКУ DRIVERENT

5.1 Реалізація серверної частини

Серверна частина web-застосунку DriveRent реалізується за допомогою мови програмування Java та фреймворку Spring Boot. Вона відповідає за процес обробки запитів, що приходять від клієнтської частини, виконання основної бізнес-логіки та взаємодії з базою даних відносно об'єктів.

Основою серверної частини є багаторівнева структура. У неї входять моделі даних, репозиторії, сервіси та контролери, що дозволяє легко розділити складну логіку дії відділу на окремі частини, що спрощує подальшу підтримку.

Основними моделями є наступні з описаних об'єктів: користувач, автомобіль, категорія автомобіля, тариф, бронювання, замовлення та їх статуси. Кожен з цих об'єктів відображає одну із основних сутностей.

Репозиторії відповідно виконують дії відносно бази даних. Вони виконують дії встановлення, читання, видалення, зміни. Репозиторії, наприклад, дозволяють отримати список всіх автомобілів, знайти відповідний автомобіль, зберегти новий або змінити існуючий.

Сервіси виконують основну логіку. Вони відповідають за перевірку доступності автомобіля, створення бронювань, змінювання статусу автомобіля, замовлення та взаємодію відповідних об'єктів.

Контролери обробляють всі запити від клієнтської частини. Вони виконують дії відносно перегляду каталогу автомобілів, перегляду детальної інформації транспортного засобу, створення бронювань, управління тарифами, статусами та замовленнями. Взаємодія між клієнтом та сервером відбувається через REST API.

Дані зберігаються в базі даних PostgreSQL. Структура Java-класів, що відповідає предметній області, встановлюється через JPA/Hibernate.

Особу увагу приділено перевірці доступності автомобіля. Перед виводом і створенням бронювання відділ перевіряє, чи зайнятий автомобіль в вибраний

період. Якщо автомобіль вже відданий або в оренді, то бронювання не створюється.

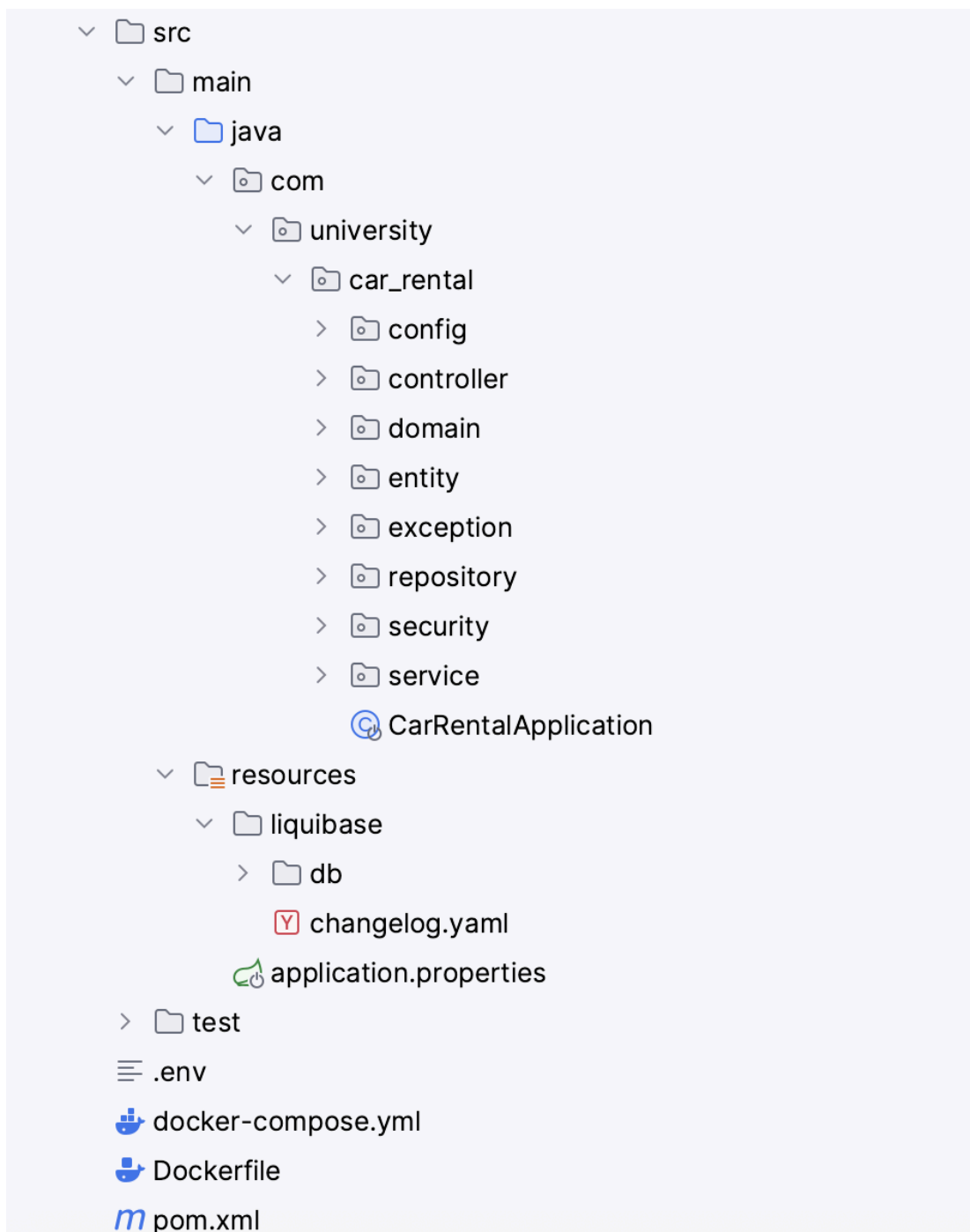


Рис. 5.1 Структура серверної частини відділу DriveRent

5.2 Реалізація клієнтської частини

Клієнтська частина web-застосунку DriveRent реалізується за допомогою бібліотеки React та мови TypeScript. Клієнтська частина відповідатиме за відображення інтерфейсу, взаємодію зі клієнтом/адміністратором, а також надсилання запитів до серверної частини та отримання відповіді у форматі JSON.

Основними завданнями клієнтської частини є зручне для клієнтів та адміністраторів доступ до всіх функцій web-застосунку. Для клієнтів передбачено можливість перегляду каталогу автомобілів, їх фільтрації, перегляду детальної інформації про автомобіль, а також відкриття форми для вибору дат оренди для автомобіля та створення бронювання. Для адміністраторів передбачена окрема панель, де можна працювати з автопарком, тарифами, статусами автомобілів та бронюваннями.

Клієнтська частина будується за компонентним підходом. Відображенням елементів інтерфейсу є окремі компоненти, які можуть використовуватися для створення інших елементів інтерфейсу. Прикладами таких компонентів для системи DriveRent є компоненти картки автомобіля, панелі фільтрації, форми бронювання, таблиці замовлень, форми редагування автомобіля та елементів адміністративної панелі.

Використання мови TypeScript дозволяє описати структуру даних, що надходять від серверної частини системи. Важливими для опису є типи або інтерфейси, що описують структури об'єктів, таких як автомобіль, користувач, тариф, бронювання або замовлення. Використання опису структури даних дозволяє зменшити кількість помилок у системі.

Взаємодія клієнтської частини зі серверною частиною відбувається через REST API. При виконанні окремих дій користувачем відбувається надсилання запиту від клієнтської частини до серверної. Відповідь від серверної частини викликає відображення результатів від клієнтської частини.

Для клієнтів важливими є наступні екрани: головна сторінка, каталог автомобілів, сторінка детальної інформації про автомобіль, форма бронювання та

сторінка власних замовлень. Для адміністраторів важливими є сторінки для керування автомобілями, тарифами, бронюваннями, замовленнями та статусами автомобілів.

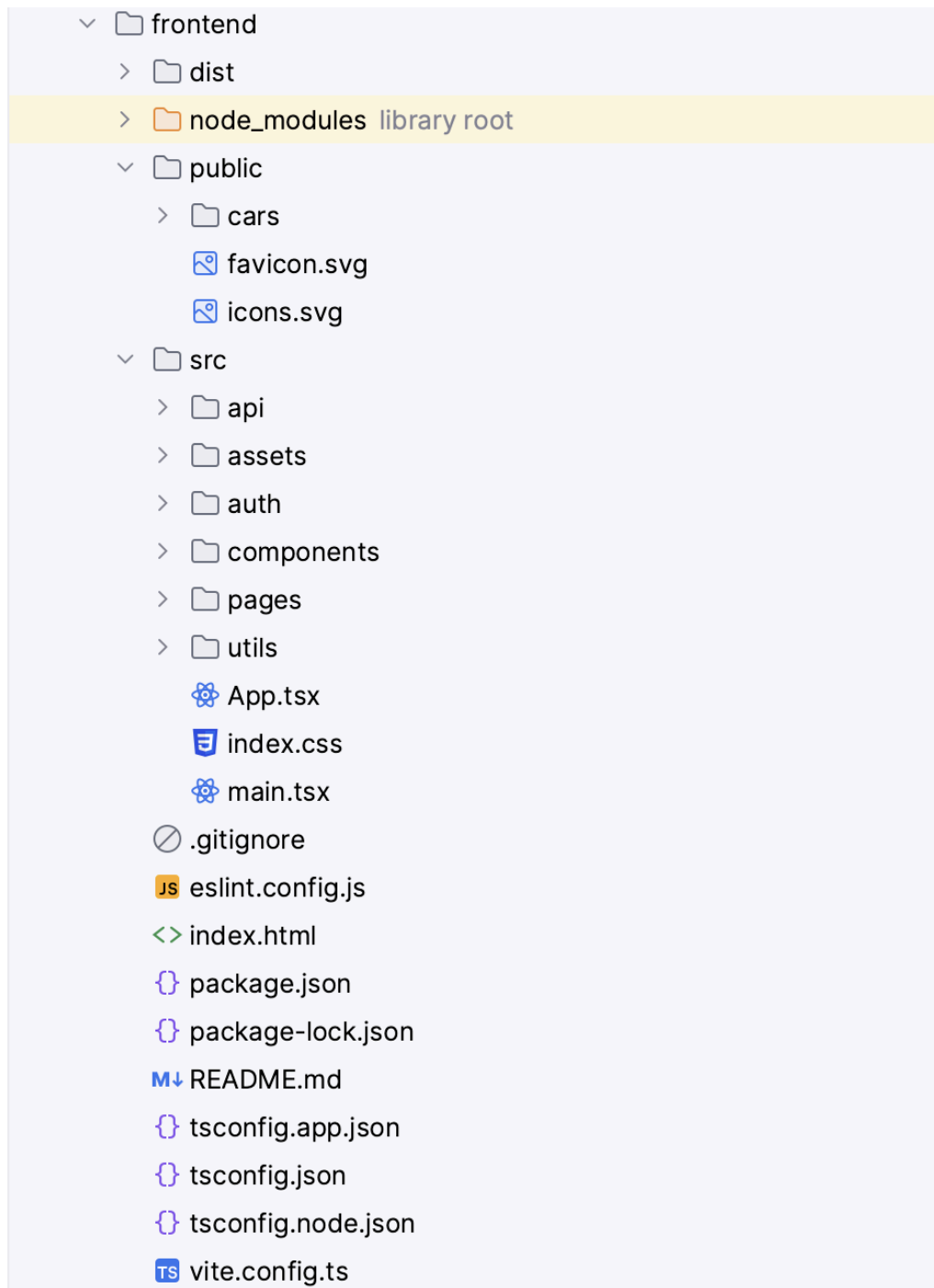


Рис. 5.2 Структура клієнтської частини web-застосунку DriveRent

5.3 Екранні форми web-застосунку

Екранні форми web-застосунку DriveRent відображають основні сценарії роботи користувачів із системою. Інтерфейс застосунку поділено відповідно до ролей: гість, звичайний користувач та адміністратор. Такий підхід дозволяє обмежити доступ до адміністративних функцій і надати кожному типу користувача лише ті можливості, які відповідають його ролі.

Для неавторизованого користувача передбачено сторінки Login та Register. Вони використовуються для входу в систему або створення нового акаунта. Екран авторизації містить welcome-блок і форму введення облікових даних, а сторінка реєстрації дозволяє створити новий обліковий запис.

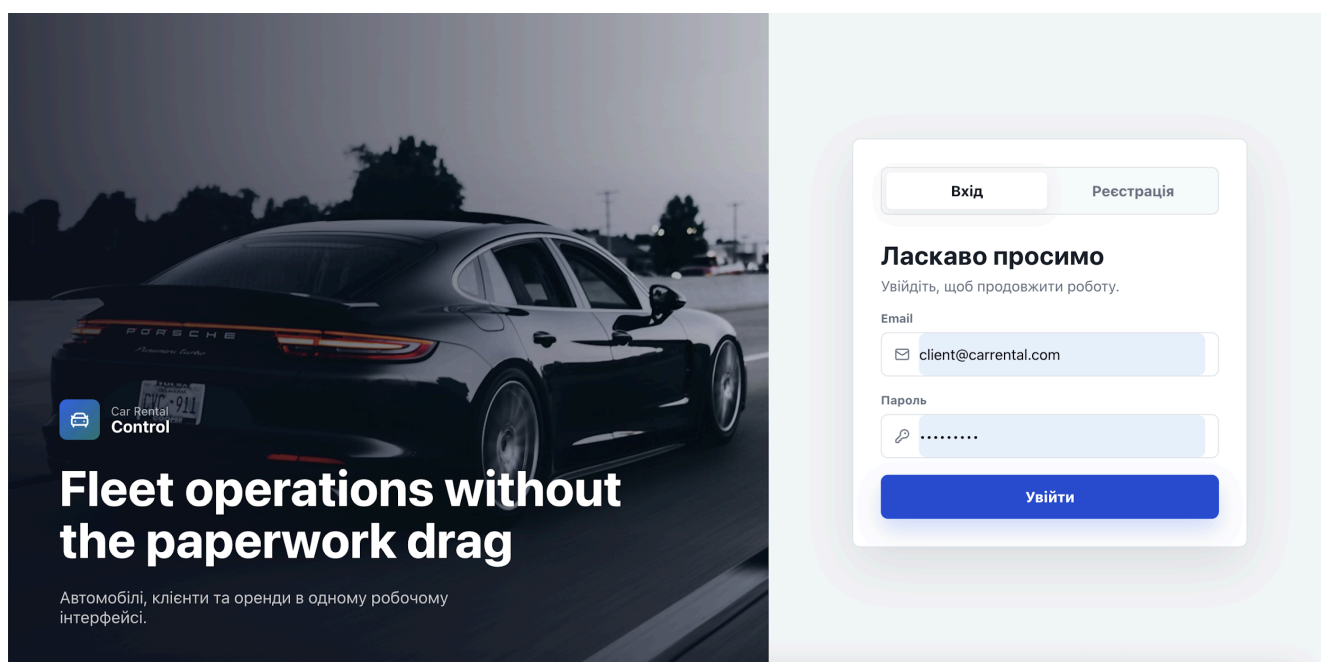


Рис. 5.3 Сторінка авторизації та реєстрації користувача

Після входу в систему користувач із роллю USER отримує доступ до основних сторінок клієнтської частини застосунку. Сторінка «Головна» є стартовою сторінкою сервісу. Вона містить короткий інформаційний блок, швидкі дії, популярні автомобілі та підказку щодо оформлення оренди.

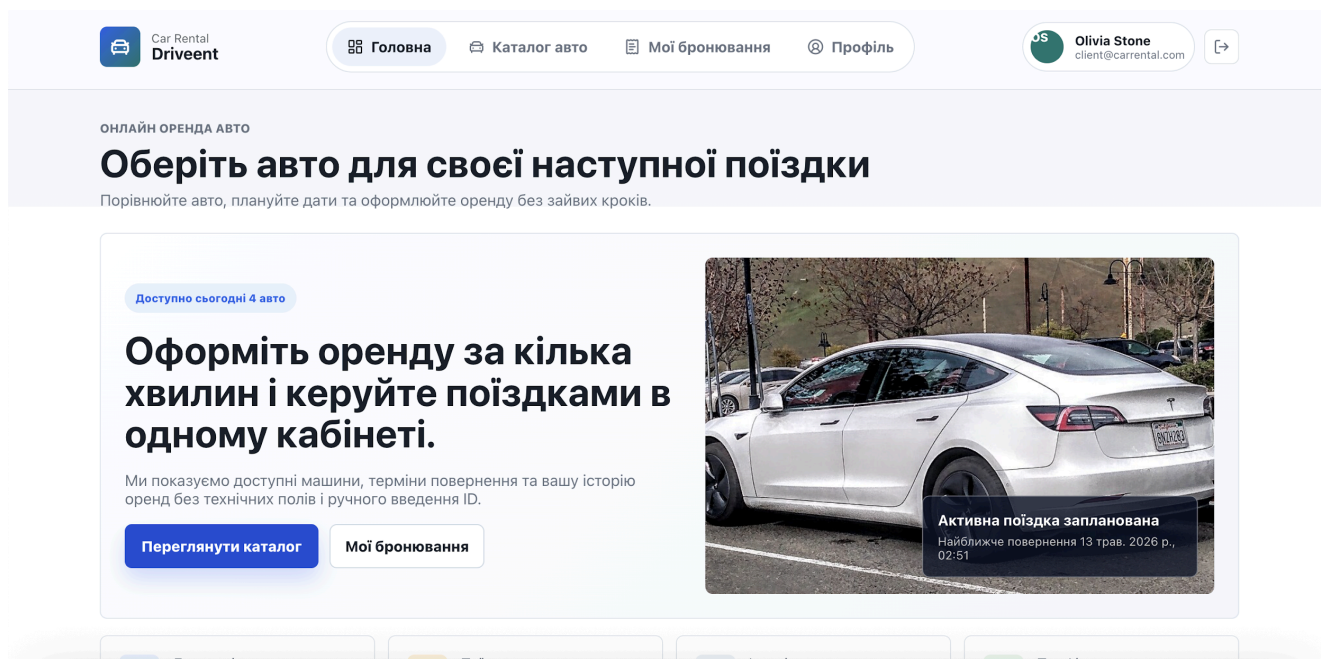


Рис. 5.4 Головна сторінка web-застосунку DriveRent

Сторінка «Каталог авто» призначена для перегляду доступних автомобілів. Користувач бачить список машин із фото, роком випуску, номерним знаком і ціною оренди за добу. З цієї сторінки можна перейти до оформлення бронювання обраного автомобіля.

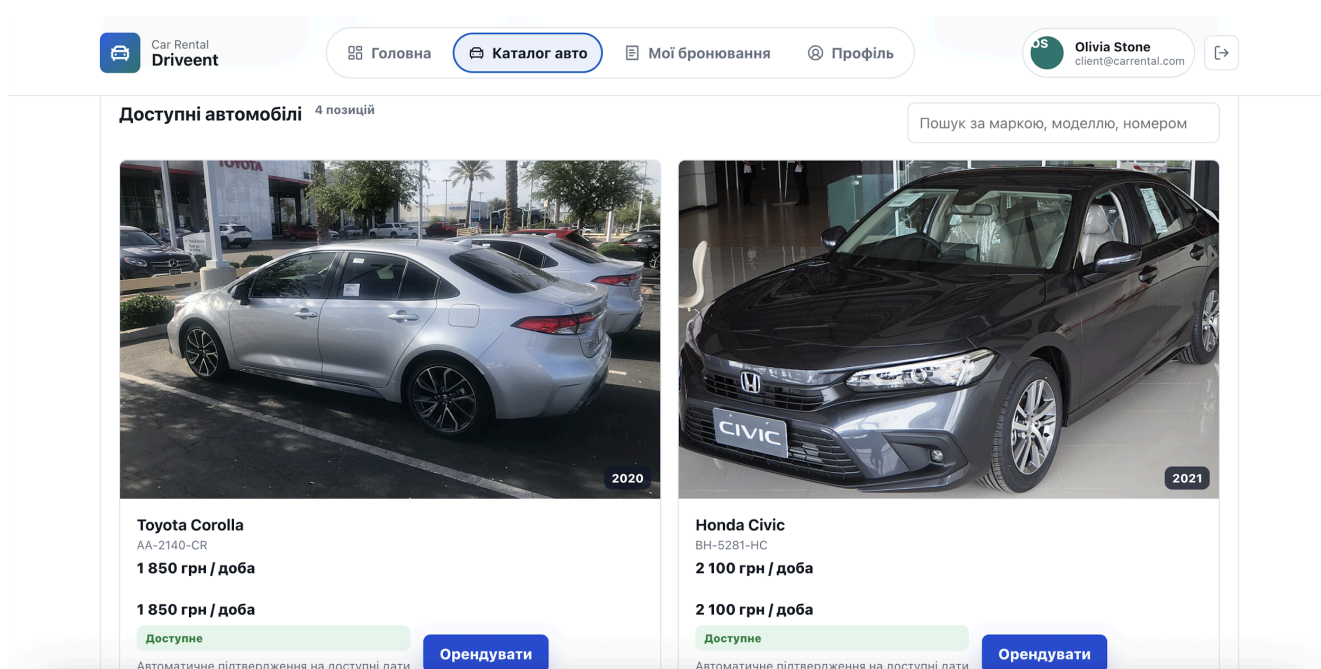


Рис. 5.5 Каталог автомобілів web-застосунку DriveRent

Сторінка «Мої бронювання» використовується для створення нової оренди та перегляду історії бронювань. Користувач може обрати автомобіль, зазначити дати оренди, створити бронювання, а також переглянути активні та завершені оренди.

The screenshot shows the 'Мої бронювання' (My Bookings) page. At the top, there's a navigation bar with links to 'Головна', 'Каталог авто', 'Мої бронювання', and 'Профіль'. The user is logged in as 'Olivia Stone'. The main heading is 'Керуйте своїми бронюваннями'. Below this, there's a section for 'Нове бронювання' (New Booking) with a car selection dropdown, start and end dates, and a 'Підтвердити бронювання' (Confirm Booking) button. To the right, there's a 'Ваш статус' (Your Status) section showing a blue car and the next return date: '13 трав. 2026 р., 02:51'.

Рис. 5.6 Сторінка «Мої бронювання» користувача

Сторінка «Профіль» дозволяє користувачу редагувати власні дані, зокрема ім'я та email, змінювати пароль і переглядати коротку статистику щодо власних оренд.

The screenshot shows the 'Ваш профіль' (Your Profile) page. The user's name is 'Olivia Stone' and email is 'client@carrental.com'. There are three tabs: '2' (Усього оренд), '1' (Активні зараз), and 'USER' (Роль акаунта). The 'Керування акаунтом' (Account Management) section allows updating name and email. The 'Контактні дані' (Contact Details) section shows the current profile with fields for name and email. The 'Зміна пароля' (Change Password) section has fields for current, new, and confirm passwords.

Рис. 5.7 Сторінка профілю користувача

Для користувача з роллю ADMIN передбачено окрему адміністративну частину. Сторінка «Огляд» містить внутрішню операційну зведену інформацію про стан системи, кількість автомобілів, клієнтів та оренд.

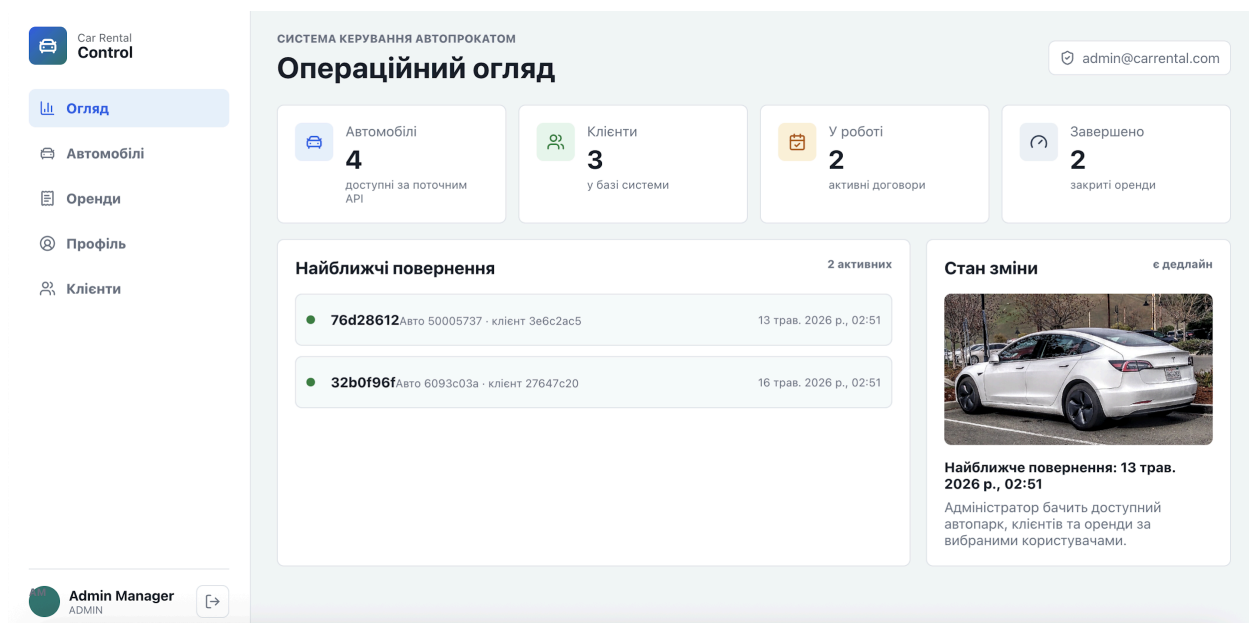


Рис.

5.8 Сторінка огляду адміністративної панелі

Сторінка «Автомобілі» використовується для керування автопарком. Адміністратор може переглядати список автомобілів, додавати нові машини, редагувати інформацію про них або видаляти записи.

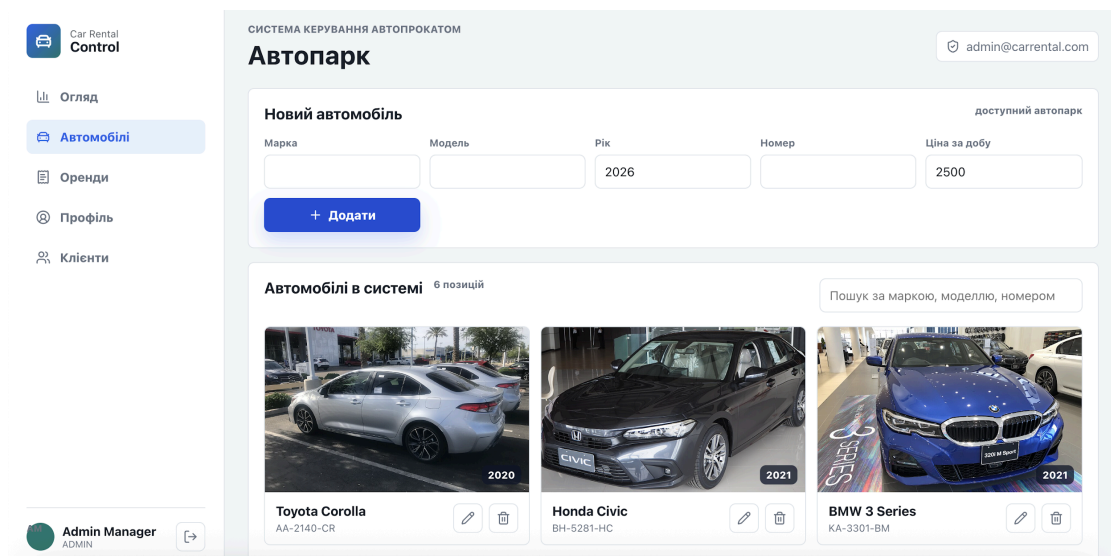


Рис. 5.9 Сторінка керування автомобілями

Сторінка «Оренди» призначена для створення та контролю оренд по користувачах. Адміністратор може переглядати активні й завершені оренди, змінювати їх статуси та контролювати виконання замовлень.

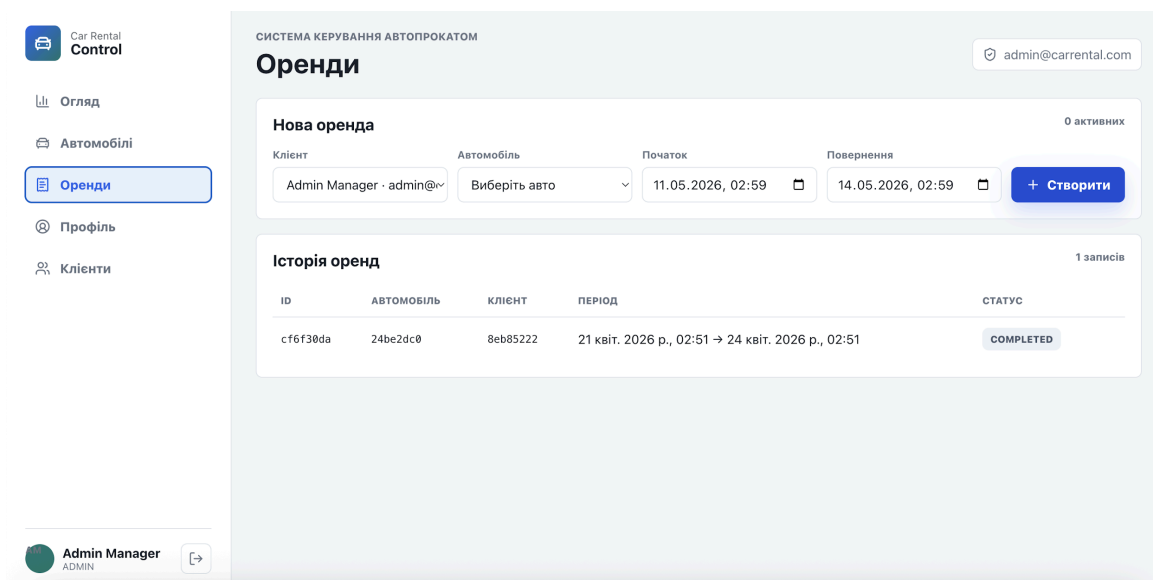


Рис. 5.10 Сторінка керування орендами

Сторінка «Клієнти» дозволяє адміністратору переглядати список користувачів і керувати ними. Це необхідно для контролю клієнтської бази та адміністрування доступу до системи.

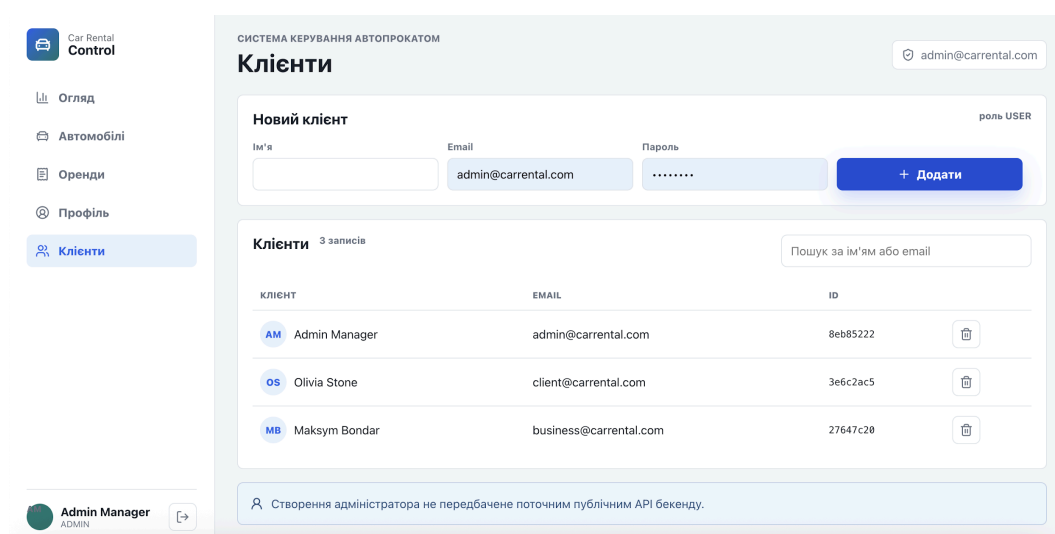


Рис. 5.11 Сторінка керування клієнтами

Сторінка «Профіль» в адміністративній частині використовується для перегляду та зміни особистих даних адміністратора, а також для зміни пароля.

The screenshot displays the 'Профіль' (Profile) page within the 'СИСТЕМА КЕРУВАННЯ АВТОПРОКАТОМ' (Car Rental Control System). The page is designed for an 'Admin Manager' user, identified by the email 'admin@carrental.com'.

Left Sidebar: Contains navigation links: 'Огляд' (Overview), 'Автомобілі' (Cars), 'Оренди' (Rentals), 'Профіль' (Profile - highlighted), and 'Клієнти' (Clients).

Main Content Area:

- Header:** 'СИСТЕМА КЕРУВАННЯ АВТОПРОКАТОМ' and 'Профіль'. A user status bar shows 'admin@carrental.com'.
- User Summary:** Displays 'Admin Manager' with email 'admin@carrental.com'. It includes three statistics: '1' (Усього оренд - Total Rentals), '0' (Активні зараз - Active Now), and 'ADMIN' (Роль акаунта - Account Role).
- Керування акаунтом (Account Management):** A section titled 'безпека та контакти' (security and contacts) with instructions to update email for confirmation.
- Контактні дані (Contact Information):** Labeled 'поточний профіль' (current profile). It shows 'Ім'я та прізвище' (Name and Surname) as 'Admin Manager' and 'Email' as 'admin@carrental.com'. A blue button 'Зберегти зміни' (Save changes) is at the bottom.
- Зміна пароля (Change Password):** Labeled 'захист акаунта' (account security). It includes fields for 'Поточний пароль' (Current password), 'Новий пароль' (New password), and 'Підтвердження нового пароля' (New password confirmation). A button 'Оновити пароль' (Update password) is at the bottom.

Bottom Bar: Shows the user's role 'Admin Manager' and 'ADMIN' with a logout icon.

Рис. 5.12 Сторінка профілю адміністратора

Таким чином, екранні форми web-застосунку DriveRent охоплюють основні сценарії роботи з системою: авторизацію, перегляд каталогу автомобілів, оформлення оренди, перегляд історії бронювань, редагування профілю та адміністрування даних системи. Поділ інтерфейсу за ролями USER і ADMIN дозволяє логічно розмежувати функції клієнта та адміністратора.

5.4 Тестування програмного забезпечення

Тестування web-застосунку DriveRent дозволить перевірити правильність роботи основних функцій системи, виявити потенційні помилки в програмному забезпеченні та переконатися, що розроблений вебзастосунок відповідає визначеним функціональним вимогам.

Особлива увага буде приділена тестуванню таких функцій вебзастосунку: аутентифікація користувачів, надання ролей користувачам, перегляд каталогу автомобілів для оренди, створення замовлення на оренду автомобіля, робота з профілем користувача та функції адміністрування.

Система буде протестована для трьох типів користувачів: гість, користувач з роллю USER та адміністратор з роллю ADMIN. Для гостей будуть протестовані сторінки реєстрації та входу. Для звичайних користувачів будуть протестовані такі сторінки: головна сторінка, каталог автомобілів для оренди, мої оренди та профіль користувача. Для адміністраторів будуть протестовані такі сторінки: огляд сервісу, автомобілі, оренди, клієнти та профіль адміністратора.

Основні напрямки тестування web-застосунку включатимуть:

- тестування реєстрації та автентифікації користувачів;
- тестування розмежування ролей між користувачами з ролями USER та ADMIN;
- тестування головної сторінки сервісу;
- тестування каталогу автомобілів напрокат;
- тестування відображення фотографій автомобілів, року випуску, номерних знаків та добової вартості оренди;
- тестування створення нової оренди користувачем;
- тестування перегляду як активних, так і завершених оренд користувачем;
- тестування редагування профілю користувача;
- тестування панелі адміністратора;
- тестування додавання, редагування та видалення автомобілів з каталогу;
- тестування управління орендами та клієнтами в системі.

Для тестування web-застосунку буде використовуватися функціональне тестування. Кожна основна функція системи буде тестуватися окремо відповідно до очікуваного результату тестування. Для кожного тестового сценарію будуть визначені дії користувача, очікуваний та фактичний результат тестування, а також статус тестування.

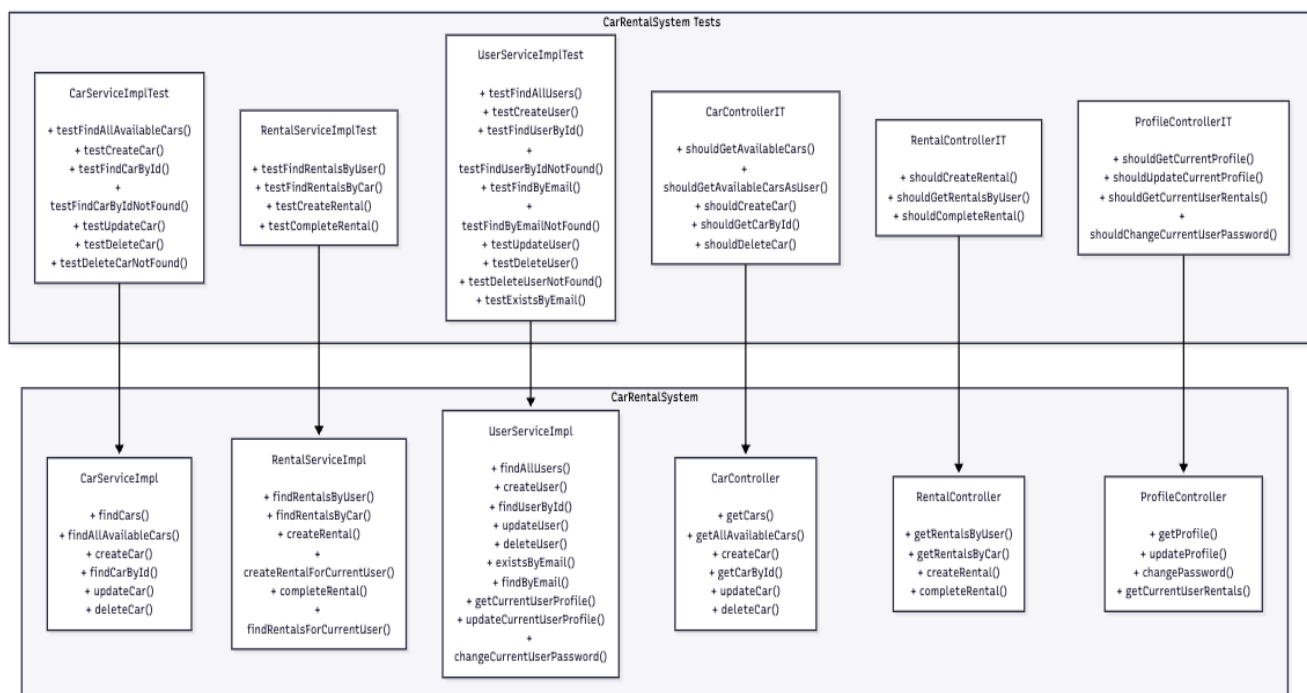


Рис. 5.13 Схема тестування web-застосунку DriveRent

5.5 Результати тестування

Результати тестування web-застосунку DriveRent наведено у таблиці 5.1. Для перевірки були обрані основні сценарії роботи системи, які охоплюють авторизацію, роботу з каталогом автомобілів, створення оренди, перегляд бронювань, редагування профілю та виконання адміністративних дій.

Тестування проводилося вручну шляхом виконання основних дій у web-інтерфейсі застосунку. Для кожного тестового сценарію було визначено очікуваний результат, після чого фактичну поведінку системи було порівняно з очікуваною.

Таблиця 5.1

Результати тестування

№	Сценарій тестування	Дії користувача	Очікуваний результат	Фактичний результат	Статус
1	Реєстрація нового користувача	Відкрити сторінку Register, заповнити форму та натиснути кнопку реєстрації	Новий акаунт створюється, користувач може перейти до авторизації	Відповідає очікуванню	Пройдено
2	Авторизація користувача	Ввести email і пароль на сторінці Login	Користувач входить у систему відповідно до своєї ролі	Відповідає очікуванню	Пройдено
3	Перевірка ролі USER	Авторизуватися як звичайний користувач	Відображаються сторінки Головна, Каталог авто, Мої бронювання, Профіль	Відповідає очікуванню	Пройдено
4	Перевірка ролі ADMIN	Авторизуватися як адміністратор	Відображаються сторінки Огляд, Автомобілі, Оренди, Клієнти, Профіль	Відповідає очікуванню	Пройдено
5	Перегляд головної сторінки	Перейти на сторінку Головна	Відображається hero-блок, швидкі дії, популярні авто та підказка щодо оренди	Відповідає очікуванню	Пройдено
6	Перегляд каталогу автомобілів	Відкрити сторінку Каталог авто	Відображається список автомобілів із фото, роком, номером і ціною за добу	Відповідає очікуванню	Пройдено

Результати тестування

№	Сценарій тестування	Дії користувача	Очікуваний результат	Фактичний результат	Статус
7	Перехід до бронювання	Обрати автомобіль у каталозі та перейти до оформлення оренди	Відкривається форма створення бронювання для вибраного авто	Відповідає очікуванню	Пройдено
8	Створення нової оренди	Обрати автомобіль, вказати дати оренди та підтвердити створення	Оренда створюється та з'являється у списку бронювань користувача	Відповідає очікуванню	Пройдено
9	Перегляд активних бронювань	Відкрити сторінку Мої бронювання	Користувач бачить список активних бронювань	Відповідає очікуванню	Пройдено
10	Перегляд завершених бронювань	Відкрити історію бронювань	Користувач бачить завершені оренди	Відповідає очікуванню	Пройдено
11	Редагування профілю користувача	Змінити ім'я або email у профілі	Дані користувача оновлюються	Відповідає очікуванню	Пройдено
12	Зміна пароля	Ввести старий і новий пароль у профілі	Пароль користувача змінюється після підтвердження	Відповідає очікуванню	Пройдено
13	Перегляд адміністративного огляду	Авторизуватися як ADMIN і відкрити сторінку Огляд	Відображається внутрішня зведена інформація по системі	Відповідає очікуванню	Пройдено

Результати тестування

№	Сценарій тестування	Дії користувача	Очікуваний результат	Фактичний результат	Статус
14	Редагування автомобіля	Відкрити запис автомобіля та змінити його дані	Дані автомобіля оновлюються у списку	Відповідає очікуванню	Пройдено
15	Видалення автомобіля	Обрати автомобіль і виконати видалення	Автомобіль видаляється зі списку автопарку	Відповідає очікуванню	Пройдено
16	Перегляд оренд адміністратором	Відкрити сторінку Оренди	Адміністратор бачить список активних і завершених оренд	Відповідає очікуванню	Пройдено
17	Керування орендою	Змінити дані або статус оренди	Інформація про оренду оновлюється	Відповідає очікуванню	Пройдено
18	Перегляд клієнтів	Відкрити сторінку Клієнти	Відображається список зареєстрованих користувачів	Відповідає очікуванню	Пройдено
19	Вихід із системи	Натиснути кнопку виходу	Користувач виходить із системи та повертається до сторінки входу	Відповідає очікуванню	Пройдено

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено web-застосунок DriveRent для оренди автомобілів, який забезпечує централізоване керування автопарком, бронюваннями, тарифами, статусами автомобілів та замовленнями користувачів.

1. Досліджено предметну галузь оренди автомобілів та визначено основні проблеми організації цього процесу. Встановлено, що ручне ведення даних про автопарк, тарифи, бронювання та замовлення ускладнює контроль доступності автомобілів, може призводити до дублювання бронювань і неактуальності статусів транспортних засобів.
2. Проаналізовано існуючі програмні засоби для автоматизації оренди автомобілів, зокрема RentSyst, HQ Rental Software, Rent Centric та Turo.. За результатами аналізу визначено їх переваги та недоліки, а також обґрунтовано доцільність розробки власного web-застосунку DriveRent, орієнтованого на базові потреби сервісу оренди автомобілів.
3. Сформульовано функціональні та нефункціональні вимоги до web-застосунку DriveRent. До основних функцій системи віднесено авторизацію користувачів за ролями, перегляд каталогу автомобілів, створення оренди, перегляд бронювань, редагування профілю користувача та адміністрування автомобілів, клієнтів і оренд.
4. Обґрунтовано вибір засобів реалізації програмного забезпечення. Для створення серверної частини використано Java та Spring Boot, для клієнтської частини — React і TypeScript, для збереження даних — PostgreSQL, а для організації середовища та підтримки розробки — Docker, Git, GitHub та IntelliJ IDEA.
5. Спроектовано логічну структуру web-застосунку DriveRent та виконано моделювання системи за допомогою UML-діаграм. Реалізовано серверну

частину web-застосунку, яка відповідає за обробку запитів, роботу з користувачами, автомобілями, орендами, тарифами та статусами.

6. Реалізовано клієнтську частину web-застосунку DriveRent. Для гостя передбачено сторінки входу та реєстрації, для користувача — головну сторінку, каталог авто, сторінку «Мої бронювання» та профіль, для адміністратора — сторінки «Огляд», «Автомобілі», «Оренди», «Клієнти» та «Профіль».
7. Організовано збереження даних у базі даних PostgreSQL. У системі передбачено збереження інформації про користувачів, автомобілі, оренди, тарифи, статуси та інші дані, необхідні для функціонування web-застосунку.
8. Проведено тестування основних функцій web-застосунку DriveRent. Перевірено сценарії роботи гостя, користувача та адміністратора. За результатами тестування встановлено, що основні функції системи працюють відповідно до очікуваної логіки.

Отже, поставлену мету кваліфікаційної роботи досягнуто: розроблено web-застосунок DriveRent, який спрощує процес оренди автомобілів за рахунок централізованого керування автопарком, орендами, тарифами, статусами та замовленнями користувачів.

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

1. Полянська Д.С., Трінтіна Н.А. Розробка web-застосунку "DriveRent" для сервісу оренди авто. Матеріали II Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. – К.: ДУІКТ, 2025, С.505-507.

2. Полянська Д.С., Трінтіна Н.А. Особливості автоматизації процесу оренди автомобілів у web-застосунку. Матеріали Всеукраїнської науково-технічної конференції «Технології цифрового розвитку: програмні системи та децентралізація», 1-3 травня 2026 року, Київ, ДУІКТ. Збірник тез. (подано до друку).

ПЕРЕЛІК ПОСИЛАНЬ

1. Маценко О. М., Маланчук Т. В., Попов В. С., Півень В. С., Скрипка Є. О. Економіко-правові основи використання каршерингових бізнес-моделей в Україні. *Механізм регулювання економіки*. 2020. № 1. С. 75–83. URL: https://mer.fem.sumdu.edu.ua/content/articles/issue_44/Oleksandr_M_Matsenko_Tetiana_Malanchuk_Vladyslav_S_Popov_Vladyslav_S_Piven_Evhenyi_O_SkrypkaEconomic_and_Legal_Bases_of_U.pdf
2. Turoń K. Car-Sharing Systems in Smart Cities: A Review of the Most Important Issues Related to the Functioning of the Systems in Light of the Scientific Research. *Smart Cities*. 2023. Vol. 6, no. 2. P. 796–808. URL: <https://doi.org/10.3390/smartcities6020038>
3. Bruglieri M., Pisacane O., Colorni A. A survey on emergent trends in the optimization of car-sharing systems. *International Transactions in Operational Research*. 2023. URL: <https://doi.org/10.1111/itor.13315>
4. Currie C. S. M., Fowler J. W., Kotiadis K., Monks T. Price optimization for round trip car sharing. *European Journal of Operational Research*. 2025. URL: <https://doi.org/10.1016/j.ejor.2025.04.034>
5. Turoń K., Kubik A., Chen F. Carsharing Vehicle Fleet Selection from the Frequent User's Point of View. *Energies*. 2022. Vol. 15, no. 17. Article 6166. URL: <https://doi.org/10.3390/en15176166>
6. Ogbiti J. T. Development of a Web-Based Car Rental Management System. *Science World Journal*. 2024. URL: <https://scienceworldjournal.org/article/download/24033/15094>
7. Albino M., Payumo J. Development of Car Rental Management System with Scheduling Algorithm. *SSRN Electronic Journal*. 2021. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3849830
8. Про автомобільний транспорт : Закон України від 05.04.2001 р. № 2344-III. URL: <https://zakon.rada.gov.ua/laws/show/2344-14>

9. КВЕД-2010. Клас 77.11. Надання в оренду автомобілів і легкових автотранспортних засобів. Державна служба статистики України. URL: https://kved.ukrstat.gov.ua/KVED2010/77/KVED10_77_11.html
10. Ринок каршерінгу в Україні. *InVenture*. 2023. URL: <https://inventure.com.ua/uk/analytics/investments/rinok-karsheringu-v-ukrayini>
11. Car Rental Software – Management & Booking. RentSyst. URL: <https://rentsyst.com/>.
12. Car Rental Software – Vehicle Rental Management & Booking Software. HQ Rental Software. URL: <https://hqrentalsoftware.com/>.
13. Car Rental Software. Rent Centric. URL: <https://o.rentcentric.com/> (дата звернення: 05.03.2026).
14. How Turo works. Turo car rental marketplace. URL: <https://turo.com/us/en/car-rental/united-states/how-turo-works>.
15. Spring Boot Reference Documentation. Spring. URL: <https://docs.spring.io/spring-boot/documentation.html>.
16. PostgreSQL Documentation: What Is PostgreSQL? PostgreSQL Global Development Group. URL: <https://o.postgresql.org/docs/current/intro-what-is.html> (дата звернення: 17.03.2026).
17. Describing the UI. React Documentation. URL: <https://react.dev/learn/describing-the-ui>.
18. The TypeScript Handbook. TypeScript Documentation. URL: <https://o.typescriptlang.org/docs/handbook/intro.html>.
19. Docker overview. Docker Documentation. URL: <https://docs.docker.com/get-started/docker-overview/>.
20. IntelliJ IDEA Documentation. JetBrains. URL: <https://o.jetbrains.com/help/idea/getting-started.html>.
21. Git Documentation. Git. URL: <https://git-scm.com/doc>.
22. About GitHub and Git. GitHub Docs. URL: <https://docs.github.com/en/get-started/start-your-journey/about-github-and-git>.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ



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



Розробка web-застосунку "DriveRent" для сервісу оренди авто

Виконала студентка 4 курсу
групи ПД-42

Полянська Дар'я Сергіївна
Керівник роботи

канд.техн.наук, доц. Трінтіна Наталія Альбертівна

Київ – 2026

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: спрощення процесу оренди автомобілів шляхом розробки web-застосунку Driverent, що забезпечує перегляд каталогу авто, оформлення оренди та керування автопарком.

Об'єкт дослідження: процес організації оренди автомобілів, що включає вибір транспортного засобу, оформлення оренди, облік клієнтів і контроль стану автопарку.

Предмет дослідження: засоби, методи та технології автоматизації процесу оренди автомобілів у web-застосунку Driverent.

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Провести аналіз предметної галузі оренди автомобілів, визначити основні проблеми ручного ведення автопарку, бронювань, тарифів і замовлень.
2. Здійснити огляд та аналіз існуючих програмних засобів для автоматизації оренди автомобілів, визначити їх переваги та недоліки.
3. Провести огляд засобів для розробки web-застосунку DriveRent.
4. Сформувати функціональні та нефункціональні вимоги до web-застосунку для оренди автомобілів.
5. Спроектувати логічну структуру web-застосунку DriveRent та побудувати UML-діаграми для моделювання системи.
6. Розробити серверну та клієнтську частини web-застосунку DriveRent для керування автомобілями, орендами, клієнтами та бронюваннями.
7. Провести тестування web-застосунку DriveRent та проаналізувати результати перевірки основних функцій системи.

3

АНАЛІЗ АНАЛОГІВ

Показник	RentSyst	HQ Rental Software	Rent Centric	Turo	Driverent
Основне призначення	CRM для оренди авто	Система керування прокатом	ПЗ для прокату та каршерингу	Маркетплейс оренди авто	Web-застосунок для оренди автомобілів
Адміністративна панель	+	+	+	лише базове керування акаунтом	+
Робота з тарифами	+	+	+	ціна задається власником авто	+
Каталог автомобілів	+	+	+	+	+
Керування автопарком	+	+	+	керування лише власними авто	+
Контроль оренд	+	+	+	контроль через заявки власників	+
Керування клієнтами	+	+	+	перегляд даних лише в межах оренди	+
Профіль користувача	+	+	+	+	+
Локальне розгортання проєкту	-	-	-	-	+
Можливість адаптації під власну логіку	налаштування в межах готової CRM	налаштування під бізнес-процеси прокату	налаштування через параметри системи	відсутня зміна логіки платформи	повна зміна коду під вимоги проєкту
Відсутність залежності від комерційної підписки	-	-	-	-	+
Простота для невеликого сервісу	потребує налаштування CRM	потребує впровадження в бізнес	потребує освоєння модулів	проста для клієнта, не для сервісу	проста структура для малого сервісу
Орієнтація на бізнес автопрокату	+	+	+	-	+

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги до web-застосунку DriveRent:

- реєстрація та авторизація користувачів (ролі клієнт, адміністратор);
- перегляд категорій автомобілів;
- фільтрація автомобілів;
- перегляд інформації про вибраний автомобілі;
- створення бронювання;
- перевірка доступності автомобіля;
- зберігання інформації замовників автомобілів;
- перегляд власних бронювань;
- керування відділом автомобілів (додавання, видалення, модифікація);
- керування відділом замовлень (замовлення, видалення, модифікація).

Нефункціональні вимоги до web-застосунку DriveRent:

- час завантаження сторінок не повинен перевищувати 3 секунди;
- арі виконує запити в межах 1-2 секунди;
- середовище повинне забезпечувати роботу не менше 50 користувачів;
- інтерфейс повинен правильно відображатися на персональному комп'ютері, планшеті та смартфоні;
- доступ до адмін-панелі повинен бути тільки для користувачів із роллю адміністратора;
- дані повинні зберігатися у базі даних;
- код застосунку повинен мати модульну структуру;

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ



React



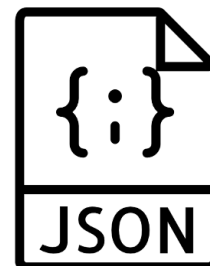
Java



docker



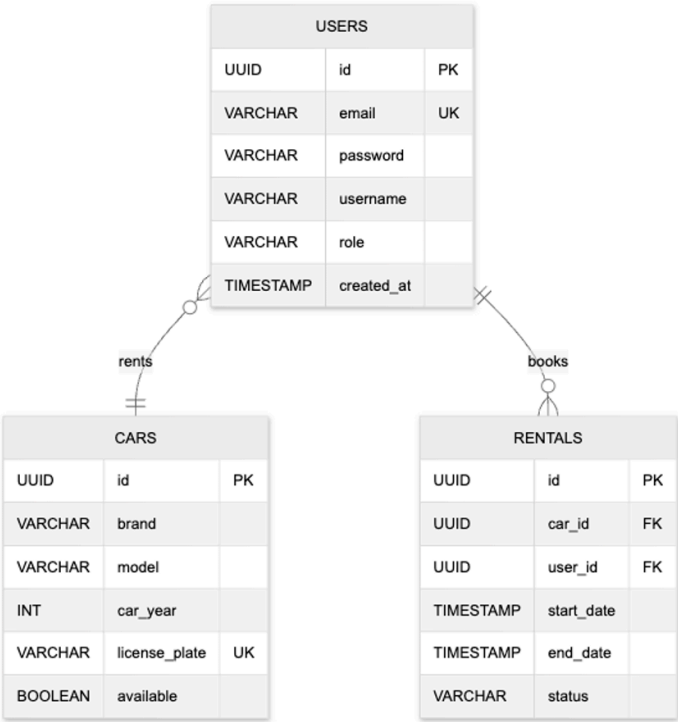
PostgreSQL



MavenTM

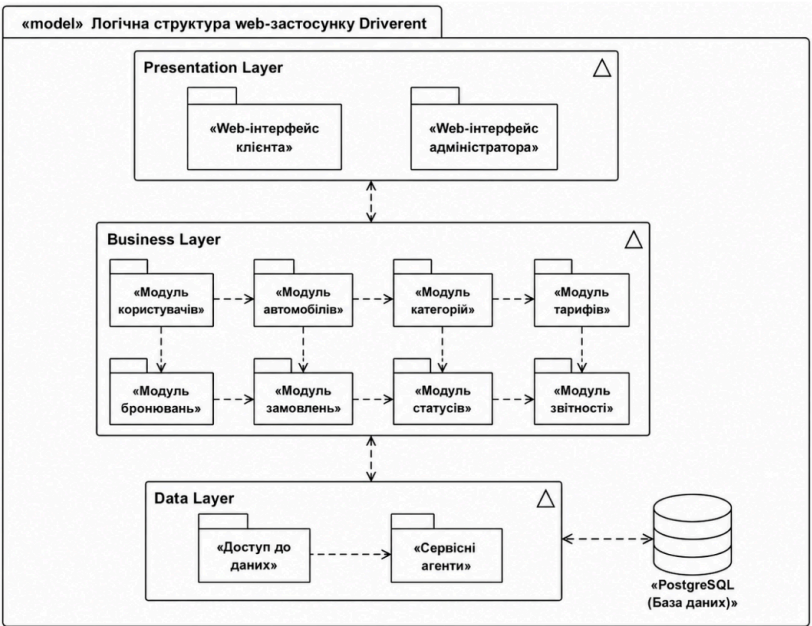
6

СХЕМА БАЗИ ДАНИХ



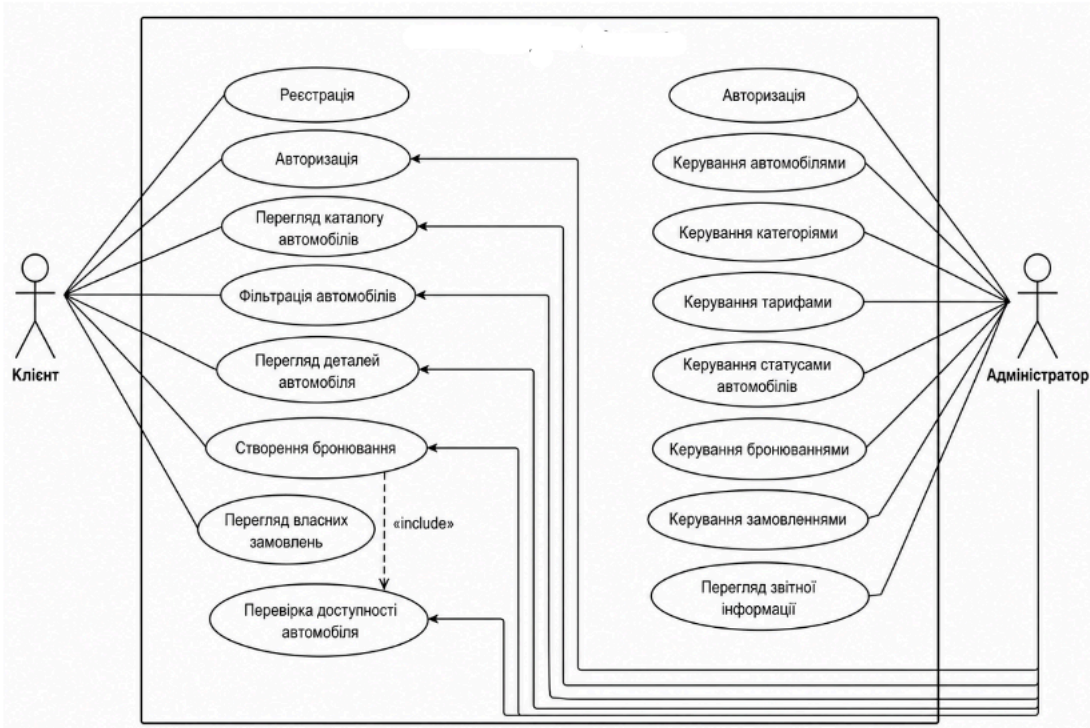
7

ЛОГІЧНА АРХІТЕКТУРА ВЕБ-ДОДАТКУ DRIVERENT



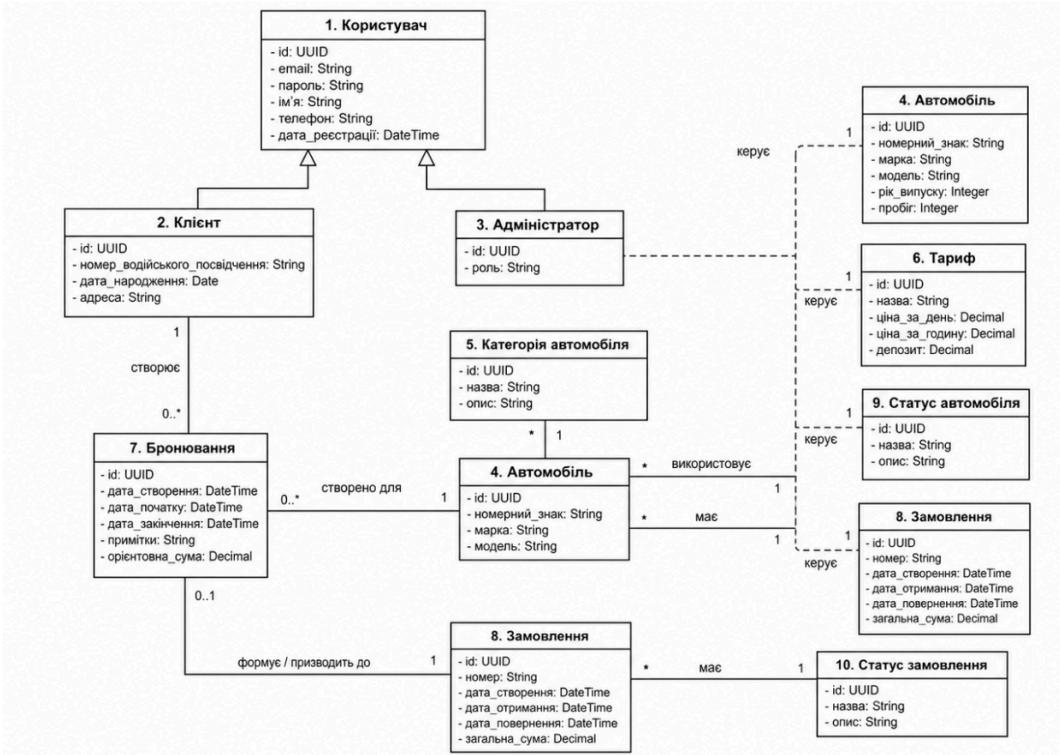
8

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



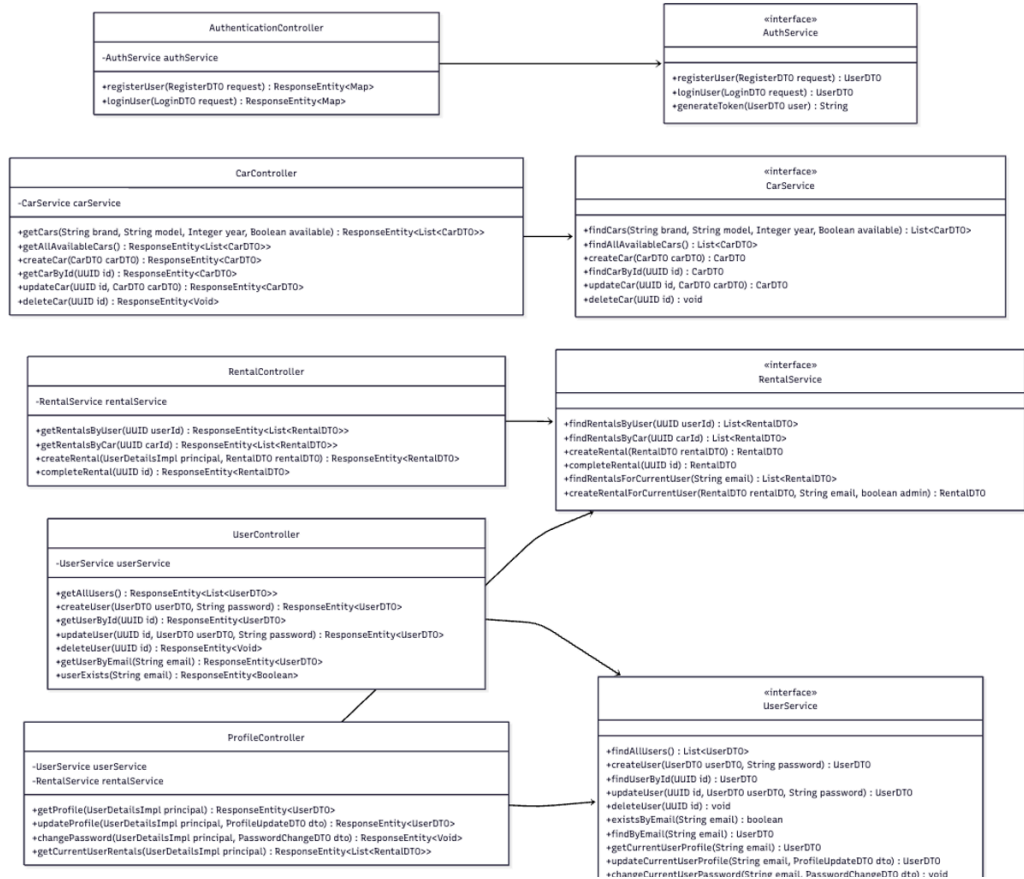
9

ДІАГРАМА ПРЕДМЕТНОЇ ГАЛУЗІ

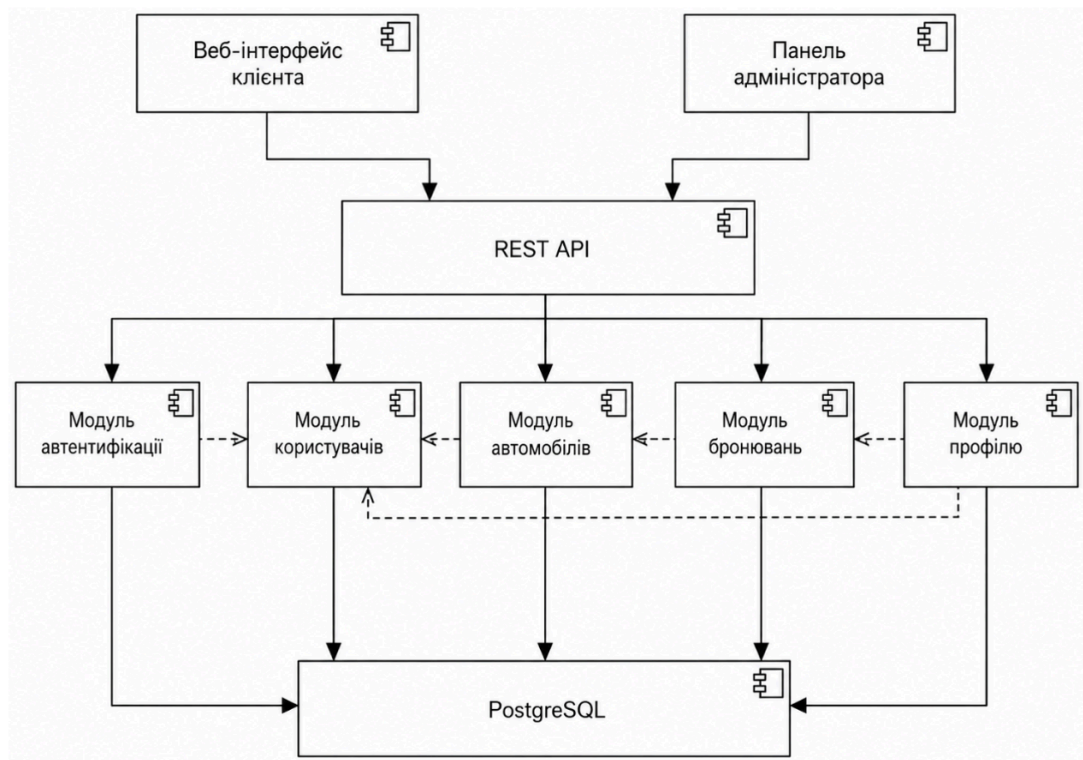


10

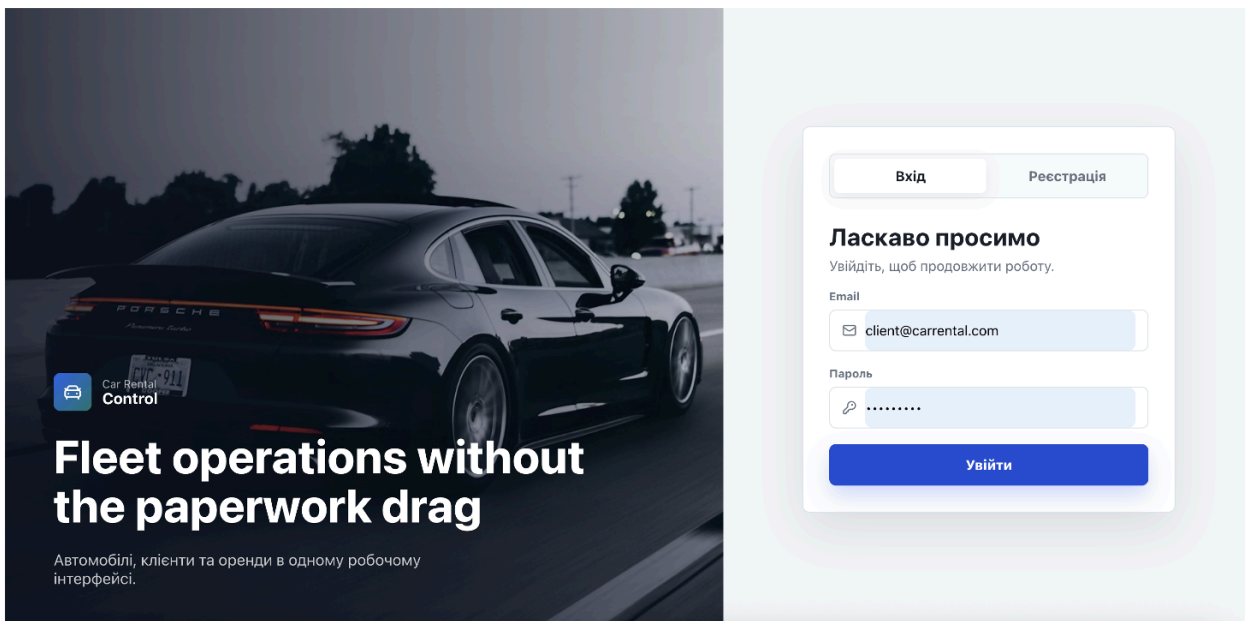
ДІАГРАМА КЛАСІВ ОСНОВНОЇ ЛОГІКИ



ДІАГРАМА КОМПОНЕНТІВ

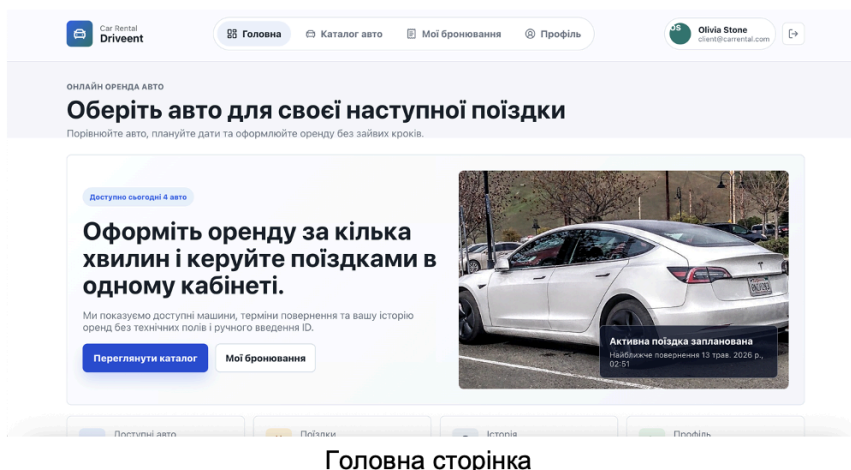


ЕКРАННІ ФОРМИ – ВХІД

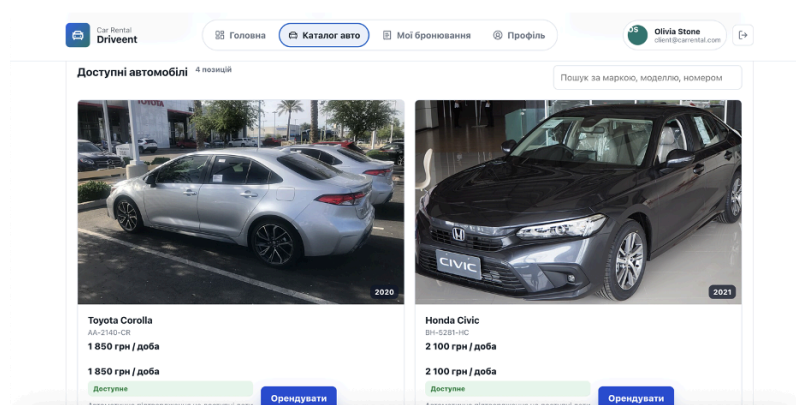


13

ЕКРАННІ ФОРМИ



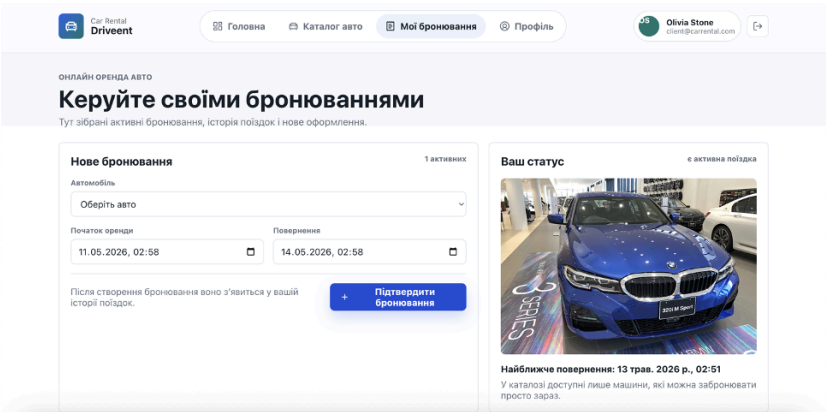
Головна сторінка



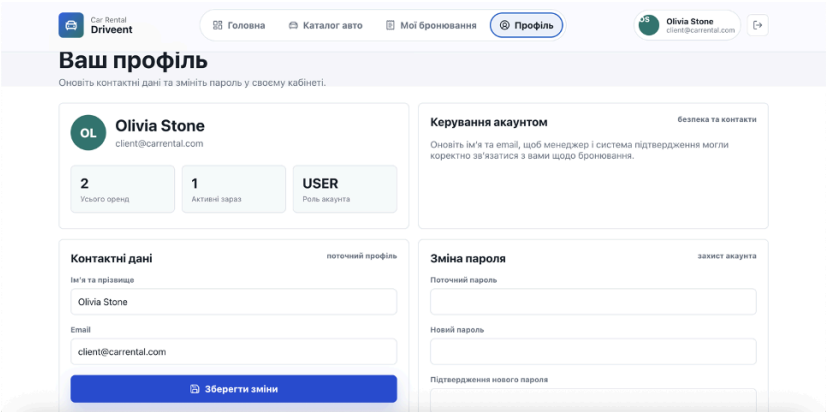
Каталог авто

14

ЕКРАННІ ФОРМИ



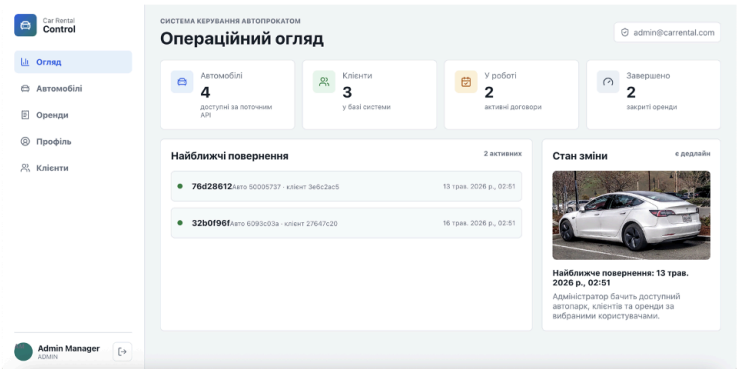
Сторінка бронювання



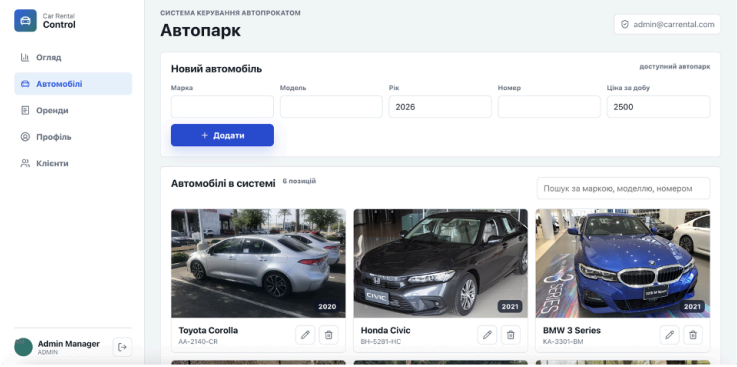
Профіль користувача

15

ЕКРАННІ ФОРМИ



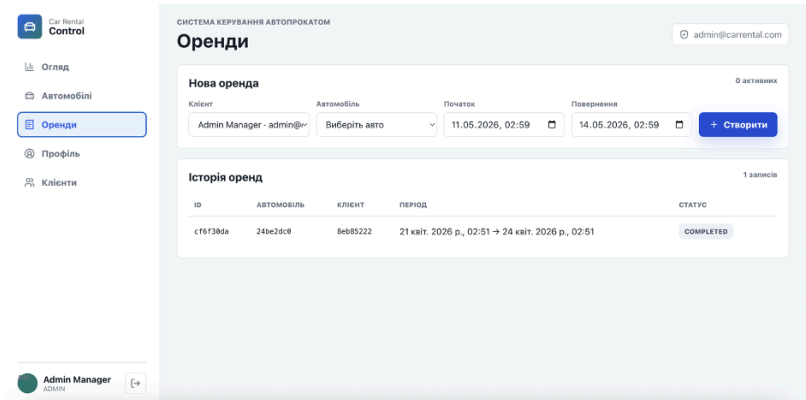
Панель адміністратора



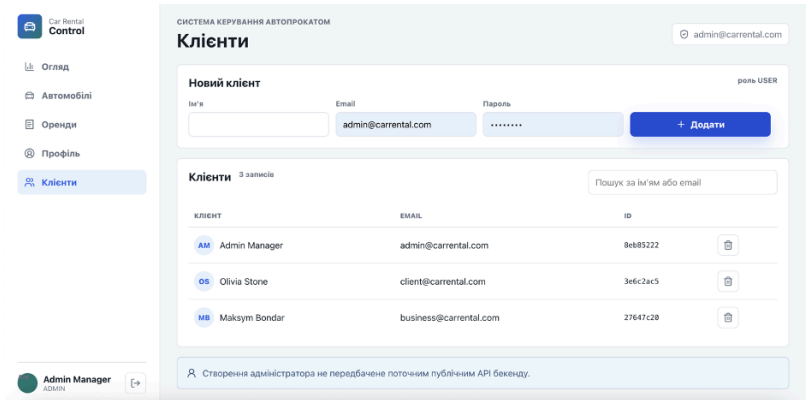
Сторінка автопарку

16

ЕКРАННІ ФОРМИ



Керування орендами



Керування клієнтами

17

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Тези доповідей:

- 1. Полянська Д.С., Трінтіна Н.А. Розробка web-застосунку "DriveRent" для сервісу оренди авто. Матеріали II Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. – К.: ДУІКТ, 2025, С.505-507.
- 2. Полянська Д.С., Трінтіна Н.А. Особливості автоматизації процесу оренди автомобілів у web-застосунку. Матеріали Всеукраїнської науково-технічної конференції «Технології цифрового розвитку: програмні системи та децентралізація», 1-3 травня 2026 року, Київ, ДУІКТ. Збірник тез. (подано до друку).

ВИСНОВКИ

1. Проведено аналіз предметної галузі оренди автомобілів, визначено основні проблеми ручного ведення автопарку, зокрема неактуальність статусів авто, дублювання бронювань, складність контролю тарифів і замовлень.
2. Здійснено огляд та аналіз існуючих програмних засобів для автоматизації оренди автомобілів, зокрема RentSyst, HQ Rental Software, Rent Centric і Turo, визначено їх переваги, недоліки та обмеження для невеликого сервісу.
3. Проведено огляд засобів для розробки web-застосунку DriveRent, обґрунтовано використання Java, Spring Boot, PostgreSQL, React, TypeScript і Docker.
4. Сформовано функціональні та нефункціональні вимоги до web-застосунку для оренди автомобілів, зокрема вимоги до каталогу авто, бронювання, ролей користувачів, швидкодії, безпеки та підтримки системи.
5. Спроектовано логічну структуру web-застосунку DriveRent та побудовано UML-діаграми для моделювання системи, зокрема діаграми прецедентів, доменної моделі, діяльності, класів, компонентів і розгортання.
6. Розроблено серверну та клієнтську частини web-застосунку DriveRent для керування автомобілями, орендами, клієнтами, бронюваннями, профілями користувачів і адміністративною панеллю.
7. Проведено тестування web-застосунку DriveRent, перевірено авторизацію, роботу каталогу, створення оренди, перегляд бронювань, редагування профілю та адміністративне керування системою.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

CarRentalApplication.java

```
@SpringBootApplication
public class CarRentalApplication {
    public static void main(String[] args) {
        SpringApplication.run(CarRentalApplication.class,
args);
    }
}
```

SecurityConfig.java

```
@Configuration
@EnableWebSecurity
@EnableMethodSecurity
@RequiredArgsConstructor
public class SecurityConfig {

    private final JwtFilter jwtFilter;
    private final AuthenticationProvider
authenticationProvider;

    @Bean
    public SecurityFilterChain
securityFilterChain(HttpSecurity http) throws Exception
{
    http
        .csrf(AbstractHttpConfigurer::disable)
        .authorizeHttpRequests(auth -> auth
            .requestMatchers("/login", "/register",
"/v3/api-docs/**", "/swagger-ui/**").permitAll()
            .anyRequest().authenticated()
        )

        .authenticationProvider(authenticationProvider)
        .addFilterBefore(jwtFilter,
UsernamePasswordAuthenticationFilter.class);

    return http.build();
}
}
```

AuthenticationController.java

```
@RestController
@RequiredArgsConstructor
public class AuthenticationController {

    private final AuthService authService;

    @PostMapping("/register")
    public ResponseEntity<Map<String, Object>>
registerUser(@RequestBody RegisterDTO request) {
        UserDTO user = authService.registerUser(request);
        String token = authService.generateToken(user);
        return ResponseEntity.ok(Map.of("message", "User
registered", "token", token, "user", user));
    }

    @PostMapping("/login")
```

```
    public ResponseEntity<Map<String, Object>>
loginUser(@RequestBody LoginDTO request) {
        UserDTO user = authService.loginUser(request);
        String token = authService.generateToken(user);
        return ResponseEntity.ok(Map.of("message",
>Login successful", "token", token, "user", user));
    }
}
```

CarController.java

```
@RestController
@RequestMapping("/api/v1/cars")
@RequiredArgsConstructor
public class CarController {

    private final CarService carService;

    @GetMapping
    @PreAuthorize("hasRole('USER') or
hasRole('ADMIN')")
    public ResponseEntity<List<CarDTO>> getCars(
        @RequestParam(required = false) String brand,
        @RequestParam(required = false) String model,
        @RequestParam(required = false) Integer year,
        @RequestParam(required = false) Boolean
available
    ) {
        return
ResponseEntity.ok(carService.findCars(brand, model,
year, available));
    }

    @GetMapping("/available")
    @PreAuthorize("hasRole('USER') or
hasRole('ADMIN')")
    public ResponseEntity<List<CarDTO>>
getAllAvailableCars() {
        return
ResponseEntity.ok(carService.findAllAvailableCars());
    }
}
```

RentalController.java

```
@RestController
@RequestMapping("/api/v1/rentals")
@RequiredArgsConstructor
public class RentalController {

    private final RentalService rentalService;

    @PostMapping
    @PreAuthorize("hasRole('USER') or
hasRole('ADMIN')")
    public ResponseEntity<RentalDTO> createRental(
        @AuthenticationPrincipal UserDetailsImpl
principal,
        @RequestBody RentalDTO rentalDTO
```

```

    ) {
        boolean admin =
principal.getUser().getRole().name().equals("ADMIN");
        return new ResponseEntity<>(

rentalService.createRentalForCurrentUser(rentalDTO,
principal.getUser().getEmail(), admin),
        HttpStatus.CREATED
    );
}

```

```

    @PutMapping("/{id}/complete")
    @PreAuthorize("hasRole('USER') or
hasRole('ADMIN')")
    public ResponseEntity<RentalDTO>
completeRental(@PathVariable UUID id) {
        return
ResponseEntity.ok(rentalService.completeRental(id));
    }
}

```

ProfileController.java

```

@RestController
@RequestMapping("/api/v1/profile")
@RequiredArgsConstructor
public class ProfileController {

    private final UserService userService;
    private final RentalService rentalService;

    @GetMapping
    public ResponseEntity<UserDTO>
getProfile(@AuthenticationPrincipal UserDetailsImpl
principal) {
        return
ResponseEntity.ok(userService.getCurrentUserProfile(pr
incipal.getUser().getEmail()));
    }

    @PutMapping
    public ResponseEntity<UserDTO> updateProfile(
        @AuthenticationPrincipal UserDetailsImpl
principal,
        @Valid @RequestBody ProfileUpdateDTO
profileUpdateDTO
    ) {
        return ResponseEntity.ok(

userService.updateCurrentUserProfile(principal.getUser(
).getEmail(), profileUpdateDTO)
        );
    }
}

```

CarEntity.java

```

@Entity
@Table(name = "cars")
@Builder(toBuilder = true)
@NoArgsConstructor
@AllArgsConstructor
@Getter
public class CarEntity {

```

```

@Id
@GeneratedValue(strategy = GenerationType.UUID)
private UUID id;

```

```

private String brand;
private String model;

```

```

@Column(name = "car_year", nullable = false)
private int year;

```

```

private String licensePlate;

```

```

@Column(name = "daily_rate", nullable = false,
precision = 10, scale = 2)
private BigDecimal dailyRate;

```

```

private boolean available;
}

```

CarServiceImpl.java

```

@Service
@RequiredArgsConstructor
@Transactional
public class CarServiceImpl implements CarService {

    private final CarRepository carRepository;

    @Override
    public List<CarDTO> findAllAvailableCars() {
        return
carRepository.findByAvailableTrue().stream()
            .map(this::mapToDTO)
            .collect(Collectors.toList());
    }

    @Override
    public CarDTO createCar(CarDTO carDTO) {
        CarEntity car = CarEntity.builder()
            .brand(carDTO.getBrand())
            .model(carDTO.getModel())
            .year(carDTO.getYear())
            .licensePlate(carDTO.getLicensePlate())
            .dailyRate(carDTO.getDailyRate())
            .available(carDTO.getAvailable() == null ||
carDTO.getAvailable())
            .build();

        return mapToDTO(carRepository.save(car));
    }
}

```

RentalServiceImpl.java

```

@Service
@RequiredArgsConstructor
@Transactional
public class RentalServiceImpl implements
RentalService {

```

```

    private final RentalRepository rentalRepository;
    private final CarRepository carRepository;

```

```

private final UserRepository userRepository;

@Override
public RentalDTO
createRentalForCurrentUser(RentalDTO rentalDTO,
String email, boolean admin) {
    UserEntity user = admin
        ?
userRepository.findById(rentalDTO.getUserId()).orElse
Throw()
    :
userRepository.findById(email).orElseThrow();

    CarEntity car =
carRepository.findById(rentalDTO.getCarId()).orElseTh
row();

    if (!car.isAvailable()) {
        throw new BusinessValidationException("Car is
not available for rental");
    }

    carRepository.save(car.toBuilder().available(false).build(
));

    RentalEntity rental = RentalEntity.builder()
        .user(user)
        .car(car)
        .startDate(rentalDTO.getStartDate())
        .endDate(rentalDTO.getEndDate())
        .status(RentalStatus.ACTIVE)
        .build();

    return mapToDTO(rentalRepository.save(rental));
}
}

```

DataSeeder.java

```

@Component
@Profile("!test")
@RequiredArgsConstructor
public class DataSeeder implements
CommandLineRunner {

    private final UserRepository userRepository;
    private final CarRepository carRepository;
    private final RentalRepository rentalRepository;
    private final PasswordEncoder passwordEncoder;

    @Override
    public void run(String... args) {
        if (userRepository.count() > 0 ||
carRepository.count() > 0) {
            return;
        }

        userRepository.save(UserEntity.builder()
            .email("admin@carrental.com")
            .username("Admin Manager")

            .password(passwordEncoder.encode("admin123"))

```

```

        .role(Role.ADMIN)
        .createdAt(LocalDateTime.now())
        .build());
    }
}

```

App.tsx

```

function App() {
    const {session} = useAuth()
    const [view, setView] = useState<View>('dashboard')
    const [selectedCarId, setSelectedCarId] = useState(o

    if (!session) {
        return <AuthPage />
    }

    return(
        <Shell activeView={view} onViewChange={setView}>
            {view==='dashboard'&&<DashboardPage
onBookCar={setSelectedCarId} onViewChange={setVie
w}/>}
            {view==='cars'&&<CarsPage
onBookCar={setSelectedCarId}/>}
            {view==='rentals'&&<RentalsPage
initialCarId={selectedCarId} onViewChange={setView}o
nSelectionConsumed={()=>setSelectedCarId(o)}/>}
            {view==='profile'&&<ProfilePage />}
            {view==='users'&&<UsersPage />}
        </Shell>
    )
}

```

Shell.tsx

```

export function
Shell( {activeView,onViewChange,children}:ShellProps)
{
    const {logout,session} = useAuth()
    const role = session?.role ?? 'UNKNOWN'

    if (role==='USER') {
        return(
            <div className="user-shell">
                <header className="user-header">
                    <nav className="user-nav">
                        <button
onClick={()=>onViewChange('dashboard')}>Головна</
button>
                        <button onClick={()=>onViewChange('cars')}>Каталог
авто</button>
                        <button onClick={()=>onViewChange('rentals')}>Мої
бронювання</button>
                        <button
onClick={()=>onViewChange('profile')}>Профіль</butt
on>
                    </nav>
                </header>
                <main>{children}</main>
            </div>
        )
    }
}

```

```

    return<div
className="app-shell">{children}</div>
    }

```

CarsPage.tsx

```

export function
CarsPage( {onBookCar}:CarsPageProps){
  const {session}=useAuth()
  const[cars,setCars]=useState<Car[]>([])

  useEffect(()=>{
    if(!session)return
    carsApi.getAvailable(session.token).then(setCars)
  },[session])

  return(
<section className="catalog-grid">
  {cars.map((car)=>(
<article className="catalog-card"key={car.id}>
<img
alt={` ${car.brand} ${car.model} `}src={getCarImage(car)}
/>
<h3>{car.brand} {car.model}</h3>
<p>{car.licensePlate}</p>
<strong>{formatCurrency(car.dailyRate)} /доба</strong>
<button
onClick={()=>onBookCar(car.id)}>Орендувати</button>
</article>
))}
</section>
  )
}

```

RentalsPage.tsx

```

export function
RentalsPage( {initialCarId}:RentalsPageProps){
  const {session}=useAuth()
  const[cars,setCars]=useState<Car[]>([])
  const[rentals,setRentals]=useState<Rental[]>([])

  const[form,setForm]=useState(defaultForm(session?.user
.id,initialCarId))

  async function
handleSubmit(event:FormEvent<HTMLFormElement>)
{
  event.preventDefault()
  if(!session)return

  await rentalsApi.create(session.token,{
    userId:form.userId||session.user.id,
    carId:form.carId,
    startDate:toApiDateTime(form.startDate),
    endDate:toApiDateTime(form.endDate),
    status:'ACTIVE',
  })
}

ProfilePage.tsx

```

```

export function ProfilePage(){
  const {session,updateSessionUser}=useAuth()

```

```

const[profileForm,setProfileForm]=useState( {username:
o,email:o})

```

```

  async function
handleProfileSubmit(event:FormEvent<HTMLFormEle
ment>){
    event.preventDefault()
    if(!session)return

    const updated=await
    profileApi.update(session.token,profileForm)
    updateSessionUser(updated)
  }

```

```

  return(
<form onSubmit={handleProfileSubmit}>
<input value={profileForm.username}/>
<input value={profileForm.email}/>
<button type="submit">Зберегти зміни</button>
</form>
  )
}

```

client.ts

```

export const carsApi={

  getAll:(token:string)=>request<Car[]>(o/api/v1/cars',{to
ken}),

  getAvailable:(token:string)=>request<Car[]>(o/api/v1/ca
rs/available',{token}),

  getById:(token:string,id:string)=>request<Car>(`/api/v1/
cars/${id}`, {token}),

  export const profileApi={

    get:(token:string)=>request<User>(o/api/v1/profile',{tok
en}),

    update:(token:string,payload:ProfileUpdatePayload)=>

    request<User>(o/api/v1/profile,{method:'PUT',token,bo
dy:payload}),
  }
}

index.css

```

```

.user-shell{
  min-height:100vh;
  background:linear-gradient(180deg, #f4f7fb 0,
#ffffff 220px);
}

.catalog-grid{
  display:grid;
  grid-template-columns:repeat(2,minmax(0,1fr));
  gap:18px;
}

```

```
.catalog-card{  
border:1px solid #dbe4ee;  
border-radius:8px;  
background: #ffffff;  
overflow:hidden;  
}
```

```
.price-tag{  
color:var(--ink);  
font-size:16px;  
font-weight:800;  
}
```