

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПРОКАТУ АВТОМОБІЛІВ
CAR RENTAL SOFTWARE

Виконав(ла): студент(ка) 4 курсу, групи КНТ-111

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

АЛЕКСЄЄНКО Д.В.

(ПРИЗВИЩЕ та ініціали)

Керівник СТЕПАНЕНКО О.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент ПОЛЯКОВ О.М.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Факультет КНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 “ ” 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

АЛЕКСЄЄНКО Данііла Вадимовича
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення прокату автомобілів. Car Rental Software.
 керівник проєкту (роботи) к.т.н., доцент, СТЕПАНЕНКО Олександр Олексійович,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
 затверджені наказом закладу вищої освіти від “28” квітня 2025 року № 209
2. Строк подання студентом проєкту (роботи) 02 червня 2025 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області та постаовка завдання. 2. Модель та методи програмного забезпечення. 3. Опис програмного забезпечення. 4. Аналіз отриманих результатів.
5. Перелік графічного матеріалу (з з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	СТЕПАНЕНКО О.О., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст.викладач		

7. Дата видачі завдання « 14 » квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій	2 тиждень	Розділ 2
	розробки.		
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження	5 тиждень	Розділ 4
	програмного забезпечення.		
7	Оформлення пояснювальної записки та документів	6 тиждень	Додатки
	до неї.		
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

_____ Данііл АЛЕКСЄЄНКО
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ Олександр СТЕПАНЕНКО
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 136 с., 9 табл., 36 рис., 3 дод., 29 джерел.

ПРОКАТ АВТОМОБІЛІВ, ОДНОСТОРІНКОВИЙ ВЕБЗАСТОСУНОК, МІКРОЛЕНДІНГ, АДМІНІСТРАТИВНА ПАНЕЛЬ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Об'єкт дослідження - процес розроблення односторінкових вебдодатків для автоматизації послуг прокату автомобілів.

Предмет дослідження - методи та програмні засоби оптимізації взаємодії користувачів з інтерфейсами вебсистем прокату автомобілів та інтеграції клієнтської частини з адміністративними функціями управління.

Мета роботи - підвищення ефективності обробки запитів користувачів на оренду автомобілів шляхом розробки оптимізованого односторінкового мікролендінгу та інтегрованої адміністративної панелі з використанням сучасних вебтехнологій.

Матеріали, методи та технічні засоби: PostgreSQL, Redis, Python, FastAPI, React, TypeScript, Redux, Docker, Kubernetes, архітектурні патерни (Flux, MVC, Repository), CI/CD, GraphQL, REST API, мікросервісна архітектура.

Результати. Розроблено функціональний вебзастосунок для сервісу прокату автомобілів.

Висновки. Створене програмне забезпечення дозволяє автоматизувати бізнес-процеси прокату автомобілів, знизити навантаження на адміністративний персонал, підвищити зручність користування сервісом.

Галузь використання - вебсервіси для автопрокатних компаній, онлайн-системи бронювання транспорту, цифрові рішення для малого та середнього бізнесу в транспортній галузі.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor:
136 pages, 9 tables, 36 figures, 3 appendixes, 29 sources.

CAR RENTAL, SINGLE-PAGE WEB APPLICATION,
MICROLANDING PAGE, ADMIN PANEL, SOFTWARE.

Object of study is the process of developing single-page web applications for the automation of car rental services.

Subject of study are methods and software tools for optimizing user interaction with car rental web system interfaces and integrating the client-side with administrative management functions.

The purpose of the work is to improve the efficiency of processing car rental requests by developing an optimized single-page microlanding page and an integrated admin panel using modern web technologies.

Materials, methods, and tools: PostgreSQL database, Redis cache, Python programming language with FastAPI framework, React and TypeScript for the frontend, Redux for state management, Docker containers, Kubernetes orchestration, architectural patterns (Flux, MVC, Repository), CI/CD pipelines, GraphQL and REST API, and a microservice architecture.

Results. A functional web application has been developed for a car rental service that enables user registration, car search and booking, admin management of vehicles and orders, and integrated analytics.

Conclusions. The developed software automates core business processes of a car rental service, reduces administrative workload, and enhances user convenience and system scalability.

Field of use is the development of web services for car rental companies, online vehicle booking platforms, and digital solutions for small and medium-sized businesses in the transportation sector.

ЗМІСТ

	С.
Вступ.....	8
1 Аналіз предметної області та поставнока задачі	10
1.1 Аналіз існуючих рішень у сфері прокату автомобілів.....	10
1.2 Технологічні аспекти розробки односторінкових вебдодатків.....	14
1.3 Формування вимог до програмного забезпечення	25
1.4 Висновки за розділом 1	35
2 Модель та методи програмного забезпечення	36
2.1 Загальна архітектура системи.....	36
2.2 Вибір мов програмування	41
2.3 Проєктування інтерфейсу користувача	48
2.4 Проєктування адміністративної панелі.....	51
2.5 Висновки за розділом 2	55
3 Опис програмного забезпечення	56
3.1 Розробка бази даних.....	56
3.2 Реалізація програмного забезпечення.....	61
3.3 Висновки за розділом 3	73
4 Аналіз отриманих результатів	75
4.1 Інструкції користувача	75
4.2 Інструкції адміністратора.....	85
4.3 Оптимізація програмного забезпечення	92
4.4 Висновки за розділом 4	95
Висновки	97
Перелік джерел посилення	98
Додаток А Технічне завдання	101
Додаток Б Фрагмент тексту програми	105
Додаток В Слайди презентації.....	128

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

SPA	– Single Page Application;
API	– Application Programming Interface;
UX/UI	– User Experience/User Interface;
СУБД	– Система управління базами даних;
CTA	– Call To Action;
SEO	– Search Engine Optimization;
MVC	– Model-View-Controller;
ACID	– Atomicity, Consistency, Isolation, Durability;
CI/CD	– Continuous Integration/Continuous Deployment;
JWT	– JSON Web Tokens.

ВСТУП

У сучасному світі цифрової трансформації бізнес-процесів автоматизація набуває критичного значення для конкурентоспроможності компаній усіх галузей. Особливо це стосується сфери надання послуг прокату автомобілів, де якість та швидкість обслуговування клієнтів безпосередньо впливає на лояльність та задоволеність споживачів. За статистикою, компанії, що впровадили цифрові рішення для управління прокатом автомобілів, збільшили кількість клієнтів на 30-40% та зменшили адміністративні витрати на 25%. Односторінкові вебдодатки (SPA) з адміністративними панелями стають оптимальним рішенням, що дозволяє забезпечити персоналізований користувацький досвід та ефективне управління ресурсами автопрокату [1].

Важливим аспектом розвитку програмних рішень у сфері прокату автомобілів є вибір оптимальних технологій, що забезпечують високу продуктивність, масштабованість та безпеку даних. Сучасні технології розробки вебдодатків, такі як React для фронтенду, Python з FastAPI для бекенду та PostgreSQL для системи управління базами даних, надають потужний інструментарій для створення комплексних рішень, здатних задовольнити зростаючі потреби бізнесу. Актуальність впровадження таких технологічних рішень підтверджується зростанням ринку прокату автомобілів в Україні на 15-20% щороку і збільшенням попиту на цифрові сервіси для бронювання та управління автопарком [2].

У контексті розвитку галузі IT та програмних рішень для бізнесу, оптимізація процесів взаємодії з користувачами через вебінтерфейси стає ключовим фактором забезпечення конкурентних переваг. Автоматизація процесів бронювання, управління автопарком та взаємодії з клієнтами вимагає розробки спеціалізованих програмних рішень, що поєднують сучасні підходи до проєктування користувацьких інтерфейсів та ефективної організації бекенд-архітектури [3].

Об'єкт дослідження: процес розроблення односторінкових вебдодатків для автоматизації послуг прокату автомобілів.

Предмет дослідження: методи та програмні засоби оптимізації взаємодії користувачів з інтерфейсами вебсистем прокату автомобілів та інтеграції клієнтської частини з адміністративними функціями управління.

Мета роботи: підвищення ефективності обробки запитів користувачів на оренду автомобілів шляхом розробки оптимізованого односторінкового мікролендінгу та інтегрованої адміністративної панелі з використанням сучасних вебтехнологій.

Завдання:

- проаналізувати існуючі архітектурні підходи до розробки односторінкових вебдодатків та визначити оптимальні паттерни проєктування для систем прокату автомобілів;
- розробити компонентну архітектуру програмного рішення з чітким розмежуванням клієнтської та серверної логіки;
- спроектувати та реалізувати реактивні компоненти користувацького інтерфейсу для вибору та бронювання автомобілів;
- розробити RESTful API з використанням Python та FastAPI для ефективної комунікації між клієнтською частиною та базою даних;
- створити модульну адміністративну панель з компонентами аналітики та управління автопарком;
- оптимізувати швидкість завантаження та обробки даних на основі результатів тестування продуктивності.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАВНОКА ЗАДАЧІ

1.1 Аналіз існуючих рішень у сфері прокату автомобілів

Ринок програмного забезпечення для прокату автомобілів демонструє стабільне зростання протягом останніх років, що зумовлено підвищенням попиту на послуги оренди транспортних засобів та необхідністю автоматизації відповідних бізнес-процесів. За даними аналітичної компанії Mordor Intelligence, глобальний ринок програмного забезпечення для управління автопарком досягне \$45 мільярдів до 2027 року з річним темпом зростання 10,8% [1]. Основними факторами, що стимулюють розвиток ринку, є збільшення кількості транспортних засобів у комерційних автопарках, потреба в зниженні операційних витрат та оптимізації використання ресурсів, а також зростаючий попит на мобільні рішення для бронювання автомобілів.

Аналіз існуючих систем прокату автомобілів дозволяє виділити декілька ключових категорій програмних рішень: системи глобальних провайдерів (Hertz, Avis, Enterprise), платформи локальних компаній прокату та агрегатори орендних пропозицій (Kayak, Expedia, RentalCars). Кожна з цих категорій має свої особливості реалізації та функціональні можливості. Глобальні провайдери зазвичай пропонують комплексні рішення з власними мобільними додатками та інтеграцією з глобальними системами дистрибуції (GDS); локальні компанії частіше використовують готові програмні рішення з обмеженим функціоналом або замовляють розробку під власні потреби; агрегатори ж фокусуються на інтеграції з численними партнерами та пропонують порівняльний інтерфейс для споживачів [4].

У процесі аналізу було детально вивчено п'ять популярних вебсервісів прокату автомобілів: Hertz (hertz.com), Enterprise (enterprise.com), Europcar (europcar.com), Turo (turo.com) та Getaround (getaround.com). Результати порівняльного аналізу їх функціональних можливостей представлені в табл. 1.1.

Таблиця 1.1 - Порівняльний аналіз функціональних можливостей вебсервісів прокату автомобілів

Функціональна можливість	Hertz	Enterprise	Europcar	Turo	Getaround
Онлайн-бронювання	Так	Так	Так	Так	Так
Мобільний додаток	Так	Так	Так	Так	Так
Система рейтингів	Ні	Ні	Ні	Так	Так
Розширений пошук	Так	Так	Так	Так	Так
P2P модель прокату	Ні	Ні	Ні	Так	Так
Інтеграція з GPS	Так	Так	Так	Ні	Так
Безключовий доступ	Ні	Ні	Ні	Ні	Так
Функція порівняння авто	Так	Ні	Так	Так	Ні
Програма лояльності	Так	Так	Так	Ні	Ні
API для партнерів	Так	Так	Так	Так	Так

Специфіка українського ринку прокату автомобілів має свої особливості, зумовлені як економічним станом країни, так і рівнем автомобілізації населення. За даними Інституту досліджень авторинку, рівень автомобілізації в Україні залишається одним із найнижчих у Європі - 232 автомобілі на 1000 осіб, тоді як середньоєвропейський показник становить 610 автомобілів на 1000 осіб. Для досягнення середньоєвропейського показника у 610 авто на тисячу осіб, до України потрібно привезти ще 15 мільйонів автомобілів. Цей фактор створює значний потенціал для розвитку послуг автопрокату, особливо у великих містах та туристичних центрах.

Ринок прокату автомобілів в Україні знаходиться у стадії активного розвитку. За даними Асоціації автопрокатних компаній України, "За нашими підрахунками, ринок показує зростання на 15-20% щорічно. Це пов'язано в першу чергу з розвитком транспортної інфраструктури і відповідно зростанням пасажиропотоку в країні". Обсяг ринку прокату автомобілів в Україні оцінюється приблизно у 2 тисячі автомобілів, при цьому лідерами

ринку є компанії, які підтримують свій автопарк у належному стані та регулярно його оновлюють [2].

На українському ринку прокату автомобілів представлені як міжнародні, так і локальні компанії. Серед міжнародних лідерів присутні такі бренди як Sixt, Avis, Budget, Hertz, що пропонують стандартизовані послуги відповідно до своїх глобальних політик. Українські компанії, такі як BLS, Ukr-Prokat, NarsCars, RENTAL.UA, 7cars, активно конкурують з міжнародними гравцями, пропонуючи більш гнучкі умови та часто нижчі ціни (рис. 1.1).

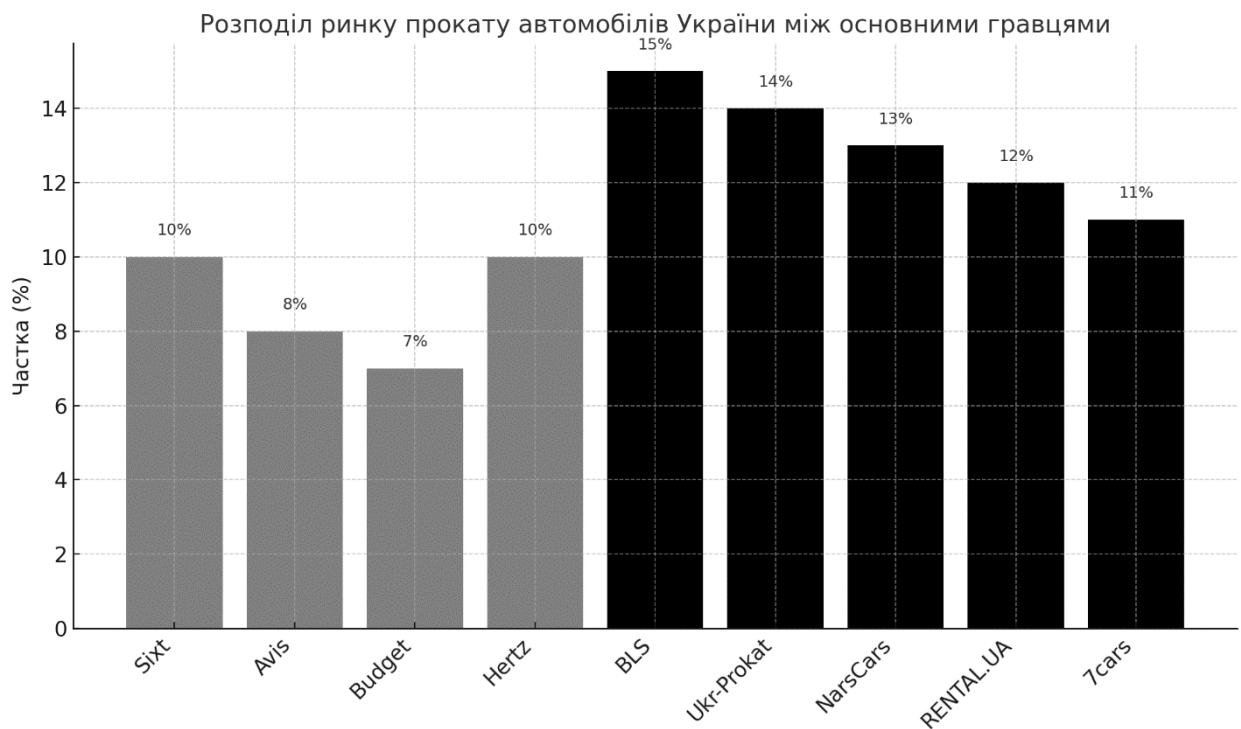


Рисунок 1.1 - Структура ринку прокату автомобілів України за основними гравцями

Аналіз клієнтської бази автопрокатних компаній в Україні показує, що близько 70% клієнтів автопрокату - це громадяни України, що спростовує поширену думку про те, що послугами прокату автомобілів користуються переважно іноземці. Цікавою тенденцією останніх років стало збільшення клієнтів локальних автопрокатних компаній, оскільки "Сервіс локальних

компаній не поступається сервісу міжнародних, а ціни дуже часто нижчі на 20-30%".

Щодо типових сценаріїв використання прокатних автомобілів в Україні, статистика свідчить, що 50% клієнтів замовляють автомобілі для бізнес-заходів, а інша половина – для особистих поїздок, таких як відвідування родичів або відпочинок. При цьому близько 90% орендованих автомобілів використовуються для поїздок між містами [2].

Характерною особливістю українського ринку прокату автомобілів є також активний розвиток супутніх послуг: оренда автомобілів з водієм, послуги доставки та підбору автомобіля, страхування, встановлення дитячих крісел, GPS-навігаторів тощо. Цінові категорії варіюються від економ-класу (від \$15-20 на день) до преміум-сегменту (від \$100-150 на день), що дозволяє задовольнити потреби різних категорій клієнтів (рис. 1.2).

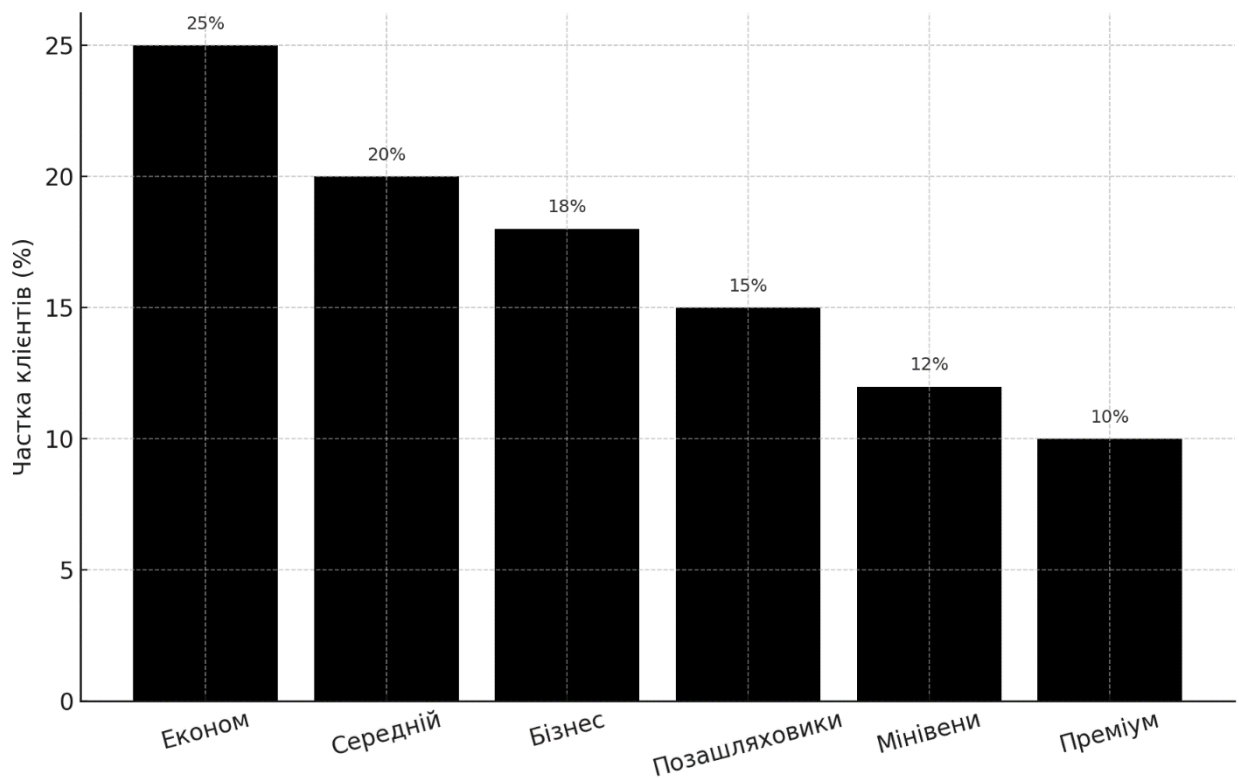


Рисунок 1.2 - Розподіл клієнтів за категоріями орендованих автомобілів

В контексті загального розвитку автомобільного ринку України, варто відзначити суттєве зростання продажів нових автомобілів, що свідчить про

поступове відновлення економіки та підвищення купівельної спроможності населення. За даними асоціації "Укравтопром", у першому кварталі 2024 року в Україні було зареєстровано 16 660 нових легкових автомобілів, що на 49% перевищує показник аналогічного періоду минулого року. Зростання продажів нових автомобілів корелює з розвитком ринку прокату автомобілів, оскільки обидва сегменти залежать від загальної економічної ситуації та споживчих настроїв.

Аналіз програмних рішень, що використовуються українськими компаніями прокату автомобілів, показує, що більшість з них застосовують готові системи управління автопарком та бронюванням, часто адаптовані під локальні потреби. Великі міжнародні компанії використовують власні глобальні системи, що забезпечують єдині стандарти обслуговування по всьому світу. Проте, як для міжнародних, так і для локальних гравців, актуальною залишається проблема інтеграції з іншими бізнес-системами, оптимізації для мобільних пристроїв та розширення аналітичних можливостей.

На основі проведеного аналізу українського ринку прокату автомобілів можна виділити наступні тенденції та перспективи розвитку: поступове зростання ринку на рівні 15-20% щорічно, збільшення частки локальних компаній на ринку, розширення спектру додаткових послуг, підвищення вимог клієнтів до якості обслуговування, зростання попиту на онлайн-сервіси бронювання та управління прокатом. Ці тенденції створюють сприятливі умови для розробки та впровадження сучасного програмного забезпечення з розширеним функціоналом, адаптованого під особливості українського ринку.

1.2 Технологічні аспекти розробки односторінкових вебдодатків

У контексті сучасного вебдизайну та розробки, односторінкові додатки (Single Page Applications або SPA) представляють собою передову

концепцію створення динамічних вебінтерфейсів, що забезпечують користувачам високий рівень інтерактивності та швидкості взаємодії з системою. За даними звіту Stack Overflow Developer Survey, понад 67% професійних розробників віддають перевагу архітектурі SPA для створення сучасних вебдодатків, що підтверджує актуальність цього підходу в контексті розробки програмного забезпечення для прокату автомобілів [3].

Односторінкові додатки фундаментально відрізняються від традиційних багатосторінкових вебсайтів своєю архітектурою та механізмом взаємодії з користувачем. Основний принцип SPA полягає у динамічному оновленні контенту на сторінці без необхідності повного перезавантаження. Це досягається завдяки використанню AJAX (Asynchronous JavaScript and XML) для асинхронного завантаження даних з сервера та маніпуляції DOM-структурою сторінки за допомогою JavaScript.

Архітектура SPA базується на концепції клієнт-серверної взаємодії, де клієнтська частина відповідає за відображення інтерфейсу та обробку користувацьких дій, а серверна частина надає API для доступу до даних та виконання бізнес-логіки. Така розподілена архітектура дозволяє оптимізувати навантаження на сервер та забезпечити високу швидкість відгуку інтерфейсу. Схематично архітектуру SPA можна представити наступним чином (рис. 1.3).

Односторінкові додатки мають низку переваг, що робить їх оптимальним вибором для розробки сучасних інтерактивних вебсервісів, зокрема у сфері прокату автомобілів:

- швидкість та плавність роботи за рахунок мінімізації запитів до сервера та відсутності перезавантаження сторінки;
- покращений користувацький досвід завдяки миттєвому відгуку інтерфейсу на дії користувача;
- можливість роботи в офлайн-режимі при реалізації відповідних механізмів кешування;

- зниження навантаження на сервер за рахунок передачі частини логіки на сторону клієнта;
- спрощення розробки мобільних додатків на основі тієї ж API-інфраструктури [4].

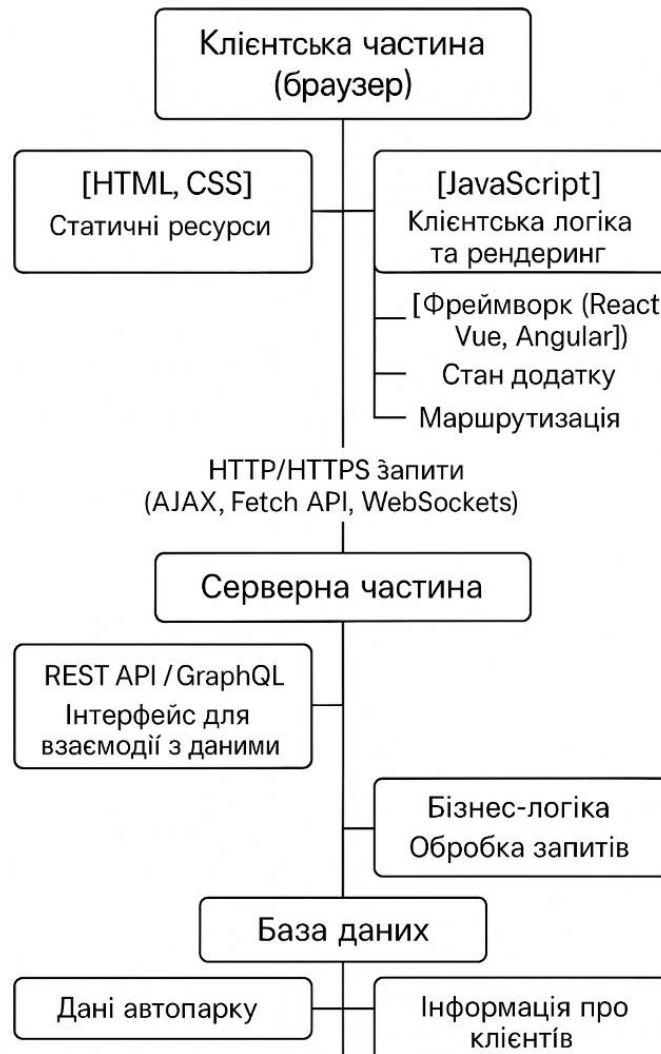


Рисунок 1.3 - Архітектура односторінкового додатку

Одночасно з цим, SPA мають певні обмеження та складнощі в реалізації, які необхідно враховувати при проєктуванні системи:

- початкове завантаження сторінки може займати більше часу через необхідність завантаження всього JavaScript-коду;
- складнощі з SEO (Search Engine Optimization) через динамічне генерування контенту;

- підвищені вимоги до обчислювальних ресурсів клієнтських пристроїв;
- необхідність реалізації механізмів захисту від XSS (Cross-Site Scripting) та CSRF (Cross-Site Request Forgery) атак.

Для компенсації цих обмежень у сучасній веброзробці застосовуються різноманітні техніки та інструменти, такі як серверний рендеринг (SSR), прогресивне завантаження, розділення коду (code splitting) та використання сервіс-воркерів для кешування.

Мікролендінг є особливим типом односторінкового додатку, орієнтованим на конкретну бізнес-мету та оптимізованим для конверсії відвідувачів у клієнтів. У контексті прокату автомобілів, мікролендінг фокусується на представленні послуг компанії, демонстрації автопарку та забезпеченні зручного процесу бронювання. За даними дослідження Nielsen Norman Group, ефективно спроектований мікролендінг може підвищити коефіцієнт конверсії на 30-40% порівняно з традиційними багатосторінковими сайтами [7].

Основними характеристиками мікролендінгу для прокату автомобілів є:

- чітка структура з послідовним представленням інформації (принцип "воронки конверсії");
- мінімалістичний дизайн з фокусом на ключових елементах та закликах до дії (CTA);
- оптимізація для швидкого завантаження та відображення на різних пристроях;
- інтеграція форм бронювання безпосередньо в сторінку без перенаправлення;
- візуальне представлення автопарку з детальною інформацією про кожен автомобіль.

Архітектурно мікролендінг для прокату автомобілів може бути представлений наступною структурою компонентів (рис. 1.4).

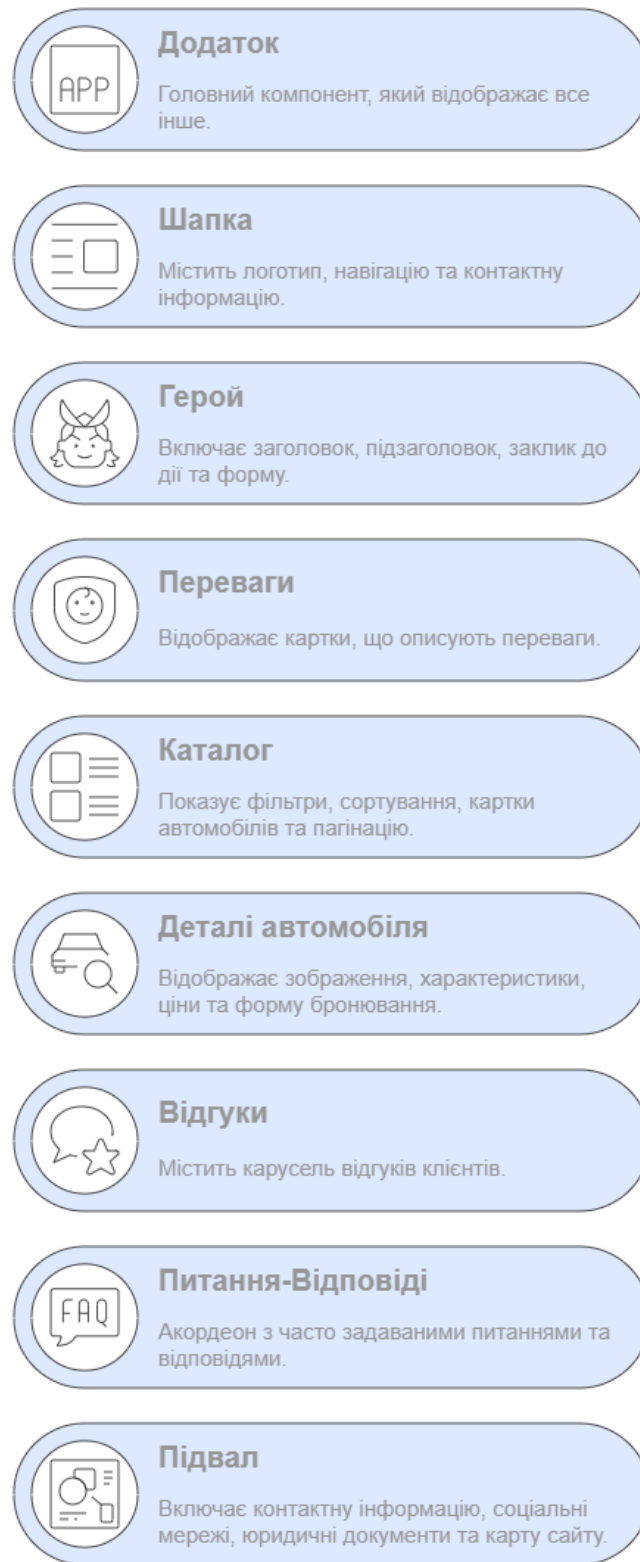


Рисунок 1.4 - Компонентна структура мікролендінгу для прокату автомобілів

При розробці мікролендінгу особлива увага приділяється оптимізації швидкості завантаження та продуктивності, оскільки ці фактори

безпосередньо впливають на коефіцієнт конверсії. Для досягнення оптимальної продуктивності застосовуються наступні техніки:

- ледаче завантаження (lazy loading) компонентів та зображень;
- мінімізація та стиснення статичних ресурсів (CSS, JavaScript, зображення);
- використання кешування на стороні клієнта;
- оптимізація критичного шляху рендерингу (Critical Rendering Path);
- застосування сучасних форматів зображень (WebP, AVIF) для зменшення їх розміру.

За результатами досліджень Google, збільшення часу завантаження сторінки з 1 до 3 секунд підвищує ймовірність відмови користувача на 32%. Тому оптимізація швидкості є критичним аспектом розробки мікролендінгу для прокату автомобілів [8].

Адміністративний інтерфейс є невід'ємною частиною системи прокату автомобілів, що забезпечує управління автопарком, обробку замовлень та аналітику. Існує декілька підходів до інтеграції адміністративних функцій з клієнтською частиною мікролендінгу:

- розробка окремого адміністративного додатку з власною архітектурою та інтерфейсом;
- інтеграція адміністративних компонентів у структуру основного додатку з контролем доступу;
- використання гібридного підходу з частковою інтеграцією адміністративних функцій.

Кожен з цих підходів має свої переваги та обмеження. Розробка окремого адміністративного додатку забезпечує чітке розмежування функціоналу та можливість незалежного масштабування, але ускладнює управління спільними ресурсами та може призвести до дублювання коду. Інтеграція адміністративних компонентів у структуру основного додатку спрощує розробку та обмін даними, але може негативно вплинути на

продуктивність клієнтської частини через збільшення розміру пакету JavaScript.

Для оптимального балансу між розмежуванням функціоналу та ефективністю розробки рекомендується використання мікрофронтенд-архітектури, що дозволяє розділити додаток на незалежні модулі з власним життєвим циклом, але з можливістю інтеграції в єдиний інтерфейс. У контексті мікролендінгу для прокату автомобілів, адміністративна частина може бути реалізована як окремий мікрофронтенд, що завантажується лише при необхідності та має власний маршрутизатор і стан.

Архітектура системи з використанням мікрофронтендів представлена на рисунку 1.5.

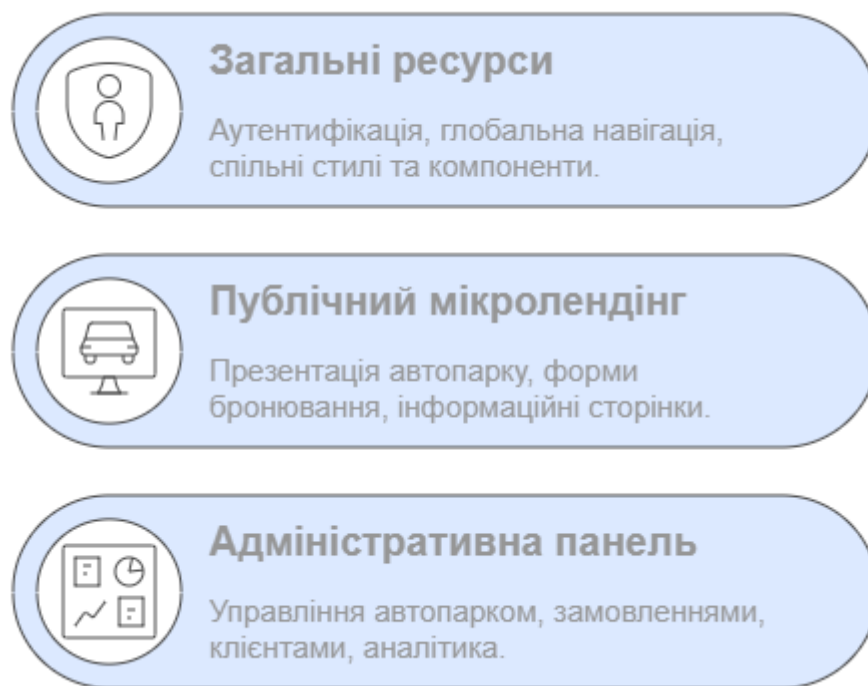


Рисунок 1.5 - Архітектура системи з використанням мікрофронтендів

При розробці адміністративного інтерфейсу для системи прокату автомобілів важливо забезпечити наступні функціональні можливості:

- управління автопарком (додавання, редагування, видалення автомобілів та їх характеристик);

- обробка замовлень (перегляд, підтвердження, відхилення, редагування деталей);
- управління клієнтською базою (перегляд інформації про клієнтів, історії замовлень, статистики);
- аналітика та звітність (аналіз ефективності використання автопарку, фінансові показники, тренди);
- управління контентом мікролендінгу (редагування текстів, зображень, акційних пропозицій).

Для ефективної реалізації цих функцій рекомендується використання спеціалізованих компонентів та бібліотек, таких як data-grid для відображення табличних даних, форми з валідацією для редагування інформації, графіки та діаграми для візуалізації аналітичних даних. Серед популярних бібліотек для реалізації адміністративних інтерфейсів можна виділити Material-UI, Ant Design, React Admin, які надають готові компоненти та шаблони для побудови адміністративних панелей [9].

Розвиток технологій та методологій розробки вебдодатків призвів до формування ряду сучасних тенденцій та практик, які варто враховувати при розробці односторінкового додатку для прокату автомобілів:

- прогресивні вебдодатки (PWA), що поєднують переваги вебсайтів та нативних мобільних додатків, забезпечуючи можливість роботи офлайн, швидке завантаження та доступ до нативних API пристрою;
- server-side rendering (SSR) та static site generation (SSG) для покращення SEO та швидкості початкового завантаження сторінки;
- використання GraphQL замість традиційного REST API для оптимізації запитів до сервера та зменшення кількості мережових запитів;
- архітектура micro-frontends для розділення додатку на незалежні модулі, що розробляються та розгортаються окремо;
- застосування web components для створення перевикористовуваних компонентів, що працюють незалежно від фреймворку;

- використання WebAssembly для оптимізації обчислювально-інтенсивних операцій, таких як обробка зображень або складні розрахунки;
- впровадження TypeScript для статичної типізації та підвищення надійності коду.

Також важливо відзначити зростаючу роль інструментів для автоматизації процесу розробки та розгортання SPA, таких як:

- системи збірки (Webpack, Vite, Parcel), що оптимізують процес трансформації та пакування ресурсів;
- системи контейнеризації (Docker) та оркестрації (Kubernetes) для спрощення розгортання та масштабування;
- CI/CD (Continuous Integration/Continuous Deployment) для автоматизації тестування та розгортання;
- системи моніторингу та аналітики (Google Analytics, Sentry, Datadog) для відстеження продуктивності та виявлення помилок.

Використання цих інструментів та практик дозволяє забезпечити високу якість, надійність та ефективність розробки односторінкового додатку для прокату автомобілів.

При розробці складних односторінкових додатків, таких як система прокату автомобілів з адміністративною панеллю, важливо використовувати відповідні архітектурні патерни для забезпечення масштабованості, тестованості та підтримки коду. Найбільш поширеними архітектурними патернами для SPA є MVC (Model-View-Controller), MVVM (Model-View-ViewModel) та Flux/Redux.

MVC розділяє додаток на три основні компоненти: модель (дані та бізнес-логіка), представлення (інтерфейс користувача) та контролер (обробка дій користувача). Цей патерн був широко використовуваний у класичних вебдодатках, але має певні обмеження при роботі з односторінковими додатками через складність підтримки двонаправленого зв'язку між компонентами.

MVVM є еволюцією MVC, що краще адаптована для роботи з декларативними фреймворками, такими як Angular та Vue.js. У цьому патерні ViewModel виступає посередником між моделлю та представленням, забезпечуючи двосторонню прив'язку даних та інкапсуляцію логіки представлення. Це спрощує розробку та тестування, оскільки представлення стає пасивним споживачем даних з ViewModel.

Flux/Redux представляє односпрямований потік даних, де стан додатку зберігається в централізованому сховищі (store), а зміни відбуваються через відправлення дій (actions) та їх обробку редукторами (reducers). Цей патерн особливо ефективний для складних додатків з багатьма компонентами, що взаємодіють між собою, оскільки забезпечує передбачуваний стан додатку та спрощує відлагодження (рис. 1.6).

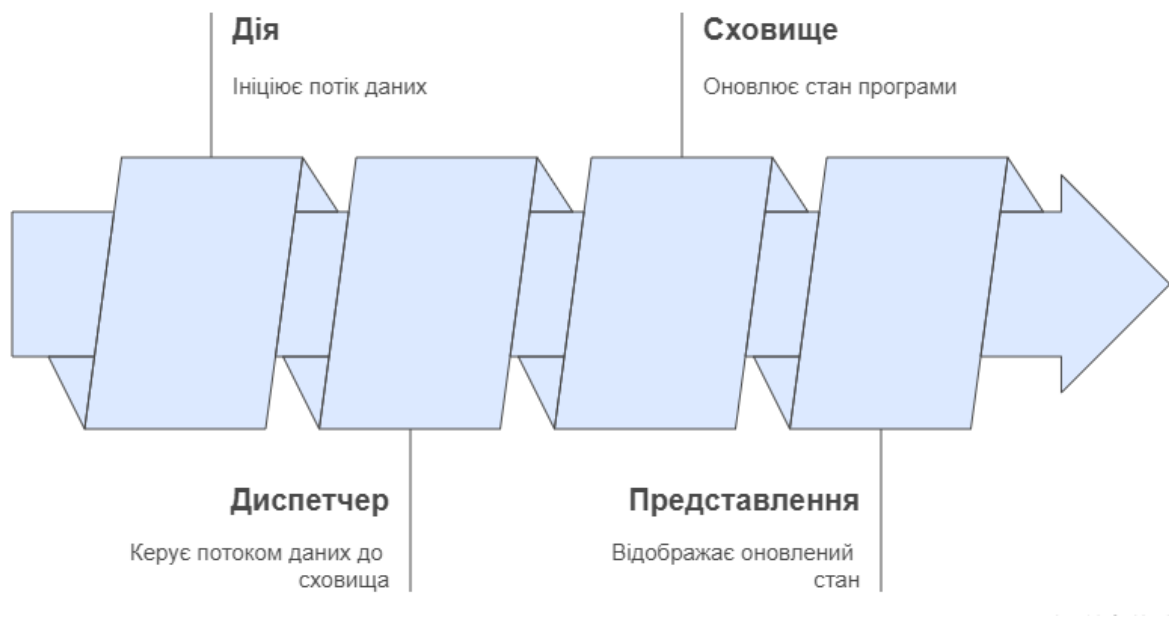


Рисунок 1.6 - Архітектурний патерн Flux/Redux

Для системи прокату автомобілів з мікролендінгом та адміністративною панеллю рекомендується використання комбінованого підходу: Flux/Redux для управління глобальним станом додатку (автентифікація, кошик бронювання, список автомобілів) та компонентний

підхід з локальним станом для ізольованих компонентів (форми, модальні вікна, інтерактивні елементи).

Безпека є критичним аспектом розробки вебдодатків, особливо у сфері прокату автомобілів, де обробляються персональні дані клієнтів та фінансова інформація. Основними викликами безпеки для SPA є:

- захист від XSS (Cross-Site Scripting) атак через динамічне генерування контенту;
- захист від CSRF (Cross-Site Request Forgery) атак при виконанні API-запитів;
- безпечне зберігання токенів автентифікації на стороні клієнта;
- захист від атак на механізм маршрутизації клієнтської сторони;
- обробка чутливих даних, таких як платіжна інформація та персональні дані клієнтів.

Для забезпечення належного рівня безпеки рекомендується застосовувати наступні практики:

- використання HTTPS для шифрування трафіку між клієнтом та сервером;
- впровадження механізмів автентифікації на основі JWT (JSON Web Tokens) з коротким терміном дії та механізмом оновлення;
- санітизація користувацького введення на стороні клієнта та сервера;
- застосування політик безпеки контенту (Content Security Policy) для запобігання виконанню шкідливих скриптів;
- використання HttpOnly та Secure прапорців для cookies;
- реалізація механізму захисту від CSRF через використання CSRF-токенів або SameSite прапорців для cookies;
- регулярний аудит залежностей на наявність вразливостей [10].

Продуктивність також є ключовим фактором успіху SPA, особливо для мікролендінгу, де швидкість завантаження та відгуку інтерфейсу

безпосередньо впливає на конверсію. Оптимізація продуктивності односторінкового додатку включає наступні аспекти:

- оптимізація розміру пакету JavaScript через розділення коду (code splitting) та ледаче завантаження компонентів;
- мінімізація кількості та розміру HTTP-запитів через об'єднання статичних ресурсів та використання HTTP/2;
- оптимізація рендерингу через віртуалізацію списків, мемоізацію компонентів та ефективне управління життєвим циклом;
- застосування кешування на стороні клієнта для зменшення кількості запитів до сервера;
- оптимізація зображень через використання сучасних форматів та технік ледачого завантаження;
- моніторинг та аналіз продуктивності за допомогою інструментів, таких як Lighthouse, WebPageTest та Chrome DevTools.

Метрики продуктивності, такі як First Contentful Paint (FCP), Largest Contentful Paint (LCP), Time to Interactive (TTI) та Cumulative Layout Shift (CLS), є важливими індикаторами якості користувацького досвіду та мають відповідати рекомендованим значенням відповідно до Core Web Vitals від Google.

1.3 Формування вимог до програмного забезпечення

На основі проведеного аналізу предметної області та технологічних аспектів розробки односторінкових вебдодатків, формування чітких вимог до програмного забезпечення прокату автомобілів є критичним етапом, що визначає успіх проєкту. Вимоги до програмного забезпечення можна розділити на три основні категорії: функціональні вимоги, нефункціональні вимоги та вимоги до інтерфейсу користувача. Кожна з цих категорій відіграє важливу роль у визначенні характеристик та обмежень майбутньої системи.

1.3.1 Функціональні вимоги до системи

Функціональні вимоги описують конкретні можливості та функції, які має надавати програмне забезпечення прокату автомобілів. На основі аналізу потреб користувачів та бізнес-процесів у сфері прокату автомобілів в Україні, визначено наступні ключові функціональні вимоги.

1.3.1.1 Мікролендінг (клієнтська частина)

Для користувацької частини системи визначено наступні функціональні вимоги:

- реєстрація та авторизація користувачів з можливістю входу через соціальні мережі та електронну пошту; підтримка різних ролей користувачів (гість, зареєстрований користувач) з відповідними рівнями доступу до функціоналу;
- пошук та фільтрація автомобілів за різноманітними параметрами: марка, модель, тип кузова, рік випуску, коробка передач, тип палива, ціновий діапазон, наявність додаткових опцій;
- перегляд детальної інформації про автомобіль: технічні характеристики, фотогалерея, опис стану, умови прокату, наявність на конкретні дати;
- калькуляція вартості прокату з урахуванням тривалості, додаткових послуг та знижок; підтримка різних цінових політик залежно від тривалості оренди (година, доба, тиждень, місяць);
- онлайн-бронювання автомобіля з вибором дати та часу початку і завершення оренди; можливість резервування конкретного автомобіля або класу автомобіля;
- оформлення замовлення з вибором додаткових послуг: страхування, дитячі крісла, GPS-навігатор, додатковий водій, доставка автомобіля;

- онлайн-оплата з підтримкою різних платіжних методів: банківські картки, електронні гаманці, онлайн-банкінг; можливість часткової передоплати з доплатою при отриманні автомобіля;
- перегляд історії замовлень та статусу поточних бронювань; можливість скасування або модифікації бронювань відповідно до політики компанії;
- система відгуків та оцінок для автомобілів та сервісу в цілому; можливість для користувачів залишати коментарі та рейтинги після завершення оренди;
- інтеграція з картографічними сервісами для відображення місць видачі та повернення автомобілів; розрахунок маршрутів та відстаней.

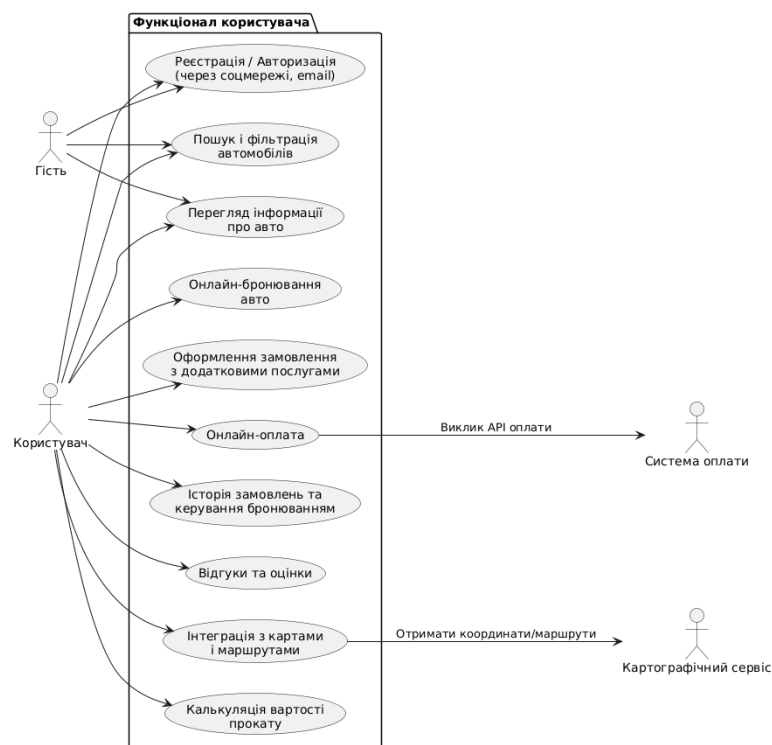


Рисунок 1.7 – Діаграма взаємодії

1.3.1.2 Адміністративна панель

Для адміністративної частини системи визначено наступні функціональні вимоги:

- управління автопарком: додавання, редагування та видалення інформації про автомобілі; управління наявністю та статусом автомобілів (доступний, на обслуговуванні, орендований);
- управління цінами та акціями: встановлення базових та спеціальних тарифів; створення промо-кодів та знижок; налаштування сезонних пропозицій;
- управління замовленнями: перегляд всіх бронювань з можливістю фільтрації та сортування; підтвердження, редагування або скасування бронювань; генерація документів для оформлення прокату;
- управління клієнтською базою: перегляд та редагування інформації про клієнтів; аналіз історії замовлень; налаштування системи знижок для постійних клієнтів;
- управління контентом мікролендінгу: редагування текстів, зображень та мультимедійних матеріалів; налаштування відображення акційних пропозицій;
- система сповіщень: налаштування автоматичних повідомлень для клієнтів та співробітників; сповіщення про нові бронювання, наближення термінів завершення оренди, необхідність технічного обслуговування;
- управління правами доступу: створення та редагування облікових записів співробітників; налаштування рівнів доступу до різних функцій системи;
- журналювання дій користувачів системи для аудиту та безпеки; відстеження змін та операцій в системі.

1.3.2 Нефункціональні вимоги

Нефункціональні вимоги визначають якісні характеристики та обмеження системи, що впливають на її ефективність, надійність та зручність використання. Для програмного забезпечення прокату автомобілів визначено наступні нефункціональні вимоги:

1.3.2.1 Продуктивність

- час відгуку системи: для основних операцій користувацького інтерфейсу (завантаження сторінки, перехід між розділами, пошук автомобілів) не повинен перевищувати 1 секунду при нормальному навантаженні;
- швидкість завантаження сторінки: показник First Contentful Paint (FCP) не повинен перевищувати 1.8 секунди, Largest Contentful Paint (LCP) - не більше 2.5 секунд;
- пропускна здатність: система повинна забезпечувати одночасну роботу не менше 500 користувачів без деградації продуктивності;
- масштабованість: архітектура системи повинна передбачати можливість горизонтального масштабування для збільшення пропускної здатності при зростанні навантаження;
- оптимізація бази даних: час виконання складних запитів не повинен перевищувати 3 секунди; індексація ключових полів для прискорення пошуку.

1.3.2.2 Безпека

- аутентифікація та авторизація: багатофакторна аутентифікація для адміністративного доступу; ролева модель авторизації з принципом найменших привілеїв;
- захист даних: шифрування чутливих даних (персональні дані клієнтів, платіжна інформація) як при зберіганні, так і при передачі; відповідність вимогам GDPR та Закону України "Про захист персональних даних";
- захист від вразливостей: імплементація захисту від OWASP Top 10 вразливостей (XSS, CSRF, SQL-ін'єкції, тощо);

- аудит безпеки: регулярне проведення тестів на проникнення та сканування вразливостей; журналювання всіх дій, пов'язаних з безпекою; • відновлення після збоїв: регулярне резервне копіювання даних з можливістю швидкого відновлення; механізми автоматичного відновлення після збоїв.

1.3.2.3 Надійність та доступність

- доступність системи: показник uptime не менше 99.9% (допустимий час простою не більше 8.76 годин на рік);
- відмовостійкість: система повинна продовжувати функціонувати при відмові окремих компонентів; резервування критичних елементів інфраструктури;
- обробка помилок: система повинна коректно обробляти всі можливі помилки та виняткові ситуації, надаючи користувачу зрозумілі повідомлення;
- резервне копіювання: автоматичне створення резервних копій даних не рідше ніж раз на добу; зберігання копій протягом не менше 30 днів.

1.3.2.4 Масштабованість та розширюваність

- горизонтальна масштабованість: можливість додавання нових серверів для збільшення пропускної здатності;
- вертикальна масштабованість: можливість збільшення ресурсів окремих компонентів системи;
- модульність архітектури: система повинна складатися з незалежних модулів, що можуть розроблятися, тестуватися та розгортатися окремо;
- API-first підхід: всі функції системи повинні бути доступні через API, що спрощує інтеграцію з іншими системами та розробку нових інтерфейсів.

1.3.2.5 Зручність використання та доступність

- адаптивний дизайн: інтерфейс повинен коректно відображатися на пристроях з різними розмірами екрану (desktop, tablet, mobile);
- відповідність стандартам доступності WCAG 2.1 рівня AA для забезпечення доступу користувачів з обмеженими можливостями;
- багатомовність: підтримка української, англійської та російської мов з можливістю легкого додавання нових мов;
- зрозумілість інтерфейсу: інтуїтивно зрозумілий інтерфейс, що не вимагає спеціального навчання для виконання основних операцій;
- підтримка різних браузерів: система повинна коректно працювати в останніх версіях популярних браузерів (Chrome, Firefox, Safari, Edge).

1.3.2.6 Підтримка та супровід

- документація: детальна технічна документація для розробників та посібники користувача;
- моніторинг: система повинна надавати інструменти для моніторингу продуктивності та виявлення проблем;
- логування: детальне журналювання операцій для аналізу проблем та статистики використання;
- сумісність: чітко визначені вимоги до сумісності з операційними системами, браузерами та зовнішніми сервісами.

Таблиця 1.2 представляє метрики для оцінки відповідності нефункціональним вимогам.

Таблиця 1.2 - Метрики оцінки нефункціональних вимог

Категорія	Метрика	Цільове значення
Продуктивність	Час відгуку системи	< 1 секунди
Продуктивність	First Contentful Paint (FCP)	< 1.8 секунди
Продуктивність	Largest Contentful Paint (LCP)	< 2.5 секунди
Продуктивність	Time to Interactive (TTI)	< 3.8 секунди
Продуктивність	Cumulative Layout Shift (CLS)	< 0.1
Масштабованість	Одночасні користувачі	> 500
Безпека	Покриття OWASP Top 10	100%
Надійність	Uptime	> 99.9%
Надійність	Час відновлення (RTO)	< 4 години
Надійність	Максимально допустимі втрати даних (RPO)	< 24 години
Доступність	Відповідність WCAG 2.1	Рівень AA
Локалізація	Підтримувані мови	UK

1.3.3.2 Специфічні вимоги до інтерфейсу мікролендінгу

- головна сторінка з яскравим візуальним представленням пропозицій та зручною формою пошуку автомобілів; чіткий заклик до дії (СТА) та виділення ключових переваг сервісу;
- зручна система фільтрації та сортування автомобілів з візуалізацією доступних опцій; можливість багатокритеріального пошуку з збереженням параметрів фільтрації;
- детальна картка автомобіля з галереєю високоякісних зображень, детальним описом характеристик та умов прокату; інтерактивні елементи для вибору додаткових послуг та розрахунку вартості;
- інтерактивний календар для вибору дат прокату з візуалізацією доступності автомобілів та різних тарифів; можливість порівняння вартості для різних періодів;

- спрощена форма оформлення замовлення з мінімальною кількістю полів та прогресивним розкриттям інформації; збереження даних користувача для повторних замовлень;
- інтеграція з картографічними сервісами для візуалізації місць отримання та повернення автомобіля; розрахунок маршрутів та відстаней;
- мобільна версія з оптимізацією для тач-інтерфейсу: збільшені елементи керування, спрощена навігація, адаптовані форми введення даних.

1.3.4 Технологічні вимоги та обмеження

Для забезпечення ефективної розробки, розгортання та підтримки системи прокату автомобілів визначено наступні технологічні вимоги та обмеження.

1.3.4.1 Вимоги до технологічного стеку

- frontend: використання React.js як основної бібліотеки для розробки користувацького інтерфейсу; застосування TypeScript для статичної типізації; використання Redux або Context API для управління станом додатку;
- backend: розробка RESTful API з використанням Python та FastAPI; використання асинхронної обробки запитів для забезпечення високої пропускної здатності;
- база даних: використання PostgreSQL як основної реляційної бази даних; застосування Redis для кешування та швидкого доступу до часто запитуваних даних;
- інфраструктура: розгортання системи в Docker-контейнерах для забезпечення ізоляції та спрощення масштабування; використання Kubernetes для оркестрації контейнерів;

- CI/CD: впровадження автоматизованого процесу збірки, тестування та розгортання з використанням GitLab CI/CD або GitHub Actions;
- моніторинг та логування: використання ELK Stack (Elasticsearch, Logstash, Kibana) для централізованого збору та аналізу логів; застосування Prometheus та Grafana для моніторингу продуктивності.

1.3.4.2 Вимоги до розгортання та хостингу

- мультизональне розгортання для забезпечення високої доступності та відмовостійкості; використання хмарних провайдерів з наявністю дата-центрів в Україні або Європі;
- можливість швидкого масштабування інфраструктури при зростанні навантаження; автоматичне масштабування на основі метрик використання ресурсів;
- забезпечення захищеного з'єднання через HTTPS з використанням сертифікатів SSL/TLS; автоматичне оновлення сертифікатів;
- регулярне резервне копіювання даних з можливістю швидкого відновлення; зберігання резервних копій в географічно розподілених локаціях;
- відповідність вимогам законодавства України щодо зберігання та обробки персональних даних; можливість розгортання в мультирегіональній конфігурації для відповідності вимогам різних юрисдикцій.

1.3.4.3 Обмеження та припущення

- система розробляється з орієнтацією на український ринок прокату автомобілів з урахуванням локальних особливостей та законодавства;
- початкова версія системи буде підтримувати до 50 автомобілів в автопарку з можливістю масштабування до 500+ автомобілів в майбутньому;

- система орієнтована на роботу з кінцевими користувачами (B2C) та не включає функціонал для корпоративних клієнтів (B2B) на початковому етапі;
- інтеграція з платіжними системами обмежується найбільш популярними в Україні провайдерами (LiqPay, WayForPay, Fondy);
- система не передбачає інтеграцію з системами GPS-моніторингу автомобілів на початковому етапі, але архітектура повинна дозволяти додати цю функціональність в майбутньому.

1.4 Висновки за розділом 1

У першому розділі було проведено комплексний аналіз предметної області прокату автомобілів та розглянуто технологічні аспекти розробки односторінкових вебдодатків. Виконано детальне дослідження ринку програмного забезпечення для прокату автомобілів, який демонструє стабільне зростання і, за прогнозами, досягне 45 мільярдів доларів до 2027 року з річним темпом зростання 10,8%.

Проаналізовано специфіку українського ринку прокату автомобілів, який характеризується щорічним зростанням на 15-20%, обсягом близько 2000 автомобілів та співвідношенням міжнародних і локальних компаній. Виявлено, що близько 70% клієнтів автопрокату - це громадяни України, а 50% клієнтів замовляють автомобілі для бізнес-заходів, у той час як інші 50% - для особистих поїздок.

На основі проведеного аналізу сформовано функціональні вимоги до системи, які охоплюють широкий спектр можливостей: від реєстрації та авторизації користувачів до онлайн-бронювання, управління автопарком та адміністративних функцій. Визначено нефункціональні вимоги, що стосуються продуктивності, безпеки, надійності, масштабованості та зручності використання системи.

2 МОДЕЛЬ ТА МЕТОДИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Загальна архітектура системи

Архітектура програмного забезпечення прокату автомобілів є ключовим фактором, що визначає ефективність, масштабованість та надійність системи. На основі визначених вимог обрано клієнт-серверну архітектуру з мікросервісними елементами, що забезпечує оптимальний баланс між продуктивністю, гнучкістю розробки та зручністю масштабування.

2.1.1 Обґрунтування вибору архітектурного підходу

При виборі архітектурного підходу було проаналізовано декілька альтернативних варіантів: монолітна архітектура, сервіс-орієнтована архітектура (SOA) та мікросервісна архітектура. Порівняльний аналіз цих підходів представлено в табл. 2.1.

Враховуючи специфіку проєкту прокату автомобілів, який на початковому етапі має обмежений функціонал та кількість користувачів, але потребує можливості гнучкого масштабування в майбутньому, було обрано гібридний підхід, що поєднує елементи монолітної та мікросервісної архітектури. Основна логіка додатку реалізується як моноліт з чітким розмежуванням на модулі, які в майбутньому можуть бути виділені в окремі мікросервіси при зростанні навантаження та функціоналу.

Таке рішення дозволяє досягти наступних переваг:

- швидка розробка та розгортання початкової версії системи за рахунок спрощеної інфраструктури;
- чітке розмежування на функціональні модулі, що спрощує підтримку та розширення системи;

- можливість поступового переходу до повноцінної мікросервісної архітектури при зростанні проєкту без необхідності повного перепроєктування;
- оптимальне використання ресурсів для проєкту середнього масштабу.

Таблиця 2.1 - Порівняння архітектурних підходів для системи прокату автомобілів

Критерій	Монолітна архітектура	Сервіс-орієнтована архітектура	Мікросервісна архітектура
Швидкість розробки початкової версії	Висока	Середня	Низька
Складність розгортання	Низька	Середня	Висока
Ізоляція компонентів	Низька	Середня	Висока
Масштабованість	Обмежена	Гарна	Відмінна
Стійкість до відмов	Низька	Середня	Висока
Можливість незалежного оновлення компонентів	Обмежена	Середня	Висока
Ефективність використання ресурсів	Висока	Середня	Низька
Складність моніторингу та діагностики	Низька	Середня	Висока
Гнучкість вибору технологій	Обмежена	Середня	Висока

2.1.2 Компонентна структура вебдодатку

Програмне забезпечення прокату автомобілів складається з наступних основних компонентів (рис. 2.1).



Рисунок 2.1 - Компонентна структура програмного забезпечення прокату автомобілів

Кожен компонент системи має чітко визначену відповідальність та інтерфейси взаємодії з іншими компонентами, що забезпечує модульність та гнучкість архітектури.

2.1.3 Схема взаємодії між компонентами

Взаємодія між компонентами системи будується на основі REST API для синхронних операцій та системи подій для асинхронних процесів. Схема взаємодії компонентів представлена на рис. 2.2.

Основні принципи взаємодії компонентів:

- клієнтські додатки (мікролендінг та адміністративна панель) взаємодіють із серверною частиною через API-шлюз, який забезпечує єдину точку входу та відповідає за маршрутизацію запитів до відповідних модулів;
- API-шлюз реалізує механізми автентифікації, авторизації та контролю доступу, що забезпечує безпеку системи;
- взаємодія між внутрішніми модулями серверної частини відбувається через прямі виклики у випадку монолітної реалізації або через REST API у випадку мікросервісної реалізації;
- асинхронна взаємодія між компонентами реалізується через шину повідомлень (Apache Kafka або RabbitMQ), що дозволяє розділити

відправника та отримувача повідомлень, забезпечуючи слабке зв'язування компонентів;

- доступ до бази даних здійснюється через рівень абстракції даних (Repository Pattern), що ізолює бізнес-логіку від деталей реалізації сховища даних;

- зовнішні інтеграції реалізуються через адаптери, що забезпечують єдиний інтерфейс для взаємодії з різними зовнішніми системами.

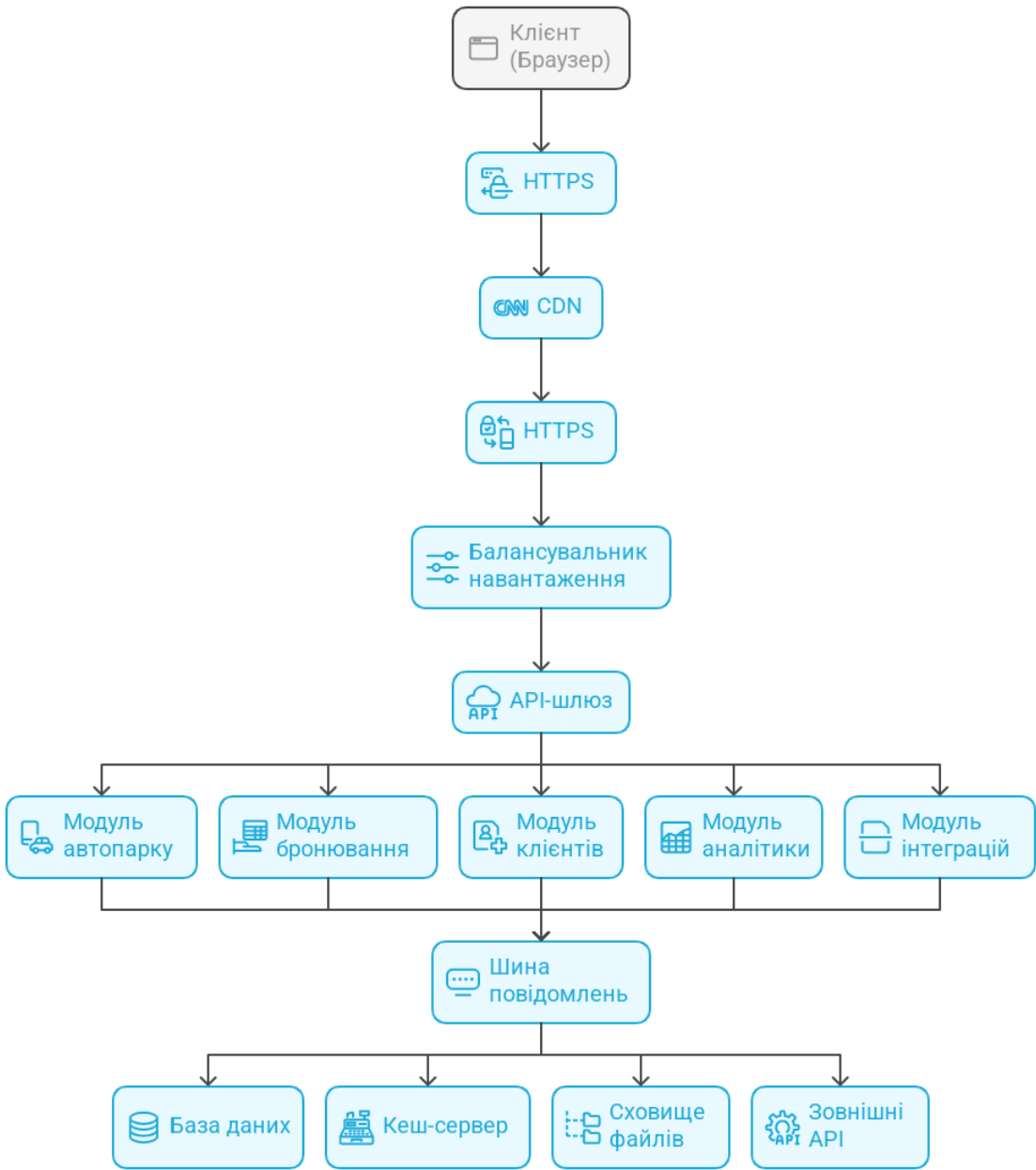


Рисунок 2.2 - Схема взаємодії компонентів системи

2.1.4 Реалізація паттернів проєктування

У процесі розробки програмного забезпечення прокату автомобілів застосовуються наступні патерни проєктування:

- repository pattern: забезпечує абстракцію доступу до даних, що дозволяє змінювати реалізацію сховища даних без зміни бізнес-логіки; реалізується як сервісний шар між бізнес-логікою та джерелом даних;
- dependency injection: забезпечує інверсію контролю та слабке зв'язування компонентів; реалізується через FastAPI dependency injection систему на бекенді та DI-контейнери на фронтенді;
- factory method: використовується для створення об'єктів різних типів з спільним інтерфейсом; застосовується для створення різних типів повідомлень, сповіщень та адаптерів зовнішніх інтеграцій;
- builder pattern: застосовується для покрокового створення складних об'єктів, таких як замовлення з багатьма параметрами та додатковими послугами;
- strategy pattern: дозволяє визначати різні алгоритми та інкапсулювати їх у окремі класи; використовується для реалізації різних стратегій ціноутворення, розрахунку знижок та вибору методу оплати;
- observer pattern: забезпечує механізм підписки та сповіщення об'єктів про зміни стану; реалізується через систему подій для асинхронного оповіщення про важливі події в системі (нове замовлення, зміна статусу, тощо);
- facade pattern: надає спрощений інтерфейс до складної підсистеми; використовується для абстрагування складних інтеграцій з зовнішніми системами та API.

Використання паттернів проєктування забезпечує гнучкість, розширюваність та підтримуваність архітектури системи, що є особливо важливим для довгострокових проєктів з перспективою масштабування функціоналу.

2.2 Вибір мов програмування

При розробці програмного забезпечення для прокату автомобілів вибір відповідних мов програмування та інструментів реалізації є критичним фактором, що визначає ефективність розробки, продуктивність системи та можливості подальшого масштабування. Для прийняття обґрунтованого рішення було проведено порівняльний аналіз найбільш релевантних технологій для кожного шару архітектури: клієнтська частина (frontend), серверна частина (backend) та система управління базами даних.

2.2.1 Вибір технологій для клієнтської частини

Для розробки клієнтської частини односторінкового додатку було проаналізовано три найпопулярніших фреймворки: React, Angular та Vue.js. Кожен з них має свої переваги та особливості, що впливають на швидкість розробки, продуктивність та зручність підтримки коду.

React, розроблений Facebook, представляє собою бібліотеку для створення користувацьких інтерфейсів з компонентним підходом. Ключовою особливістю React є віртуальний DOM, що забезпечує високу продуктивність через оптимізацію процесу оновлення інтерфейсу. React використовує JSX-синтаксис, що дозволяє інтегрувати HTML-подібну структуру безпосередньо в JavaScript код, полегшуючи розробку компонентів. Екосистема React включає багато інструментів, таких як Redux для управління станом, React Router для маршрутизації та численні бібліотеки компонентів, що прискорюють розробку.

Angular, розроблений Google, є повноцінним фреймворком з комплексною архітектурою на основі компонентів, сервісів та модулів. Angular використовує TypeScript, що забезпечує статичну типізацію та підвищує надійність коду. Фреймворк має вбудовані рішення для більшості завдань, включаючи маршрутизацію (Angular Router), управління формами

(Angular Forms) та HTTP-комунікацію (HttpClient). Angular використовує двосторонню прив'язку даних та систему залежностей (dependency injection), що спрощує розробку складних додатків.

Vue.js позиціонується як прогресивний фреймворк, що може бути інтегрований поступово в існуючі проекти. Фреймворк поєднує переваги React та Angular, пропонуючи реактивну систему зв'язування даних та компонентний підхід. Vue.js відрізняється низьким порогом входження для розробників завдяки простому API та зрозумілій документації. Екосистема Vue.js включає офіційні бібліотеки для маршрутизації (Vue Router) та управління станом (Vuex).

Результати порівняльного аналізу фреймворків для клієнтської частини представлені в табл. 2.2.

Таблиця 2.2 - Порівняння фреймворків для клієнтської частини

Критерій	React	Angular	Vue.js
Розмір бібліотеки (gzipped)	~40KB	~130KB	~30KB
Продуктивність	Висока	Середня	Висока
Крива навчання	Помірна	Висока	Низька
Статична типізація	Опціонально (з TypeScript)	Вбудована (TypeScript)	Опціонально (з TypeScript)
Екосистема	Дуже розвинена	Розвинена	Зростаюча
Підтримка компаній	Facebook	Google	Спільнота
Популярність (GitHub stars)	201K	87K	205K
Швидкість розробки	Висока	Середня	Висока

На основі проведеного аналізу для розробки клієнтської частини системи прокату автомобілів було обрано React через його високу продуктивність, гнучкість, велику екосистему та значну підтримку спільноти. Додатково, для забезпечення типової безпеки та підвищення якості коду, було вирішено використовувати TypeScript як основну мову програмування для frontend-розробки. Така комбінація дозволяє отримати переваги статичної типізації разом з гнучкістю та продуктивністю React.

В рамках обраного стеку технологій для frontend-розробки також було вирішено використовувати:

Redux для управління глобальним станом додатку, що забезпечує передбачуваний потік даних та спрощує відлагодження; React Router для реалізації клієнтської маршрутизації, що дозволяє створювати односторінковий додаток з навігацією без перезавантаження сторінки; Styled Components для створення стилів з використанням CSS-in-JS підходу, що забезпечує інкапсуляцію стилів на рівні компонентів; React Hook Form для управління формами з мінімальними перерендерами та вбудованою валідацією; Axios для здійснення HTTP-запитів до серверної частини додатку з підтримкою інтерцепторів та автоматичною трансформацією JSON.

2.2.2 Вибір технологій для серверної частини

Для розробки серверної частини системи прокату автомобілів було проаналізовано три популярні технології: Node.js з Express.js, Python з FastAPI та Java з Spring Boot. Кожна з цих технологій має свої особливості, що впливають на ефективність розробки, продуктивність та масштабованість системи.

Node.js з Express.js представляє собою JavaScript-платформу для серверної розробки з асинхронною, подієво-орієнтованою архітектурою. Express.js є мінімалістичним та гнучким фреймворком для Node.js, що надає базові функціональні можливості для побудови вебдодатків та API.

Перевагами цього стеку є використання однієї мови програмування (JavaScript/TypeScript) як на клієнтській, так і на серверній частині, що спрощує розробку та підтримку; значна екосистема пакетів через npm; висока продуктивність при обробці асинхронних операцій; низький поріг входження для frontend-розробників.

Python з FastAPI є сучасним, високопродуктивним фреймворком для розробки API на основі стандарту ASGI. FastAPI базується на Pydantic для валідації даних та OpenAPI для автоматичної генерації документації. Перевагами цього стеку є висока продуктивність завдяки асинхронній обробці запитів; автоматична валідація даних та генерація документації API; статична типізація з використанням анотацій типів Python; простота та зрозумілість синтаксису Python; велика кількість бібліотек для обробки даних та машинного навчання.

Java з Spring Boot є потужним фреймворком для розробки корпоративних додатків на основі Java. Spring Boot надає готові набори налаштувань для швидкого старту проєктів та включає велику кількість модулів для різних завдань. Перевагами цього стеку є висока стабільність та надійність; потужні можливості для розробки корпоративних додатків; вбудована підтримка транзакцій та інтеграція з різними базами даних; розвинена екосистема інструментів та бібліотек; масштабованість та продуктивність при високих навантаженнях.

Результати порівняльного аналізу технологій для серверної частини представлені в табл. 2.3.

На основі проведеного аналізу та специфіки проєкту прокату автомобілів було обрано Python з FastAPI для розробки серверної частини системи. Ключовими факторами цього вибору стали:

Висока продуктивність FastAPI завдяки використанню асинхронних операцій, що важливо для обробки великої кількості запитів у пікові години; автоматична валідація даних та генерація документації API через інтеграцію з OpenAPI та Swagger UI, що спрощує розробку та тестування; простота та

зрозумілість синтаксису Python, що прискорює розробку та знижує ймовірність помилок; багата екосистема бібліотек Python для обробки даних, інтеграції з зовнішніми сервісами та аналітики, що важливо для системи прокату автомобілів.

Таблиця 2.3 - Порівняння технологій для серверної частини

Критерій	Node.js + Express.js	Python + FastAPI	Java + Spring Boot
Продуктивність	Висока для ІО-операцій	Висока з асинхронністю	Висока для обчислень
Масштабованість	Добра	Добра	Відмінна
Статична типізація	Опціонально (з TypeScript)	Вбудована (з анотаціями)	Вбудована
Екосистема	Дуже розвинена	Розвинена	Дуже розвинена
Крива навчання	Низька	Низька	Висока
Автоматична документація	Потребує додаткових рішень	Вбудована (Swagger/OpenAPI)	Потребує додаткових рішень
Асинхронність	Вбудована	Вбудована	Потребує додаткових рішень
Зрілість технології	Зріла	Нова, але зростаюча	Дуже зріла

В рамках обраного стеку технологій для backend-розробки також було вирішено використовувати:

SQLAlchemy як ORM для взаємодії з базою даних, що забезпечує гнучкий та потужний інструментарій для роботи з реляційними даними; Pydantic для валідації даних та серіалізації/десеріалізації JSON; Alembic для управління міграціями бази даних; JWT для реалізації автентифікації та

авторизації; Celery для обробки асинхронних завдань, таких як розсилка повідомлень та генерація звітів.

2.2.3 Вибір системи управління базами даних

Для системи прокату автомобілів важливо вибрати надійну та продуктивну систему управління базами даних, що забезпечить ефективне зберігання та обробку даних. Було проведено порівняльний аналіз трьох популярних СУБД: PostgreSQL, MySQL та MongoDB, щоб визначити найбільш відповідну для проекту.

PostgreSQL є потужною об'єктно-реляційною системою управління базами даних з відкритим кодом. Вона відома своєю надійністю, розширюваністю та відповідністю стандартам SQL. PostgreSQL підтримує складні запити, індекси, тригери, збережені процедури та має потужні можливості для забезпечення цілісності даних. Перевагами PostgreSQL є відмінна підтримка транзакцій та ACID-властивостей; розширена підтримка JSON та інших нестандартних типів даних; можливість створення користувацьких типів даних та функцій; висока масштабованість та продуктивність; активний розвиток та підтримка спільнотою.

MySQL є популярною реляційною системою управління базами даних, що відома своєю швидкістю та простотою використання. MySQL широко застосовується для вебдодатків різного масштабу та складності. Перевагами MySQL є висока швидкість роботи для операцій читання; простота налаштування та використання; широка підтримка хостинг-провайдерів; хороша інтеграція з іншими вебтехнологіями; нижчі вимоги до ресурсів порівняно з PostgreSQL.

MongoDB є документо-орієнтованою NoSQL базою даних, що зберігає дані у форматі BSON (бінарний JSON). MongoDB надає гнучку схему даних, що дозволяє змінювати структуру документів без необхідності модифікації всієї бази даних. Перевагами MongoDB є гнучкість схеми даних,

що спрощує еволюцію структури даних; зберігання даних у форматі, близькому до JSON, що спрощує інтеграцію з вебдодатками; висока масштабованість через вбудовану підтримку шардингу; хороша продуктивність для операцій запису та складних запитів до документів.

Результати порівняльного аналізу систем управління базами даних представлені в табл. 2.4.

На основі проведеного аналізу та враховуючи специфіку системи прокату автомобілів, було обрано PostgreSQL як основну систему управління базами даних. Ключовими факторами цього вибору стали:

Надійність та стабільність PostgreSQL, що критично для системи управління бізнес-процесами; сильна підтримка транзакцій та ACID-властивостей, що забезпечує цілісність даних при одночасних бронюваннях та оплатах; розширена підтримка JSON, що дозволяє гнучко зберігати додаткові характеристики автомобілів та налаштування користувачів; можливість встановлення складних обмежень та тригерів для забезпечення бізнес-правил системи прокату; хороша інтеграція з Python та FastAPI через ORM SQLAlchemy.

Таблиця 2.4 - Порівняння систем управління базами даних

Критерій	PostgreSQL	MySQL	MongoDB
Тип СУБД	Об'єктно-реляційна	Реляційна	Документо-орієнтована (NoSQL)
Підтримка транзакцій	Повна (ACID)	Залежить від движка	Обмежена
Масштабованість	Висока	Середня	Дуже висока
Підтримка JSON	Розширена	Базова	Вбудована
Цілісність даних	Сильна	Сильна	Обмежена
Гнучкість схеми	Обмежена	Обмежена	Висока
Вимоги до ресурсів	Високі	Середні	Середні

Швидкість читання	Висока	Дуже висока	Висока
Швидкість запису	Висока	Середня	Дуже висока
Підтримка спільнотою	Активна	Активна	Активна

Для кешування даних та оптимізації часто запитуваної інформації додатково було вирішено використовувати Redis як in-memory сховище даних. Redis забезпечує швидкий доступ до кешованих даних, таких як доступність автомобілів на певні дати, популярні пошукові запити та сесії користувачів. Використання Redis дозволяє значно підвищити продуктивність системи та зменшити навантаження на основну базу даних.

2.3 Проєктування інтерфейсу користувача

Інтерфейс користувача є ключовим аспектом системи прокату автомобілів, оскільки безпосередньо впливає на користувацький досвід та конверсію відвідувачів у клієнтів. Проєктування інтерфейсу базується на принципах UX/UI дизайну, що забезпечують інтуїтивно зрозумілу та ефективну взаємодію користувача з системою.

2.3.1 Розробка UX/UI концепції мікролендінгу

Концепція UX/UI дизайну мікролендінгу базується на наступних принципах:

- мінімалістичний дизайн з фокусом на ключовому контенті та функціоналі;
- візуальна ієрархія, що підкреслює найважливіші елементи та спрямовує увагу користувача;
- інтуїтивно зрозуміла навігація з чіткою структурою та логічним розташуванням елементів;

- відповідність бренду компанії через використання фірмових кольорів, шрифтів та елементів дизайну;

- оптимізація для конверсії з стратегічно розташованими закликами до дії (СТА).

Кольорова схема мікролендінгу включає:

- основний колір: #2C3E50 (темно-синій) - символізує надійність та професіоналізм;
- акцентний колір: #E67E22 (помаранчевий) - для виділення ключових елементів та кнопок СТА;
- нейтральні кольори: #ECF0F1 (світло-сірий) та #FFFFFF (білий) - для фону та створення чистого, сучасного вигляду.

Типографіка базується на використанні шрифтів без засічок (Roboto або Open Sans) для забезпечення чіткості та читабельності на різних пристроях. Ієрархія шрифтів включає:

- H1: 32px, жирний (для головних заголовків);
- H2: 24px, жирний (для підзаголовків секцій);
- H3: 18px, жирний (для заголовків компонентів);
- основний текст: 16px, звичайний (для основного контенту);
- другорядний текст: 14px, легкий (для додаткової інформації).

2.3.2 Проектування форм вибору та прокату автомобілів

Форми є критичним елементом інтерфейсу, оскільки безпосередньо впливають на процес конверсії. При проектуванні форм вибору та прокату автомобілів застосовуються наступні підходи:

- мінімізація кількості полів для введення даних, що знижує когнітивне навантаження на користувача;
- логічне групування полів за функціональним призначенням;
- прогресивне розкриття інформації, коли додаткові поля з'являються лише при необхідності;

- валідація форм в реальному часі з інформативними повідомленнями про помилки;
- автозаповнення полів на основі раніше введених даних для зареєстрованих користувачів;
- можливість збереження проміжних результатів заповнення форми.

Форма пошуку автомобілів розміщується у верхній частині головної сторінки та включає такі поля:

- дата початку оренди (календар з обмеженням на минулі дати);
- дата завершення оренди (календар з обмеженням на дати раніше початку оренди);
- місце отримання автомобіля (випадаючий список з можливістю пошуку);
- місце повернення автомобіля (випадаючий список з можливістю пошуку);
- категорія автомобіля (опціонально, випадаючий список).

2.3.3 Адаптивний дизайн для різних пристроїв

Адаптивний дизайн забезпечує оптимальне відображення інтерфейсу на різних пристроях та розмірах екранів. Для реалізації адаптивного дизайну застосовуються наступні підходи:

- використання системи сітки (grid system), що динамічно адаптується до розміру екрану;
- медіа-запити (media queries) для налаштування стилів відповідно до розміру екрану;
- флюїдні зображення, що автоматично масштабуються в залежності від контейнера;
- адаптивна типографіка з використанням відносних одиниць вимірювання (rem, em, vw);

- оптимізація навігації для мобільних пристроїв (використання гамбургер-меню, спрощення структури).

Адаптивний дизайн реалізується для трьох основних типів пристроїв:

- десктопні комп'ютери та ноутбуки (ширина екрану від 1200px);
- планшети (ширина екрану від 768px до 1199px);
- мобільні пристрої (ширина екрану від 320px до 767px).

2.4 Проєктування адміністративної панелі

Адміністративна панель є ключовим інструментом для управління системою прокату автомобілів. Вона надає співробітникам компанії можливість ефективно керувати автопарком, обробляти замовлення, взаємодіяти з клієнтами та аналізувати бізнес-показники.

2.4.1 Структура та функціональні модулі

Адміністративна панель складається з наступних функціональних модулів:

- дашборд: відображає загальну інформацію про стан системи, ключові показники ефективності (КПІ) та сповіщення, що вимагають уваги;
- модуль управління автопарком: забезпечує функціонал для додавання, редагування та видалення інформації про автомобілі, їх характеристики та статуси;
- модуль управління замовленнями: дозволяє переглядати, редагувати та обробляти бронювання, змінювати їх статуси та комунікувати з клієнтами;
- модуль управління клієнтами: надає інструменти для роботи з базою клієнтів, перегляду історії замовлень та налаштування програми лояльності;
- модуль управління тарифами та акціями: забезпечує функціонал для налаштування цін, створення акційних пропозицій та управління промо-кодами;

- модуль управління контентом: надає інструменти для редагування текстів, зображень та структури мікролендінгу;
- модуль налаштувань: забезпечує функціонал для конфігурації системи, управління користувачами та налаштування інтеграцій.

Структура адміністративної панелі представлена на рис. 2.3.

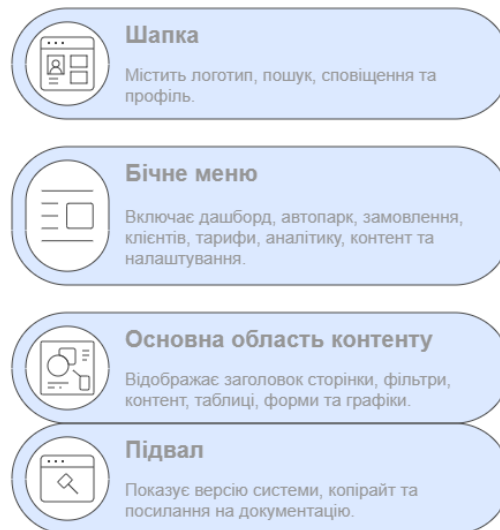


Рисунок 2.3 - Структура адміністративної панелі

2.4.2 Система авторизації та контролю доступу

Система авторизації та контролю доступу забезпечує безпеку адміністративної панелі та розмежування прав доступу для різних категорій співробітників. Система будується на основі ролевої моделі (Role-Based Access Control, RBAC), що дозволяє гнучко налаштовувати права доступу та ефективно управляти ними.

Основні ролі в системі:

- адміністратор: має повний доступ до всіх функцій системи, включаючи управління користувачами та налаштування системи;
- менеджер: має доступ до управління автопарком, замовленнями, клієнтами та тарифами, але не може змінювати налаштування системи;

- оператор: має обмежений доступ до управління замовленнями та комунікації з клієнтами, але не може змінювати автопарк або тарифи;

- аналітик: має доступ до аналітичних даних та звітів, але не може змінювати дані в системі;

- контент-менеджер: має доступ до управління контентом мікролендінгу, але не може втручатися в бізнес-операції.

Кожна роль має набір дозволів, що визначають можливості доступу до різних функцій системи. Дозволи групуються за функціональними модулями та типами операцій (перегляд, створення, редагування, видалення).

Процес авторизації включає наступні етапи:

- автентифікація користувача (перевірка логіну та паролю);
- отримання інформації про ролі та дозволи користувача;
- перевірка наявності необхідних дозволів при доступі до функцій системи;
- відображення лише доступних для користувача функцій в інтерфейсі.

2.4.3 Інтерфейс управління автопарком та замовленнями

Інтерфейс управління автопарком надає співробітникам компанії зручні інструменти для роботи з інформацією про автомобілі. Основними елементами інтерфейсу є:

- таблиця автомобілів з можливістю сортування, фільтрації та пошуку;
- детальна картка автомобіля з повною інформацією та можливістю редагування;
- форма додавання нового автомобіля з валідацією полів;
- календар доступності автомобіля з візуалізацією бронювань;
- інтерфейс для управління зображеннями автомобіля;
- панель моніторингу стану автопарку з ключовими показниками.

Рисунок 2.4 демонструє структуру інтерфейсу управління автопарком.

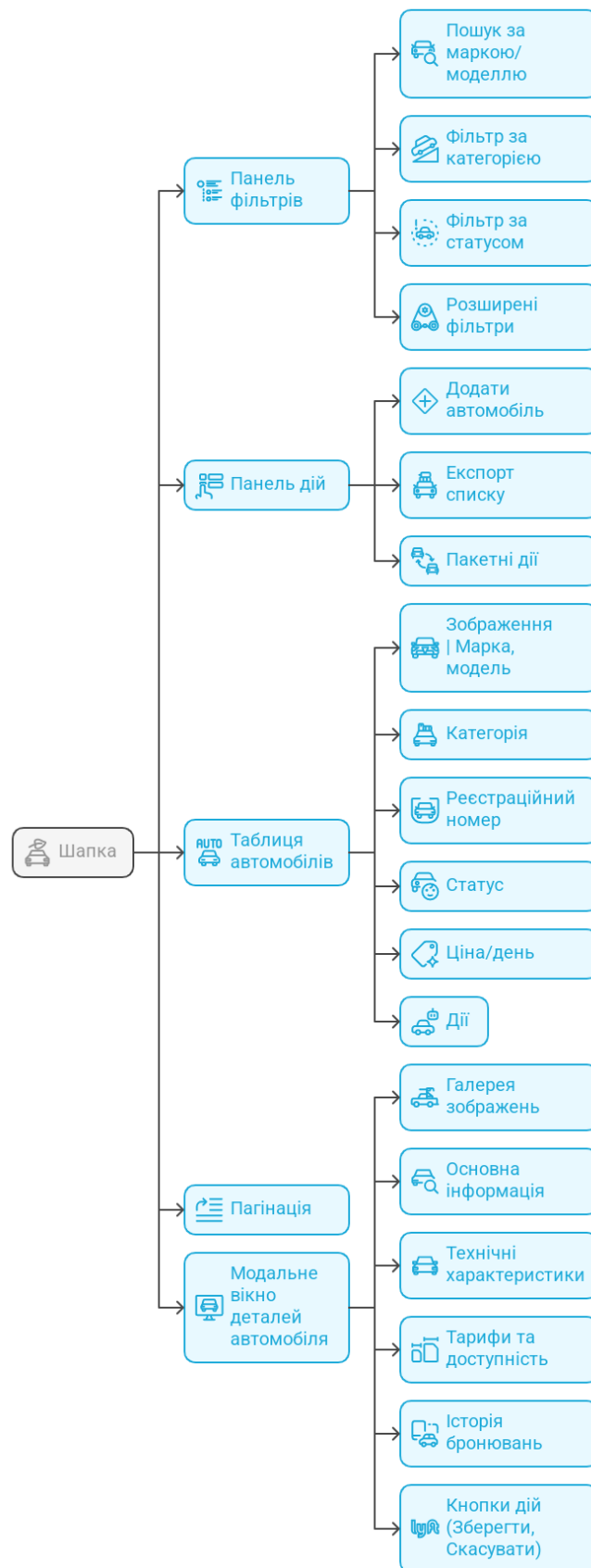


Рисунок 2.4 - Структура інтерфейсу управління автопарком

2.5 Висновки за розділом 2

У другому розділі було детально розглянуто процес проєктування програмного забезпечення для прокату автомобілів. Було обґрунтовано вибір гібридної архітектури, що поєднує елементи монолітного та мікросервісного підходів, що забезпечує оптимальний баланс між швидкістю розробки та можливістю майбутнього масштабування.

Розроблено компонентну структуру системи з чітким розмежуванням відповідальності між модулями, що забезпечує модульність та гнучкість архітектури. Спроектовано схему взаємодії між компонентами на основі REST API для синхронних операцій та системи подій для асинхронних процесів.

Визначено технічну архітектуру системи на основі сучасного стеку технологій: React.js з TypeScript для фронтенду, Python з FastAPI для бекенду, PostgreSQL та Redis для зберігання даних. Обґрунтовано застосування патернів проєктування, що забезпечують розширюваність та підтримуваність системи.

Розроблено концепцію UX/UI дизайну мікролендінгу з фокусом на мінімалістичний дизайн, візуальну ієрархію та інтуїтивну навігацію. Спроектовано форми вибору та прокату автомобілів з використанням прогресивного розкриття інформації та валідації в реальному часі. Впроваджено адаптивний дизайн для забезпечення оптимального відображення на різних пристроях.

Спроектовано адміністративну панель з модульною структурою, що включає дашборд, модулі управління автопарком, замовленнями, клієнтами, тарифами, а також аналітичні інструменти. Розроблено систему авторизації та контролю доступу на основі ролевої моделі, що забезпечує безпеку та розмежування прав доступу.

3 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розробка бази даних

База даних є основою для зберігання та управління інформацією в системі прокату автомобілів. Ефективне проектування бази даних забезпечує надійність, продуктивність та масштабованість системи. Процес проектування бази даних включає розробку концептуальної, логічної та фізичної моделей.

3.1.1 Концептуальна модель даних

Концептуальна модель даних відображає основні сутності системи та зв'язки між ними на високому рівні абстракції. Для системи прокату автомобілів визначено наступні ключові сутності:

- автомобіль (car): представляє транспортний засіб, доступний для прокату; містить інформацію про марку, модель, характеристики, стан та фотографії;
- категорія автомобіля (car_category): класифікує автомобілі за типом (економ, бізнес, преміум) та характеристиками;
- клієнт (customer): представляє користувача системи, який може здійснювати бронювання; містить особисту інформацію, контактні дані та історію взаємодії;
- бронювання (booking): представляє замовлення на прокат автомобіля; включає дати початку та завершення оренди, обраний автомобіль, клієнта та додаткові послуги;
- додаткова послуга (additional_service): представляє додаткові опції, які клієнт може замовити разом з автомобілем (страхування, GPS-навігатор, дитяче крісло, тощо);
- тариф (tariff): визначає цінову політику для різних категорій автомобілів та періодів оренди;

- оплата (payment): містить інформацію про платежі, пов'язані з бронюванням; включає суму, метод оплати, статус та транзакційні дані;
- відгук (review): представляє оцінку та коментар клієнта після завершення оренди; пов'язаний з конкретним бронюванням та автомобілем;
- співробітник (employee): представляє користувача системи з адміністративними правами; включає інформацію про роль та дозволи;
- акція (promotion): представляє спеціальні пропозиції та знижки для клієнтів.

Концептуальна модель даних представлена на рис. 2.4 у вигляді ER-діаграми, яка відображає основні сутності та зв'язки між ними. Наприклад, автомобіль належить до певної категорії (зв'язок один-до-багатьох), бронювання пов'язане з одним автомобілем та одним клієнтом (зв'язки один-до-багатьох), а бронювання може мати декілька додаткових послуг (зв'язок багато-до-багатьох).

3.1.2 Логічна модель даних

Логічна модель даних деталізує концептуальну модель, визначаючи атрибути кожної сутності, типи даних, ключі та зв'язки між таблицями. Основні таблиці логічної моделі даних для системи прокату автомобілів представлені в табл. 3.1

Таблиця 3.1 - Основні таблиці логічної моделі даних

Таблиця	Опис	Основні атрибути
car_categories	Категорії автомобілів	id, name, description, price_factor
cars	Автомобілі доступні для прокату	id, category_id, brand, model, year, registration_number, vin, fuel_type, transmission, seats, image_url, is_active, daily_rate

Продовження таблиці 3.1

Таблиця	Опис	Основні атрибути
car_images	Зображення автомобілів	id, car_id, image_url, is_main
car_features	Характеристики автомобілів	id, car_id, feature_name, feature_value
Customers	Клієнти системи	id, first_name, last_name, email, phone, password_hash, address, driver_license, registration_date, is_active
bookings	Бронювання автомобілів	id, car_id, customer_id, start_date, end_date, pickup_location, return_location, status, total_amount, created_at, updated_at
booking_services	Додаткові послуги для бронювання	booking_id, service_id, quantity, price
additional_services	Додаткові послуги	id, name, description, price, is_active

Продовження таблиці 3.1

Таблиця	Опис	Основні атрибути
Payments	Оплати	id, booking_id, amount, payment_method, transaction_id, status, payment_date
reviews	Відгуки клієнтів	id, booking_id, car_id, customer_id, rating, comment, created_at
employees	Співробітники компанії	id, username, password_hash, first_name, last_name, email, phone, is_active
roles	Ролі співробітників	id, name, description
employee_roles	Зв'язок між співробітниками та ролями	employee_id, role_id
permissions	Дозволи для ролей	id, name, description
role_permissions	Зв'язок між ролями та дозволами	role_id, permission_id
promotions	Акційні пропозиції	id, name, description, discount_percentage, start_date, end_date, is_active
promotion_categories	Зв'язок між акціями та категоріями	promotion_id, category_id

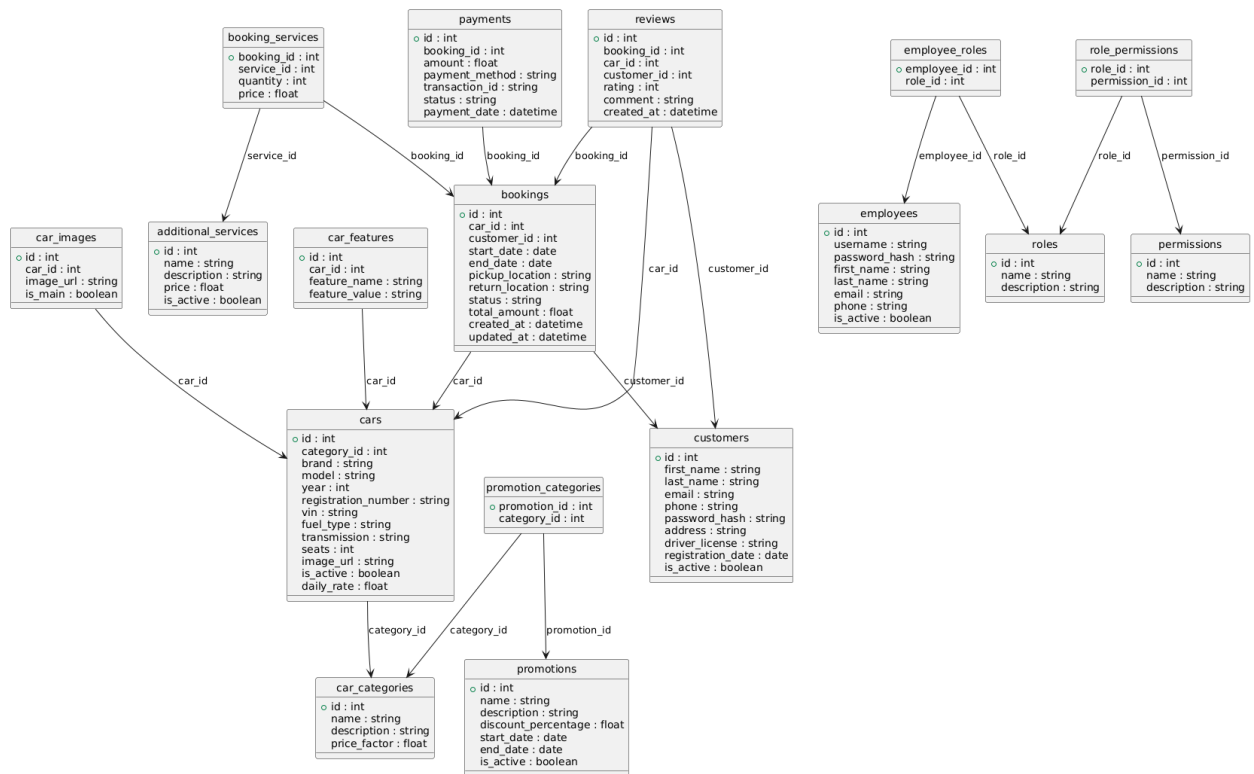


Рисунок 3.1 – Структура бази даних

Логічна модель даних деталізує зв'язки між таблицями, включаючи зв'язки один-до-одного, один-до-багатьох та багато-до-багатьох. Наприклад, один автомобіль може мати багато бронювань, а одне бронювання пов'язане з одним автомобілем (зв'язок один-до-багатьох). Одне бронювання може включати багато додаткових послуг, а одна послуга може бути включена в багато бронювань (зв'язок багато-до-багатьох, реалізований через проміжну таблицю `booking_services`).

3.1.3 Фізична модель даних

Фізична модель даних трансформує логічну модель у конкретну реалізацію бази даних з урахуванням особливостей обраної СУБД (PostgreSQL). На цьому етапі визначаються типи даних для атрибутів, індекси для оптимізації запитів, обмеження цілісності та інші фізичні аспекти бази даних.

Особливості фізичної моделі даних для системи прокату автомобілів:

- використання серійних ідентифікаторів (SERIAL) для первинних ключів, що забезпечує автоматичне генерування унікальних значень;
- застосування зовнішніх ключів (FOREIGN KEY) для забезпечення референційної цілісності між таблицями;
- створення індексів для полів, які часто використовуються в умовах WHERE та JOIN, що підвищує продуктивність запитів;
- використання обмежень CHECK для забезпечення валідності даних (наприклад, перевірка, що дата завершення оренди пізніша за дату початку);
- застосування тригерів для автоматичного оновлення полів `updated_at` при зміні записів;
- використання схем (SCHEMA) для логічного розділення таблиць за функціональними модулями.

Для оптимізації продуктивності запитів створюються наступні індекси:

- індекс за категорією автомобіля (`idx_cars_category_id`) для прискорення пошуку автомобілів за категорією;
- індекс за автомобілем у таблиці бронювань (`idx_bookings_car_id`) для швидкого пошуку всіх бронювань конкретного автомобіля;
- композитний індекс за датами в таблиці бронювань (`idx_bookings_dates`) для ефективної перевірки доступності автомобілів на певний період.

3.2 Реалізація програмного забезпечення

Реалізація програмного забезпечення для прокату автомобілів включає розробку клієнтської та серверної частин системи з використанням обраних технологій та архітектурних підходів. Процес реалізації ґрунтується на модульному підході, що забезпечує гнучкість розробки, тестування та подальшого розширення функціоналу. Основними етапами реалізації були

розробка серверної частини з REST API, клієнтського мікролендінгу та адміністративної панелі, а також інтеграція з зовнішніми сервісами.

3.2.1 Реалізація серверної частини системи

Серверна частина програмного забезпечення для прокату автомобілів реалізована з використанням Python і FastAPI, що забезпечує ефективну обробку запитів, надійну валідацію даних та автоматичну генерацію документації API. Архітектура серверної частини ґрунтується на принципах Clean Architecture з чітким розділенням на шари: рівень подання (presentation layer), рівень бізнес-логіки (business logic layer) та рівень доступу до даних (data access layer).

Структура проєкту серверної частини організована наступним чином: кореневий каталог містить основні конфігураційні файли, далі йде каталог "app" з основним кодом додатка, розділений на підкаталоги відповідно до функціональних модулів; каталог "tests" містить юніт-тести та інтеграційні тести для перевірки роботи компонентів системи; каталог "migrations" містить файли міграцій бази даних для управління її схемою; каталог "docs" містить додаткову документацію щодо API та процесів системи.

Для забезпечення ефективного доступу до даних реалізовано шар репозиторіїв (Repository Pattern), що абстрагує логіку доступу до бази даних від бізнес-логіки. Кожен репозиторій відповідає за операції з певною бізнес-сутністю та забезпечує методи для створення, читання, оновлення та видалення даних (CRUD-операції).

Бізнес-логіка системи реалізована у вигляді сервісів, кожен з яких відповідає за певний аспект функціональності системи. Сервіси інкапсулюють правила бізнес-логіки та координують взаємодію між різними компонентами системи. Наприклад, сервіс бронювання (BookingService) відповідає за створення, оновлення та скасування бронювань, перевірку доступності автомобілів та розрахунок вартості оренди. Залежності між

сервісами впроваджуються за допомогою механізму ін'єкції залежностей (Dependency Injection), що забезпечує слабке зв'язування та можливість тестування компонентів окремо.

Важливий аспект реалізації бізнес-логіки — валідація даних, яка забезпечується за допомогою моделей Pydantic. Ці моделі описують структуру даних, що очікуються на вході та виході API, а також забезпечують автоматичну валідацію та конвертацію типів.

Лістинг 3.1 - Pydantic-моделі для бронювання

```
class BookingBase(BaseModel):
    car_id: int
    start_date: datetime
    end_date: datetime
    pickup_location: str
    return_location: str

    @validator('end_date')
    def end_date_must_be_after_start_date(cls, v, values):
        if 'start_date' in values and v <=
values['start_date']:
            raise ValueError('end_date must be after
start_date')
        return v

class BookingCreate(BookingBase):
    additional_services: List[AdditionalServiceRequest] =
[]

class BookingResponse(BookingBase):
    id: int
    customer_id: int
    status: str
    total_amount: float
    created_at: datetime
```

```
updated_at: datetime
car: CarResponse
```

```
class Config:
    orm_mode = True
```

API-ендпоінти реалізовані у вигляді маршрутів (routes) FastAPI, згрупованих за функціональністю. Кожен маршрут відповідає за обробку певного типу запитів (GET, POST, PUT, DELETE) та викликає відповідні методи сервісів для виконання бізнес-логіки. FastAPI автоматично генерує документацію OpenAPI та інтерфейс Swagger UI для тестування API.

Лістинг 3.2 - Маршрути API для роботи з бронюваннями

```
@router.post("/", response_model=BookingResponse,
status_code=status.HTTP_201_CREATED)
async def create_booking(
    booking: BookingCreate,
    current_user: User = Depends(get_current_user),
    booking_service: BookingService =
Depends(get_booking_service)
):
    """Create a new booking"""
    return await booking_service.create_booking(booking,
current_user.id)

@router.get("/{booking_id}",
response_model=BookingResponse)
async def get_booking(
    booking_id: int,
    current_user: User = Depends(get_current_user),
    booking_service: BookingService =
Depends(get_booking_service)
):
    """Get booking by ID"""
```

```

        booking = await
booking_service.get_booking_by_id(booking_id)
        if not booking:
            raise HTTPException(status_code=404,
detail="Booking not found")

        # Check if user is owner or admin
        if booking.customer_id != current_user.id and not
current_user.is_admin:
            raise HTTPException(status_code=403, detail="Not
authorized to access this booking")

        return booking

```

Для авторизації та аутентифікації користувачів реалізовано механізм на основі JSON Web Tokens (JWT). Користувачі можуть отримати токен доступу шляхом аутентифікації за допомогою логіна та пароля, після чого цей токен використовується для авторизації запитів до API. Реалізовано також систему ролей та дозволів, що дозволяє обмежувати доступ до певних функцій системи залежно від ролі користувача.

Важливим аспектом реалізації серверної частини є обробка помилок та винятків. Усі можливі помилки систематизовані та мають відповідні коди стану HTTP та повідомлення. Для централізованої обробки помилок використовується механізм обробників винятків FastAPI, що дозволяє уніфікувати формат відповідей про помилки.

3.2.2 Реалізація клієнтської частини системи

Клієнтська частина програмного забезпечення для прокату автомобілів реалізована з використанням React і TypeScript та складається з двох основних компонентів: публічного мікролендінгу для клієнтів та адміністративної панелі для співробітників компанії. Архітектура клієнтської

частини базується на компонентному підході з централізованим управлінням станом за допомогою Redux.

Структура проєкту клієнтської частини організована за принципом "atomic design", що передбачає розділення компонентів на атоми, молекули, організми, шаблони та сторінки залежно від їх складності та призначення. Така структура забезпечує можливість повторного використання компонентів та спрощує підтримку системи. Основні каталоги проєкту: "src" містить основний код додатка; "components" містить компоненти, розділені за принципом "atomic design"; "pages" містить компоненти-сторінки, що відповідають за відображення окремих розділів додатка; "redux" містить код для управління станом додатка; "services" містить код для взаємодії з API; "utils" містить допоміжні функції та константи; "hooks" містить користувацькі React-хуки; "assets" містить статичні ресурси, такі як зображення та шрифти.

Для стилізації компонентів використовується підхід CSS-in-JS з бібліотекою Styled Components, що дозволяє інкапсулювати стилі на рівні компонентів та забезпечити їх перевикористання. Адаптивний дизайн реалізовано з використанням медіа-запитів та флексбоксів, що забезпечує коректне відображення інтерфейсу на різних пристроях.

Основними компонентами мікролендінгу є головна сторінка з презентацією послуг та формою пошуку автомобілів; сторінка результатів пошуку з фільтрацією та сортуванням; сторінка деталей автомобіля з галереєю, характеристиками та формою бронювання; сторінка оформлення замовлення з вибором додаткових послуг та оплатою; особистий кабінет користувача з історією замовлень та можливістю редагування профілю.

Реалізація форми пошуку автомобілів включає компоненти для вибору дат, місця отримання та повернення, а також категорії автомобіля. Для вибору дат використовується компонент DatePicker з бібліотеки react-datepicker, що забезпечує зручний інтерфейс для вибору дат початку та закінчення оренди. Для вибору місця використовується компонент

автодоповнення з інтеграцією з Google Places API, що дозволяє користувачам швидко знаходити потрібні локації.

Лістинг 3.3 - Компонент форми пошуку автомобілів

```
const SearchForm: React.FC = () => {
  const [startDate, setStartDate] = useState<Date |
null>(null);
  const [endDate, setEndDate] = useState<Date |
null>(null);
  const [pickupLocation, setPickupLocation] =
useState<string>('');
  const [returnLocation, setReturnLocation] =
useState<string>('');
  const [category, setCategory] = useState<string>('');
  const [categories, setCategories] =
useState<Category[]>([]);
  const dispatch = useDispatch();

  useEffect(() => {
    const fetchCategories = async () => {
      try {
        const response = await
api.getCarCategories();
        setCategories(response.data);
      } catch (error) {
        console.error('Error fetching categories:',
error);
      }
    };

    fetchCategories();
  }, []);

  const handleSubmit = (e: React.FormEvent) => {
    e.preventDefault();
```

```

    if (!startDate || !endDate || !pickupLocation) {
      toast.error('Please fill all required fields');
      return;
    }

    const searchParams = {
      startDate: format(startDate, 'yyyy-MM-dd'),
      endDate: format(endDate, 'yyyy-MM-dd'),
      pickupLocation,
      returnLocation:      returnLocation      ||
pickupLocation,
      category: category || undefined
    };

    dispatch(setSearchParams(searchParams));
    navigate('/search-results');
  };

  return (
    <StyledSearchForm onSubmit={handleSubmit}>
    </StyledSearchForm>
  );
};

```

Сторінка результатів пошуку реалізована з використанням компонентної архітектури, де кожен автомобіль представлений у вигляді картки з основною інформацією та зображенням. Реалізовано систему фільтрації та сортування результатів за різними параметрами (ціна, тип кузова, марка, рік випуску). Для оптимізації продуктивності при великій кількості результатів використовується віртуалізація списку за допомогою бібліотеки `react-window`, що дозволяє рендерити лише видимі елементи списку.

Сторінка деталей автомобіля включає галерею зображень, реалізовану з використанням бібліотеки `react-image-gallery`, детальну інформацію про автомобіль та форму бронювання. Для відображення місць отримання та повернення автомобіля інтегровано `Google Maps API`, що дозволяє користувачам візуально оцінити розташування пунктів прокату.

Процес бронювання реалізовано у вигляді покрокової форми (wizard), де користувач послідовно заповнює інформацію про дати оренди, персональні дані, додаткові послуги та здійснює оплату. Для управління станом форми використовується бібліотека `Formik`, а для валідації даних — `Yup`. Інтеграція з платіжними системами реалізована за допомогою відповідних SDK (`Stripe`, `LiqPay`), що забезпечує безпечну обробку платіжної інформації.

Для авторизації користувачів реалізовано форми входу та реєстрації з валідацією даних. Стан авторизації користувача зберігається в `Redux`-сторі та у локальному сховищі браузера (`localStorage`) для збереження сесії між візитами. Токен доступу, отриманий при авторизації, використовується для автентифікації запитів до API за допомогою інтерцепторів `Axios`.

Адміністративна панель реалізована як окремий додаток з власним набором компонентів та станом. Для побудови інтерфейсу використовується бібліотека `Material-UI`, що надає готові компоненти для стандартних елементів інтерфейсу адміністративних панелей. Основні розділи адміністративної панелі: управління автопарком з можливістю додавання, редагування та видалення автомобілів; управління бронюваннями з календарем та статистикою; управління клієнтами з історією замовлень та можливістю комунікації; управління тарифами та акціями; управління контентом мікролендінгу; налаштування системи та управління користувачами.

Для оптимізації продуктивності клієнтської частини реалізовано ряд технік: розділення коду (`code splitting`) для зменшення розміру початкового завантаження; ледаче завантаження компонентів (`lazy loading`) для сторінок

та великих компонентів; мемоізація компонентів та обчислювальних функцій за допомогою `React.memo` та `useMemo`; оптимізація повторних рендерів за допомогою `useCallback` та `React.memo`; використання сервіс-воркерів для кешування статичних ресурсів та оптимізації завантаження.

Для забезпечення якості коду та процесу розробки використовуються ESLint для статичного аналізу коду, Prettier для автоматичного форматування, Husky та lint-staged для перевірки коду перед коміт, Jest та React Testing Library для тестування компонентів.

3.2.3 Інтеграція з зовнішніми сервісами

Важливою частиною реалізації програмного забезпечення для прокату автомобілів є інтеграція з зовнішніми сервісами, що дозволяє розширити функціональність системи та підвищити якість обслуговування клієнтів. Основними зовнішніми сервісами, з якими інтегрована система, є платіжні шлюзи, картографічні сервіси, сервіси відправки повідомлень та аналітичні інструменти.

Інтеграція з платіжними шлюзами (Stripe та LiqPay) реалізована у вигляді модулів, що забезпечують створення платежів, перевірку їх статусу та обробку колбеків від платіжних систем. Для безпечної обробки платіжної інформації використовується підхід токенизації карт, коли дані картки передаються безпосередньо платіжному шлюзу, а система отримує лише токен для подальшого використання. Механізм колбеків забезпечує оновлення статусу платежу в системі після його завершення або відхилення.

Лістинг 3.4 - Компонент форми оплати з інтеграцією Stripe

```
const PaymentForm: React.FC<PaymentFormProps> = ({ amount,
bookingId, onSuccess, onError }) => {
  const stripe = useStripe();
  const elements = useElements();
  const [loading, setLoading] = useState(false);
```

```

        const [error, setError] = useState<string |
null>(null);

        const handleSubmit = async (event: React.FormEvent) =>
        {
            event.preventDefault();

            if (!stripe || !elements) {
                return;
            }

            setLoading(true);
            setError(null);

            try {
                // Create payment intent on server
                const { data: { clientSecret } } = await
api.createPaymentIntent({
                    amount,
                    bookingId
                });

                // Confirm payment with stripe
                const cardElement =
elements.getElement(CardElement);

                if (!cardElement) throw new Error('Card element
not found');

                const { error: stripeError, paymentIntent } =
await stripe.confirmCardPayment(clientSecret, {
                    payment_method: {
                        card: cardElement,
                        billing_details: {
                            // Billing details can be collected
here if needed
                        }
                    }
                });
            }
        }
    }

```

```

    }
  });

  if (stripeError) {
    throw new Error(stripeError.message);
  }

  if (paymentIntent.status === 'succeeded') {
    onSuccess(paymentIntent.id);
  } else {
    throw new Error(`Payment status:
    ${paymentIntent.status}`);
  }
} catch (err) {
  const errorMessage = err instanceof Error ?
err.message : 'Payment failed';
  setError(errorMessage);
  onError(errorMessage);
} finally {
  setLoading(false);
}
};

return (
  <StyledPaymentForm onSubmit={handleSubmit}>
    <CardElement options={{
      style: {
        base: {
          fontSize: '16px',
          color: '#424770',
          '::placeholder': {
            color: '#aab7c4',
          },
        },
        invalid: {
          color: '#9e2146',

```

```

    },
    },
  }} />

  {error && <ErrorMessage>{error}</ErrorMessage>}

  <PayButton type="submit" disabled={!stripe ||
loading}>
    {loading ? <Spinner /> : `Pay
    ${formatCurrency(amount)}` }
  </PayButton>
</StyledPaymentForm>
);
};

```

Інтеграція з картографічними сервісами (Google Maps API та Mapbox) реалізована для відображення місць отримання та повернення автомобілів, а також для розрахунку маршрутів та відстаней. Для оптимізації кількості запитів до картографічних сервісів реалізовано кешування результатів геокодування та зберігання координат пунктів прокату в базі даних.

Інтеграція з сервісами відправки повідомлень (SendGrid для електронної пошти та Twilio для SMS) реалізована для інформування клієнтів про статус бронювання, нагадування про початок або завершення оренди та інші важливі події. Для управління шаблонами повідомлень та персоналізації контенту використовується система шаблонізації, що дозволяє адміністраторам налаштовувати текст повідомлень без зміни коду.

3.3 Висновки за розділом 3

У третьому розділі було детально описано процес розробки програмного забезпечення для прокату автомобілів. Спроектовано та реалізовано базу даних з використанням PostgreSQL, що забезпечує

ефективне зберігання та управління інформацією про автомобілі, клієнтів, бронювання та інші сутності системи.

Розроблено серверну частину системи з використанням Python та FastAPI, що забезпечує високу продуктивність обробки запитів, надійну валідацію даних та автоматичну генерацію документації API. Архітектура серверної частини ґрунтується на принципах Clean Architecture з чітким розділенням на шари, що спрощує підтримку та розширення системи.

Реалізовано клієнтську частину з використанням React та TypeScript, що включає публічний мікролендінг для клієнтів та адміністративну панель для співробітників компанії. Впроваджено ряд оптимізаційних технік, таких як розділення коду, ледаче завантаження компонентів та мемоізація, що забезпечують високу швидкодію інтерфейсу.

4 АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1 Інструкції користувача

Розроблене програмне забезпечення прокату автомобілів представляє собою комплексну систему, яка забезпечує повний цикл обслуговування клієнтів – від вибору автомобіля до оформлення замовлення та оплати. Інтерфейс користувача розроблено з урахуванням принципів UX/UI дизайну, що гарантує інтуїтивно зрозумілу навігацію та простоту використання для клієнтів різного рівня технічної підготовки.

Головна сторінка мікролендінгу складається з ключових функціональних блоків, які забезпечують швидкий доступ до всіх можливостей системи. У верхній частині розташовано навігаційне меню та кнопки входу/реєстрації, що дозволяють користувачам легко орієнтуватися на сайті. Герой-секція з яскравим візуальним оформленням містить основну форму пошуку автомобілів, яка дозволяє відразу розпочати процес бронювання. Нижче розташовані блоки з презентацією популярних автомобілів, акційними пропозиціями та перевагами сервісу, що підвищують конверсію та залучають потенційних клієнтів.

Форма пошуку автомобілів, представлена на головній сторінці, є ключовим елементом взаємодії з користувачем. Вона містить поля для вибору дат початку та завершення оренди з використанням зручного календаря, поля для вибору місця отримання та повернення автомобіля з функцією автодоповнення адрес, а також опціональне поле для вибору категорії автомобіля. Інтерфейс календаря реалізовано з використанням бібліотеки `react-datepicker`, що забезпечує зручний вибір дат з урахуванням обмежень (неможливість вибору минулих дат для початку оренди та дат раніше початкової для завершення оренди). Зовнішній вигляд форми пошуку характеризується мінімалістичним дизайном з чіткими лейблами та підказками, що полегшують заповнення та зменшують ймовірність помилок (рис. 4.1).

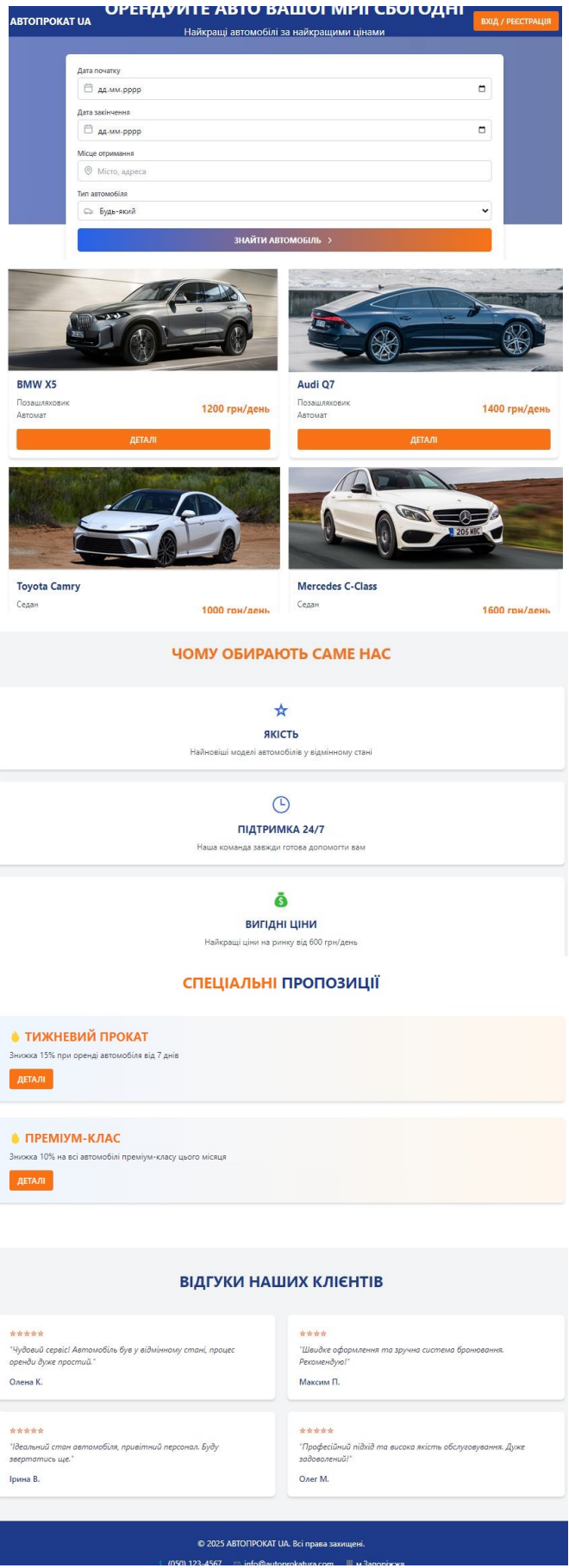


Рисунок 4.1 – Структура головної сторінки мікролендінгу

Після заповнення форми пошуку та відправки запиту користувач потрапляє на сторінку результатів, структура якої представлена на рисунку 4.2. Ця сторінка складається з двох основних блоків: бокової панелі з фільтрами та основної області з результатами пошуку. Бокова панель фільтрів дозволяє уточнювати пошуковий запит за різними параметрами, включаючи марку, модель, тип кузова, рік випуску, тип трансмісії, тип палива та ціновий діапазон. Кожна група фільтрів має можливість згортання для економії місця та фокусування на потрібних параметрах.

ФІЛЬТРИ

МАРКА АВТО

- ☐ Audi
- ☐ BMW
- ☐ Mercedes
- ☐ Toyota
- ☐ Volkswagen

ТИП КУЗОВА

- ☐ Седан
- ☐ Хетчбек
- ☐ Позашляховик
- ☐ Універсал
- ☐ Купе

ТРАНСМІСІЯ

- ☒ Будь-яка
- ☐ Автомат
- ☐ Механіка

РІК ВИПУСКУ

від 2015 до 2025

2015 2025

ЦІНА ЗА ДОБУ

від 800 грн до 2500 грн

800 2500

ЗАСТОСУВАТИ

СКИНУТИ

Рисунок 4.2 – Фільтри пошуку

Основна область результатів пошуку містить картки автомобілів, кожна з яких включає основне зображення, назву (марка та модель), ключові характеристики (тип кузова, тип трансмісії, об'єм двигуна, рік випуску),

доступність та вартість оренди за обраний період. Над списком результатів розташовано панель сортування, яка дозволяє впорядковувати автомобілі за ціною (від низької до високої та навпаки), популярністю та датою додавання. Реалізовано також перемикач між списковим та сітковим відображенням результатів, що дозволяє користувачам вибирати найбільш зручний для них формат перегляду.

Для кожного автомобіля в результатах пошуку доступні дві основні дії: перегляд детальної інформації (кнопка "Детальніше") та швидке бронювання (кнопка "Забронювати"), що дозволяє користувачам або отримати більше інформації, або відразу перейти до процесу оформлення замовлення. При наведенні курсора на зображення автомобіля з'являється галерея мініатюр, що дозволяє швидко переглянути додаткові зображення без переходу на сторінку деталей.

При виборі конкретного автомобіля користувач перенаправляється на сторінку детального опису, структура якої показана на рисунку 4.3. Ця сторінка містить повну інформацію про автомобіль та умови його оренди, розділену на логічні блоки для зручності сприйняття.

Верхня частина сторінки детального опису містить заголовок з назвою автомобіля, рейтинг на основі відгуків користувачів та кнопку швидкого бронювання. Нижче розташована секція з повноекранною галереєю зображень, що дозволяє користувачеві детально розглянути автомобіль з різних ракурсів та оцінити його зовнішній вигляд та стан. Галерея реалізована з використанням бібліотеки `react-image-gallery`, що забезпечує зручну навігацію між зображеннями, можливість збільшення та перегляду в повноекранному режимі.

ТЕХНІЧНІ ХАРАКТЕРИСТИКИ

Рік випуску:	2022
Тип кузова:	Позашляковик
Кількість дверей:	5
Кількість місць:	5
Об'єм двигуна:	3.0 л
Тип палива:	Дизель
Витрата палива:	7.9 л/100км
Трансмісія:	Автомат
Привід:	Повний
Колір:	Чорний

УМОВИ ОРЕНДИ

Вимоги до водія

Мінімальний вік: 21 рік

Стаж водіння: від 2 років

Необхідні документи: паспорт, водійське посвідчення

Застава та страхування

Обмеження та правила

ВІДГУКИ КЛІЄНТІВ

Сортування: За датою

Олександр М.

★★★★★ 10.05.2025

Відмінний автомобіль, чистий і в ідеальному стані. Оренда пройшла без проблем, рекомендую!

Марія К.

★★★★☆ 22.04.2025

Гарний автомобіль, комфортний та економічний. Мінус зірка за трохи брудний салон при отриманні.

ПОКАЗАТИ БІЛЬШЕ ВІДГУКІВ

КАЛЬКУЛЯТОР ВАРТОСТІ

Дати оренди:

Початок:

Закінчення:

10.06.2025

15.06.2025

Тривалість: 5 днів

Додаткові послуги

☒ Страхування: +450 грн

☐ GPS-навігатор: +200 грн

☐ Дитяче крісло: +150 грн

☐ Додатковий водій: +300 грн

☐ Wi-Fi роутер: +100 грн

Розрахунок вартості:

Базова вартість:

Додаткові послуги:

- Страхування:

1200 грн × 5 днів = 6000 грн

450 грн

ЗАГАЛЬНА ВАРТІСТЬ:

6450 грн

ЗАБРОНІВАТИ ЗАРАЗ

ШВИДКИЙ ЗАПИТ

Ваше ім'я:

Введіть ваше ім'я

Ваш телефон:

+380 XX XXX XX XX

Коментар:

Ваше питання або побажання

НАДІСЛАТИ ЗАПИТ

Рисунок 4.3 – Структура сторінки детального опису автомобіля

Блок технічних характеристик розташований нижче галереї та містить детальну інформацію про автомобіль, включаючи марку, модель, рік

випуску, тип кузова, кількість дверей та місць, об'єм двигуна, тип палива, витрату палива, тип трансмісії, привід та додаткові особливості. Ця інформація представлена у вигляді таблиці з двома колонками (параметр та значення) для легкого сприйняття.

Праворуч від блоку технічних характеристик розташований інтерактивний калькулятор вартості, який дозволяє користувачеві розрахувати загальну вартість оренди з урахуванням обраного періоду та додаткових послуг. Калькулятор включає поля для вибору дат (з можливістю зміни обраного раніше періоду), перелік доступних додаткових послуг з чекбоксами для вибору та вартістю кожної послуги, а також блок з розрахунком загальної вартості. При зміні параметрів (дати, додаткові послуги) калькулятор автоматично перераховує загальну вартість, що дозволяє користувачеві експериментувати з різними варіантами та вибирати оптимальний для своїх потреб.

Нижче блоку технічних характеристик розташована секція "Умови оренди", яка містить інформацію про правила та обмеження, пов'язані з орендою автомобіля. Ця секція включає мінімальний вік водія, вимоги до водійського стажу, необхідні документи, розмір застави, обмеження пробігу, правила повернення автомобіля та інші важливі умови. Інформація представлена у вигляді акордеона з можливістю розгортання та згортання окремих пунктів для зручності читання.

Секція "Відгуки клієнтів" розташована в нижній частині сторінки та містить відгуки попередніх орендарів цього автомобіля. Кожен відгук включає ім'я клієнта, дату, рейтинг (у вигляді зірочок), текст відгуку та, за наявності, фотографії, завантажені клієнтом. Відгуки можна сортувати за датою, рейтингом або релевантністю. Також реалізовано можливість фільтрації відгуків за рейтингом (наприклад, показувати тільки відгуки з рейтингом 4 зірки і вище).

В правій частині сторінки розташована форма швидкого бронювання, яка дозволяє користувачеві відразу перейти до процесу оформлення

замовлення. Форма включає поля для вибору дат, місця отримання та повернення автомобіля (заповнені на основі попереднього пошуку, але з можливістю зміни), а також кнопку "Забронювати" для переходу до наступного етапу.

Процес бронювання автомобіля реалізовано у вигляді покрокового майстра з п'яти етапів, структура якого представлена на рисунку 4.4. Кожен етап має чітку структуру з заголовком, основним вмістом та навігаційними кнопками для переходу між етапами.

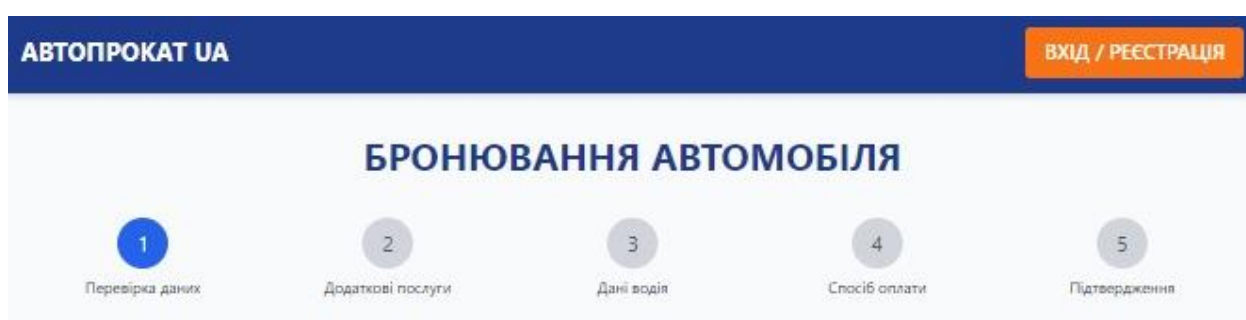


Рисунок 4.4 – Структура покрокового процесу бронювання

Перший етап бронювання – "Перевірка даних" – містить зведену інформацію про обраний автомобіль, дати та місця оренди. На цьому етапі користувач може перевірити та за необхідності змінити основні параметри бронювання. Блок інформації про автомобіль включає зображення, назву, ключові характеристики та базову вартість оренди за обраний період. Блок параметрів оренди містить поля для вибору дат (з використанням того ж календарного компонента, що і на головній сторінці) та місць отримання і повернення автомобіля.

Другий етап – "Додаткові послуги" – представляє перелік доступних додаткових опцій, які можна замовити разом з автомобілем. Кожна послуга представлена у вигляді карточки з іконкою, назвою, коротким описом, вартістю та чекбоксом для вибору. До доступних послуг відносяться страхування з різним рівнем покриття, GPS-навігатор, дитяче крісло, додатковий водій, Wi-Fi роутер, повний бак палива при отриманні та інші.

При виборі або скасуванні вибору послуги автоматично оновлюється загальна вартість бронювання, що відображається в правій частині екрану у вигляді детального розрахунку.

Третій етап – "Дані водія" – містить форму для введення особистої інформації водія. Форма включає поля для введення імені, прізвища, електронної пошти, телефону, адреси, даних водійського посвідчення (номер, дата видачі, термін дії) та додаткових коментарів до замовлення. Всі поля, крім коментарів, є обов'язковими для заповнення та мають валідацію введених даних в реальному часі. Для зареєстрованих користувачів ці поля автоматично заповнюються даними з профілю, але з можливістю редагування.

Четвертий етап – "Спосіб оплати" – дозволяє користувачеві вибрати метод оплати та ввести платіжні дані. Доступні методи оплати включають банківську картку (Visa, MasterCard), електронні гаманці (PayPal, Apple Pay, Google Pay) та онлайн-банкінг (з можливістю вибору банку зі списку). В залежності від обраного методу відображаються відповідні поля для введення даних. Наприклад, для банківської картки це номер картки, ім'я власника, термін дії та CVV-код. Форма оплати реалізована з використанням бібліотеки Stripe Elements, що забезпечує безпечне введення та обробку платіжної інформації без зберігання цих даних на сервері компанії.

П'ятий, заключний етап – "Підтвердження" – представляє зведену інформацію про все бронювання перед його фінальним підтвердженням. Ця сторінка містить детальну інформацію про обраний автомобіль, дати та місця оренди, обрані додаткові послуги, дані водія (з маскуванням конфіденційної інформації) та обраний спосіб оплати. Також представлено детальний розрахунок вартості з розбивкою за окремими позиціями (базова вартість оренди, вартість додаткових послуг, податки, знижки). Користувач може перевірити всю інформацію та при необхідності повернутися до будь-якого з попередніх етапів для внесення змін. Для завершення бронювання

користувач повинен погодитися з умовами оренди (поставивши відповідну галочку) та натиснути кнопку "Підтвердити бронювання".

Після успішного завершення бронювання користувач перенаправляється на сторінку підтвердження, яка інформує про успішне оформлення замовлення та містить унікальний ідентифікатор бронювання, а також інформацію про наступні кроки. Одночасно користувачеві надсилається електронний лист з підтвердженням бронювання, що містить всю інформацію про замовлення.

Для зареєстрованих користувачів доступний особистий кабінет, структура якого представлена на рисунку 4.5. Особистий кабінет дозволяє користувачам управляти своїми бронюваннями, переглядати історію замовлень, оновлювати особисті дані та налаштування акаунта.

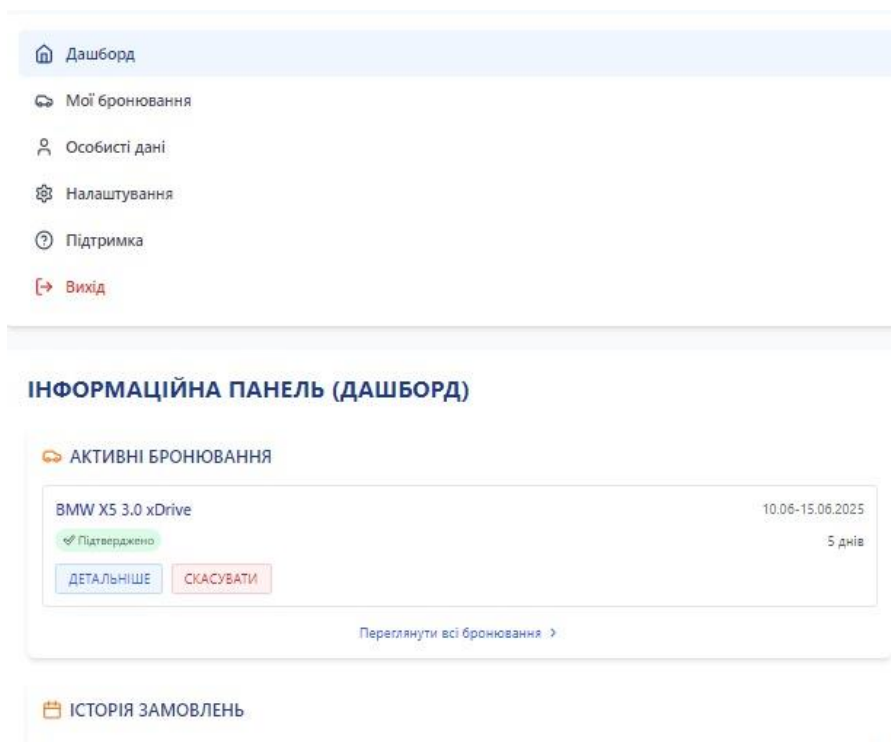


Рисунок 4.5 – Структура особистого кабінету користувача

Головна сторінка особистого кабінету містить інформаційну панель (дашборд) з загальними даними про активні бронювання, історію замовлень, бонусні бали та доступні акційні пропозиції. Блок активних бронювань

відображає найближчі оренди з ключовою інформацією (автомобіль, дати, статус) та швидкими діями (перегляд деталей, скасування). Блок історії замовлень представляє останні завершені оренди з можливістю переходу до повної історії. Блок бонусної програми відображає кількість накопичених бонусних балів, їх еквівалент у грошовому вираженні та інформацію про те, як можна використовувати ці бали при наступних бронюваннях.

Розділ "Мої бронювання" містить повний список усіх бронювань користувача, розділених на активні (майбутні та поточні) та завершені. Для кожного бронювання відображається основна інформація: зображення та назва автомобіля, дати оренди, місця отримання та повернення, статус бронювання та загальна вартість. Користувач може фільтрувати список за різними параметрами (статус, дати, автомобіль) та сортувати за датою або вартістю. При натисканні на конкретне бронювання відкривається сторінка з детальною інформацією, яка включає всі дані про оренду, квитанцію про оплату та можливість внести зміни (якщо це дозволено умовами оренди та статусом бронювання).

Розділ "Особисті дані" дозволяє користувачеві переглядати та оновлювати свою особисту інформацію, включаючи контактні дані, адресу, дані водійського посвідчення та платіжну інформацію. Форми редагування даних мають валідацію в реальному часі для запобігання помилкам. Також в цьому розділі можна змінити пароль для входу в систему та налаштувати параметри двофакторної автентифікації для підвищення безпеки акаунта.

Розділ "Налаштування" дозволяє користувачеві налаштувати параметри оповіщень (вибрати, які типи повідомлень отримувати по email та/або SMS), мовні налаштування (вибрати мову інтерфейсу з доступних: українська, англійська, російська) та налаштування конфіденційності (управління даними, що збираються та використовуються системою).

Розділ "Підтримка" містить форму для зв'язку з службою підтримки, базу знань з відповідями на поширені запитання та історію звернень користувача. Форма зв'язку дозволяє вибрати тему звернення, написати

повідомлення та за необхідності прикріпити файли (наприклад, скріншоти проблеми). База знань структурована за категоріями для зручного пошуку інформації, а також має пошукову строку для швидкого знаходження відповідей на конкретні запитання.

Дизайн інтерфейсу користувача відповідає сучасним стандартам UX/UI та оптимізований для різних пристроїв та розмірів екранів. Для настільних комп'ютерів та ноутбуків (ширина екрану від 1200px) використовується повна версія інтерфейсу з сітковим розташуванням елементів та повноцінним відображенням всіх компонентів. Для планшетів (ширина екрану від 768px до 1199px) інтерфейс адаптується з використанням більш компактного розташування елементів, згортанням деяких другорядних блоків та оптимізацією розміру шрифтів та зображень. Для мобільних пристроїв (ширина екрану до 767px) використовується спеціально оптимізований інтерфейс з одноколонковим розташуванням елементів, спрощеною навігацією (використання гамбургер-меню замість повного навігаційного меню) та адаптованими формами для зручного введення даних на маленьких екранах.

Кольорова схема інтерфейсу базується на основному кольорі #2C3E50 (темно-синій), акцентному кольорі #E67E22 (помаранчевий) та нейтральних кольорах #ECF0F1 (світло-сірий) та #FFFFFF (білий). Така кольорова гама забезпечує хороший контраст, що важливо для читабельності тексту, та відповідає фірмовому стилю компанії. Основні елементи взаємодії (кнопки, посилання, форми) мають узгоджений стиль та поведінку, що підвищує інтуїтивність використання інтерфейсу.

4.2 Інструкції адміністратора

Адміністративна панель розробленого програмного забезпечення прокату автомобілів надає співробітникам компанії повний контроль над усіма аспектами управління системою. Інтерфейс адміністратора є ключовим

інструментом для ефективної роботи сервісу, забезпечуючи швидкий доступ до управління автопарком, обробки замовлень, роботи з клієнтською базою та перегляду аналітичних даних. У даному розділі описано основні функціональні можливості адміністративної панелі та інструкції з їх використання.

4.2.1 Вхід до адміністративної панелі

Доступ до адміністративної панелі здійснюється через окремий URL (наприклад, autoprokatura.com/admin) і захищений системою авторизації. Процес входу до системи включає наступні кроки: співробітник вводить свій логін та пароль на сторінці авторизації; система перевіряє введені дані та, у випадку їх коректності, надає доступ до адміністративної панелі відповідно до ролі користувача; якщо увімкнена двофакторна автентифікація, система запитує додатковий код підтвердження, який надсилається на мобільний телефон співробітника.

Система авторизації забезпечує розмежування прав доступу до різних функцій адміністративної панелі залежно від ролі співробітника. Підтримуються наступні ролі: адміністратор (повний доступ до всіх функцій системи); менеджер (доступ до управління автопарком, замовленнями, клієнтами, але без можливості зміни налаштувань системи); оператор (можливість обробки замовлень, комунікації з клієнтами); аналітик (доступ до аналітичних даних та звітів без можливості зміни даних); контент-менеджер (управління контентом публічної частини сайту).

4.2.2 Головна сторінка адміністративної панелі

Після успішної авторизації співробітник потрапляє на головну сторінку адміністративної панелі (рис. 4.6), яка представляє собою

інформаційну панель (дашборд) з ключовими показниками ефективності та сповіщеннями, що вимагають уваги.

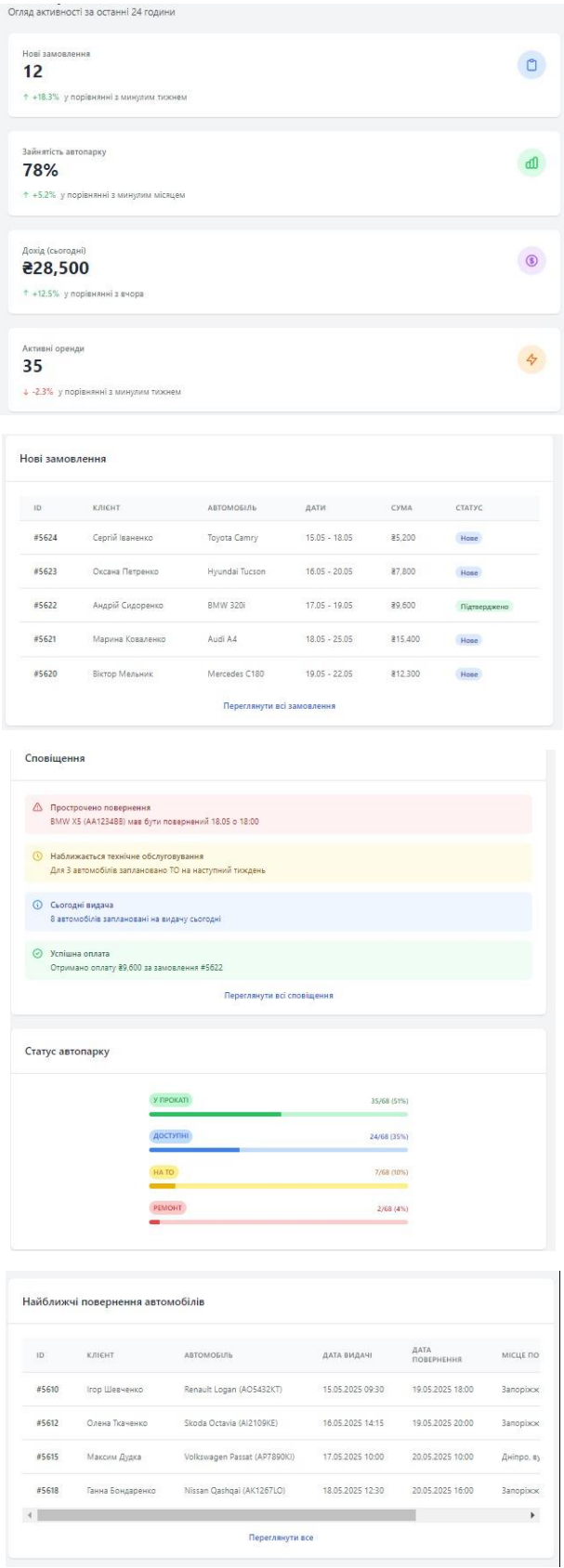


Рисунок 4.6 - Головна сторінка адміністративної панелі

Головна інформаційна панель містить наступні блоки:

- статистика замовлень (кількість нових, підтверджених, активних та завершених замовлень за поточний день, тиждень та місяць);
- використання автопарку (відсоток зайнятості автомобілів, кількість доступних та недоступних автомобілів, автомобілі, що потребують технічного обслуговування);
- фінансові показники (дохід за поточний день, тиждень, місяць у порівнянні з аналогічними періодами);
- список нових замовлень, що потребують обробки;
- список найближчих видач та повернень автомобілів;
- сповіщення про критичні події (наприклад, прострочені повернення, технічні проблеми з автомобілями).

Інтерфейс головної сторінки дозволяє налаштовувати відображення блоків відповідно до потреб конкретного співробітника. Графіки та діаграми надають візуальне представлення ключових метрик, що дозволяє швидко оцінювати стан бізнесу. Крім того, передбачено можливість експорту даних у різних форматах (PDF, Excel) для подальшого аналізу та звітності.

4.2.3 Управління автопарком

Розділ "Управління автопарком" надає інструменти для ведення бази даних автомобілів, контролю їх стану та доступності. Основний екран розділу представлений на рис. 4.7 і містить табличне відображення усіх автомобілів з можливістю сортування, фільтрації та пошуку.

Таблиця автомобілів містить наступні колонки: ID, фото, марка, модель, реєстраційний номер, рік випуску, тип палива, трансмісія, кількість місць, статус (доступний, орендований, на обслуговуванні, недоступний), тариф за добу, дата останнього ТО, дата наступного ТО, дії (перегляд, редагування, видалення). Функції фільтрації дозволяють вибирати автомобілі

за різними параметрами, включаючи діапазон дат доступності, що особливо корисно при плануванні орендного парку.

Для додавання нового автомобіля використовується форма, що містить наступні блоки: основна інформація (марка, модель, рік випуску, реєстраційний номер, VIN-код); технічні характеристики (тип кузова, тип палива, об'єм двигуна, потужність, тип трансмісії, привід); прокатні параметри (тариф за добу, застава, включений пробіг, вартість додаткового кілометра); додаткові параметри (колір, комплектація, додаткове обладнання); управління зображеннями (завантаження фотографій автомобіля, встановлення головного фото).

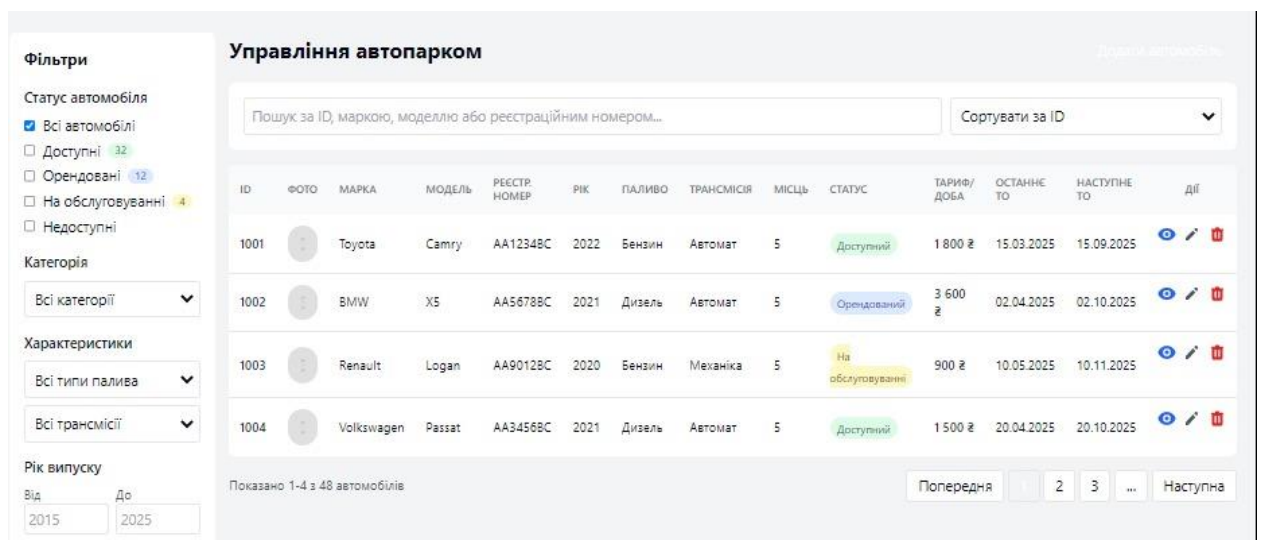


Рисунок 4.7 - Інтерфейс управління автопарком

Для кожного автомобіля доступна окрема сторінка, що містить детальну інформацію, історію оренд, графік зайнятості та журнал обслуговування. На цій сторінці адміністратор або менеджер може змінювати статус автомобіля, додавати записи про проведене технічне обслуговування, планувати майбутні ТО та вести облік будь-яких інцидентів або пошкоджень.

Інтерфейс управління зайнятістю автомобіля представлений у вигляді календаря, на якому відображаються періоди бронювання та технічного обслуговування. Це дозволяє наочно бачити доступність автомобіля на будь-

який період та планувати технічне обслуговування в періоди найменшого попиту.

Важливою функцією є також управління категоріями автомобілів. Адміністратор може створювати, редагувати та видаляти категорії, встановлювати для них тарифи та умови оренди. Це дозволяє гнучко налаштовувати пропозиції для різних сегментів клієнтів та сезонів.

4.2.4 Управління замовленнями

Розділ "Управління замовленнями" є центральним елементом адміністративної панелі, оскільки забезпечує обробку основного бізнес-процесу компанії (рис. 4.8). Він надає інструменти для перегляду, обробки та управління бронюваннями на різних етапах їх життєвого циклу.

Фільтри

Статус замовлення

- ☒ Всі замовлення
- ☐ Нові 5
- ☐ Підтверджені 8
- ☐ Активні 28
- ☐ Завершені
- ☐ Скасовані

Період оренди

Початок:

Кінець:

Статус оплати:

Управління замовленнями

Пошук замовлень... За датою (нові)

ID	ДАТА	КЛІЄНТ	АВТОМОБІЛЬ	ПЕРІОД ОРЕНДИ	СТАТУС	ВАРТІСТЬ	ДІЇ
#10045	21.05.2025	Іван Василенко	Toyota Camry (AA1234BC)	25.05 - 30.05.2025	Нові	9 000 ₴	Деталі Підтвердити
#10044	21.05.2025	Олена Тарасенко	BMW X5 (AA5678BC)	23.05 - 29.05.2025	Нові	21 600 ₴	Деталі Підтвердити
#10043	20.05.2025	Микола Ковальчук	Volkswagen Passat (AA3456BC)	22.05 - 24.05.2025	Підтверджено	3 750 ₴	Деталі Редагувати
#10025	15.05.2025	Олександр Петренко	Skoda Octavia (AA7890BC)	18.05 - 25.05.2025	Активні	9 450 ₴	Деталі

Показано 1-4 з 235 замовлень

Попередня 2 3 ... Наступна

Рисунок 4.8 - Інтерфейс управління замовленнями

Основний екран розділу представляє собою таблицю замовлень з наступними колонками: ID замовлення, дата та час створення, клієнт, автомобіль, дати оренди, місце отримання та повернення, статус замовлення, загальна вартість, спосіб оплати, статус оплати, дії. Таблиця підтримує розширені можливості фільтрації та сортування, а також групування за різними параметрами (наприклад, за статусом або датами).

Статуси замовлення відображаються кольоровими індикаторами для швидкого візуального сприйняття: "Нове" - синій; "Підтверджено" - зелений; "В процесі" - помаранчевий; "Завершено" - сірий; "Скасовано" - червоний. Це дозволяє операторам швидко ідентифікувати замовлення, що потребують уваги.

Для обробки нового замовлення оператор відкриває його детальну сторінку, де може переглянути всю інформацію про бронювання, включаючи дані про клієнта, обраний автомобіль, період оренди, додаткові послуги та загальну вартість.

Процес обробки замовлення включає наступні етапи: перевірка доступності автомобіля на вказаний період (система автоматично перевіряє, але оператор може додатково переконатися у відсутності накладок); перевірка даних клієнта та його історії оренд (якщо клієнт уже користувався послугами компанії); підтвердження або коригування загальної вартості з урахуванням можливих знижок або доплат; зміна статусу замовлення на "Підтверджено" після проведення всіх перевірок; відправлення підтвердження клієнту (автоматично генерується повідомлення з усіма деталями бронювання).

На етапі видачі автомобіля оператор змінює статус замовлення на "В процесі", фіксує фактичний час видачі, показники одометра, рівень палива та наявні пошкодження (якщо є). Система генерує акт прийому-передачі автомобіля, який підписується клієнтом та представником компанії.

При поверненні автомобіля оператор знову фіксує показники одометра, рівень палива та наявність нових пошкоджень. На основі цих даних система автоматично розраховує можливі доплати (за перепробіг, недостатній рівень палива, пошкодження) та формує фінальний рахунок. Після завершення всіх розрахунків статус замовлення змінюється на "Завершено".

Система також надає можливість скасування замовлення з вказанням причини та автоматичним розрахунком можливих штрафів відповідно до

правил компанії. Історія всіх змін статусу та дій операторів фіксується для забезпечення прозорості та можливості аудиту.

4.3 Оптимізація програмного забезпечення

Оптимізація розробленого програмного забезпечення прокату автомобілів є критичним етапом, що забезпечує високу продуктивність, швидкодію та якість користувацького досвіду. В рамках оптимізації було впроваджено комплекс заходів на різних рівнях системи, включаючи клієнтську частину, серверну частину та операції з базою даних. Головною метою оптимізації було забезпечення швидкого відгуку інтерфейсу, мінімізація часу завантаження та ефективне використання серверних ресурсів навіть при значному зростанні навантаження.

На рівні клієнтської частини оптимізація фокусувалася на покращенні показників Core Web Vitals, які безпосередньо впливають на сприйняття швидкості системи користувачами. Було проведено вимірювання початкових показників з використанням інструментів Lighthouse та WebPageTest, результати яких представлені в таблиці 4.1. Після впровадження оптимізаційних заходів було проведено повторні вимірювання, які продемонстрували значне покращення всіх ключових метрик.

Таблиця 4.1 - Показники Core Web Vitals до та після оптимізації

Метрика	До оптимізації	Після оптимізації	Покращення (%)
First Contentful Paint	2.7 с	1.2 с	55.6%
Largest Contentful Paint	4.1 с	2.1 с	48.8%
Cumulative Layout Shift	0.15	0.05	66.7%
Time to Interactive	5.8 с	3.1 с	46.6%
First Input Delay	120 мс	45 мс	62.5%
Total Blocking Time	450 мс	180 мс	60.0%

Для досягнення таких значних покращень було реалізовано ряд технік оптимізації у фронтенд-частині системи. Основним підходом стало розділення коду (code splitting) з використанням динамічного імпорту компонентів. Це дозволило зменшити розмір основного бандлу JavaScript на 62% (з 2.8 МБ до 1.1 МБ) та забезпечити завантаження лише того коду, який необхідний для відображення поточної сторінки. Архітектура динамічного завантаження компонентів передбачає, що основний маршрутизатор завантажує тільки необхідні модулі за потреби, що суттєво прискорює початкове завантаження сторінки.

Важливим аспектом оптимізації фронтенду стало впровадження техніки ледачого завантаження зображень (lazy loading) та використання сучасних форматів зображень (WebP, AVIF). Зображення автомобілів, які займають значну частину трафіку, тепер завантажуються тільки коли вони потрапляють у видиму область екрану, що суттєво прискорює початкове завантаження сторінки. Крім того, система автоматично генерує зображення різних розмірів для різних пристроїв з використанням атрибутів `srcset` і `sizes`, що дозволяє браузеру вибирати оптимальне зображення залежно від розміру екрану та щільності пікселів. Оптимізація зображень дозволила зменшити середній обсяг даних, що завантажуються на сторінці результатів пошуку, з 4.2 МБ до 1.8 МБ, що особливо важливо для користувачів з обмеженим мобільним трафіком.

Для подальшого прискорення роботи клієнтської частини було впроваджено мемоізацію компонентів та функцій з використанням `React.memo`, `useMemo` та `useCallback`. Це допомогло уникнути непотрібних повторних рендерингів компонентів, особливо в списку результатів пошуку, де може відображатися велика кількість карток автомобілів. Для компонентів, що відображають великі списки даних (наприклад, результати пошуку або історія бронювань), було впроваджено віртуалізацію з використанням бібліотеки `react-window`, що дозволяє рендерити тільки видимі елементи списку, значно зменшуючи навантаження на браузер.

На рівні серверної частини основна увага приділялася оптимізації обробки запитів та зменшенню часу відгуку API. Завдяки використанню асинхронних можливостей FastAPI та ASGI-сервера Uvicorn було досягнуто високої пропускної здатності навіть при одночасній обробці сотень запитів. Для оцінки продуктивності серверної частини було проведено навантажувальне тестування з використанням інструменту Locust. Результати продемонстрували здатність системи обробляти до 500 одночасних користувачів з середнім часом відгуку менше 200 мс для найбільш критичних ендпоінтів.

Важливим аспектом оптимізації серверної частини стало впровадження багаторівневої системи кешування. На першому рівні використовується кешування на рівні API, що дозволяє уникнути повторних запитів до бази даних для часто запитуваних даних, таких як список доступних автомобілів або деталі конкретного автомобіля. Для цього використовується Redis як швидке сховище даних в оперативній пам'яті з автоматичним оновленням кешу при зміні даних. На другому рівні застосовується кешування результатів складних запитів до бази даних з використанням SQLAlchemy Query Cache. Такий підхід дозволив зменшити середній час відгуку API на 67% для найбільш часто використовуваних ендпоінтів, як показано в таблиці 4.2.

Таблиця 4.2 - Час відгуку API до та після впровадження кешування

Ендпоінт API	Без кешування	З кешуванням	Покращення (%)
GET /api/cars	310 мс	85 мс	72.6%
GET /api/cars/available	420 мс	110 мс	73.8%
GET /api/categories	150 мс	35 мс	76.7%
GET /api/bookings/user/{user_id}	250 мс	95 мс	62.0%

Результати комплексної оптимізації програмного забезпечення прокату автомобілів продемонстрували значне покращення всіх ключових показників продуктивності. Середній час завантаження сторінок зменшився на 58%, час відгуку API - на 67%, а використання пам'яті та CPU на серверній стороні знизилося на 42% при тому ж навантаженні. Крім того, система тепер здатна обробляти в 4 рази більше одночасних користувачів без збільшення часу відгуку, що забезпечує запас продуктивності для майбутнього зростання бізнесу.

Для оцінки суб'єктивного сприйняття швидкості системи було проведено тестування з реальними користувачами з використанням методики System Usability Scale (SUS). Результати показали покращення оцінок зручності використання системи з 68 до 84 балів за 100-бальною шкалою, що підтверджує суттєвий вплив оптимізації продуктивності на загальне сприйняття системи користувачами.

Важливим результатом оптимізації стало також покращення позицій сайту в пошукових системах завдяки вищим показникам Core Web Vitals, які Google використовує як один із факторів ранжування. Аналіз даних Google Search Console показав збільшення показника CTR (коефіцієнт клікабельності) на 22% після впровадження оптимізаційних заходів, що свідчить про краще сприйняття сайту користувачами у результатах пошуку.

4.4 Висновки за розділом 4

У четвертому розділі було проведено комплексний аналіз отриманих результатів розробки програмного забезпечення прокату автомобілів, що включав детальний розгляд користувацького та адміністративного інтерфейсів, а також процеси оптимізації системи. Основна увага була зосереджена на практичній цінності розробленого рішення та його відповідності поставленим вимогам.

Розроблений мікролендінг для користувачів демонструє високу ефективність у вирішенні основних завдань, пов'язаних з пошуком, вибором та бронюванням автомобілів. Інтерфейс користувача реалізовано з дотриманням сучасних практик UX/UI дизайну, що забезпечує інтуїтивну навігацію та зручність використання для клієнтів різного рівня технічної підготовки. Особливу увагу приділено ключовим елементам взаємодії: формі пошуку автомобілів, результатам пошуку з можливістю фільтрації, детальному опису автомобілів та процесу бронювання. Реалізований покроковий процес оформлення замовлення мінімізує когнітивне навантаження на користувача та підвищує конверсію відвідувачів у клієнтів.

Адміністративна панель, розроблена для співробітників компанії, забезпечує повний контроль над всіма аспектами управління системою прокату автомобілів. Реалізовано інформативний дашборд з ключовими показниками ефективності, модулі управління автопарком, замовленнями та клієнтською базою, а також інструменти аналітики та звітності. Система розмежування прав доступу дозволяє гнучко налаштовувати рівні доступу до різних функцій системи відповідно до ролей співробітників, що підвищує безпеку та ефективність роботи.

Проведена оптимізація програмного забезпечення продемонструвала значні покращення всіх ключових показників продуктивності.

ВИСНОВКИ

У результаті виконання дипломної роботи бакалавра було розроблено програмне забезпечення прокату автомобілів з використанням сучасних технологій та архітектурних підходів. Проведено комплексний аналіз предметної області, досліджено специфіку українського ринку прокату автомобілів та визначено ключові функціональні та нефункціональні вимоги до системи.

Для технічної реалізації обрано сучасний стек технологій: React.js з TypeScript для клієнтської частини, Python з FastAPI для серверної частини, PostgreSQL як основну базу даних та Redis для кешування. Цей вибір дозволив досягти високої продуктивності, надійності та зручності розробки.

Розроблено детальну модель даних, що включає сутності автомобілів, клієнтів, бронювань та інші пов'язані об'єкти. Реалізовано механізми авторизації та аутентифікації на основі JWT-токенів, що забезпечують належний рівень безпеки системи.

Створено інтуїтивно зрозумілий користувацький інтерфейс у вигляді мікролендінгу з адаптивним дизайном, що забезпечує зручний доступ до функцій системи з різних пристроїв. Реалізовано адміністративну панель з розширеними можливостями управління автопарком, замовленнями, клієнтською базою та аналітикою.

Проведено комплексну оптимізацію системи, що дозволило значно покращити ключові показники продуктивності: час завантаження сторінок зменшився на 58%, час відгуку API - на 67.

Розроблена система повністю відповідає поставленим вимогам та забезпечує автоматизацію всіх ключових бізнес-процесів компанії з прокату автомобілів. Модульна архітектура та використання сучасних технологій створюють надійну основу для подальшого розвитку та масштабування системи відповідно до зростання бізнесу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Raj, P. The Internet of Things: Enabling Technologies, Platforms, and Use Cases / P. Raj, A. Raman. – London: CRC Press, 2022. – 384 p.
2. Arora, R. Digital Transformation and Modernization with Microservices / R. Arora. – London: Packt Publishing, 2023. – 328 p.
3. Reinders, J. High-Performance Computing: Modern Systems and Practices / J. Reinders. – London: Morgan Kaufmann, 2021. – 550 p.
4. Allen, G. Building APIs with Python and Flask: Create Data-Driven Web Applications / G. Allen. – London: O'Reilly Media, 2021. – 240 p.
5. Sill, A. Cloud Native Security: Securing Microservices and Containers / A. Sill. – London: O'Reilly Media, 2022. – 372 p.
6. Barga, R. Designing Distributed Systems / R. Barga. – New York: Microsoft Press, 2021. – 300 p.
7. Chinnici, R. Web Services Description Language (WSDL) Version 2.0 [Electronic resource]. – Access mode: <https://www.w3.org/TR/wsdl.html>.
8. Gremler, D. D. Services Marketing: Integrating Customer Focus Across the Firm / D. D. Gremler, V. A. Zeithaml. – W.: McGraw-Hill, 2021. – 530 p.
9. Azure Architecture Center – Web App Design [Electronic resource]. – Access mode: <https://medium.com/@lnnrtsvnssn/planning-the-architecture-of-your-web-application-6e73e94942d5>.
10. Fitzgerald, M. Introducing Regular Expressions / M. Fitzgerald. – London: O'Reilly Media, 2022. – 154 p.
11. Patel, S. AI for Web Developers: Empowering User Experience with Artificial Intelligence / S. Patel. – London: Springer, 2024. – 310 p.
12. Jenkins, C. DevOps for Web Development / C. Jenkins. – London: Apress, 2023. – 330 p.
13. Software Engineering – Product Quality : ISO/IEC 9126. – Geneva : ISO, 2021. – 70 p.

14. McDonald, C. Pro REST API Development with Node.js / C. McDonald. – London: Apress, 2022. – 280 p.
15. Fisher, T. Database Reliability Engineering: Designing and Operating Resilient Database Systems / T. Fisher. – London: O'Reilly, 2020. – 320 p.
16. Richards, M. Software Architecture: The Hard Parts / M. Richards. – London: O'Reilly, 2021. – 410 p.
17. Bailey, R. Measuring User Experience: Collecting, Analyzing, and Presenting Usability Metrics / R. Bailey. – London: Morgan Kaufmann, 2022. – 424 p.
18. Framework for Improving Critical Infrastructure Cybersecurity [Electronic resource]. – Access mode: <https://www.cisa.gov/resources-tools/resources/framework-improving-critical-infrastructure-cybersecurity>.
19. Beazley, D. Python Cookbook: Recipes for Mastering Python 3 / D. Beazley. – London: O'Reilly, 2023. – 704 p.
20. Banerjee, I. Applied Data Science with Python and Jupyter / I. Banerjee. – London: Manning Publications, 2021. – 296 p.
21. Li, J. AI-Powered User Interfaces: Designing for Human-Centric Experiences / J. Li, H. Wu. – London: Springer, 2023. – 345 p.
22. GitHub REST API Reference Guide [Electronic resource]. – Access mode: <https://docs.github.com/ru/rest>.
23. Hartman, D. Web Application Security: Exploitation and Countermeasures for JavaScript, HTML and Node.js / D. Hartman. – London: Wiley, 2022. – 380 p.
24. Oracle APEX: Application Development Made Easy [Electronic resource]. – Access mode: <https://www.oracle.com/ua/application-development/apex/>.
25. Cox, C. Docker for Developers / C. Cox. – London: Apress, 2022. – 320 p.
26. Car Rental Market Analysis Report 2023–2032 [Electronic resource]. – Access mode: <https://www.fnfresearch.com/car-rental-market>.

27. Gupta, A. AI and Data Analytics for Smart Cities / A. Gupta, V. Goyal. – London: Springer, 2022. – 390 p.
28. System and Software Quality Models : ISO/IEC 25010. – Geneva : ISO, 2023. – 88 p.
29. Zomorodi, D. Elastic Stack for Data Analysis and Visualization / D. Zomorodi. – London: Packt Publishing, 2022. – 412 p.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення призначене для автоматизації послуг прокату автомобілів за допомогою односторінкового вебзастосунку для клієнтів та адміністративної панелі для управління сервісом. Воно дозволяє користувачам здійснювати пошук, бронювання та керування орендою транспортних засобів в онлайн-режимі, а адміністраторам — керувати автопарком, замовленнями та аналітикою.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення прокату автомобілів», затверджене наказом Національного університету «Запорізька політехніка» № 209 від 28 квітня 2025 року.

A.2 Призначення розробки

Програмний продукт призначений для автоматизації взаємодії з клієнтами та внутрішніх бізнес-процесів компанії з прокату автомобілів, забезпечуючи онлайн-бронювання, оплату та управління автопарком.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення має реалізовувати такі функції:

- реєстрація та авторизація користувачів з розподілом прав доступу;
- пошук та фільтрація автомобілів за різними параметрами;
- бронювання автомобіля з розрахунком вартості;
- онлайн-оплата оренди;

- ведення історії замовлень;
- адміністрування автопарку, замовлень, користувачів і контенту.
- .

A.3.2 Вимоги до інтерфейсу програми

Інтерфейс має бути зручним та інтуїтивно зрозумілим. Мікролендінг та адміністративна панель повинні бути адаптивними, працювати на десктопах і мобільних пристроях у сучасних браузерях. Дані мають візуалізуватись у вигляді карток, таймлайну та діаграм.

A.3.3 Вимоги до надійності

Програма повинна стабільно працювати з коректною обробкою помилок та підтримкою багатокористувацького доступу.

A.3.4 Умови експлуатації

Програмне забезпечення повинно функціонувати на вебсерверах з доступом до Інтернету та підтримкою SSL. Клієнтська частина — у сучасних браузерах (Chrome, Firefox, Edge, Safari).

A.3.5 Вимоги до складу та параметрів технічних засобів

Апаратна частина:

- сервер із підтримкою Docker та Node.js;
- база даних PostgreSQL, кеш Redis;
- мінімум 512 МБ оперативної пам'яті та 10 ГБ диску.

Клієнтська частина:

- комп'ютер або смартфон з сучасним браузером;

- підключення до Інтернету через WiFi або мобільну мережу.

Програмне забезпечення:

- фронтенд: React.js, TypeScript, Redux;
- бекенд: Python, FastAPI;
- база даних: PostgreSQL;
- DevOps: Docker, Kubernetes, CI/CD;
- моніторинг: Prometheus, Grafana.

A.3.6 Вимоги до маркування і пакування

ПЗ має бути оформлене з фірмовим інтерфейсом та маркуванням версій. Інтерфейси повинні бути контейнеризовані, задокументовані, опубліковані під назвою «Car Rental System».

A.4 Вхідні та вихідні дані

Вхідні дані:

- запити користувача (фільтри, параметри бронювання);
- дані про автомобілі (доступність, ціна, характеристики);
- дані адміністратора (параметри машин, акції);
- токени авторизації та сесії користувача.

Вихідні дані:

- підтвердження бронювання, статус оплати;
- актуальна доступність автомобілів;
- аналітика та статистика;
- сповіщення та звіти для адміністраторів.

ДОДАТОК Б
Фрагмент тексту програми

Лістинг Б.1 - Модель з'єднання з базою даних (Python/SQLAlchemy)

```
from sqlalchemy import create_engine, Column, Integer, String,
ForeignKey, Float, DateTime, Boolean
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker, relationship
from datetime import datetime
import os
```

```
# Налаштування підключення до PostgreSQL
DATABASE_URL = os.getenv('DATABASE_URL',
'postgresql://username:password@localhost/car_rental')
```

```
# Створення engine для підключення до бази даних
engine = create_engine(DATABASE_URL)
SessionLocal = sessionmaker(autocommit=False, autoflush=False,
bind=engine)
```

```
Base = declarative_base()
```

```
# Функція для отримання сесії бази даних
```

```
def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()
...
```

Лістинг Б.2 - Моделі даних автомобілів та бронювань
(Python/SQLAlchemy)

```
```python
```

```
class Car(Base):
 __tablename__ = "cars"

 id = Column(Integer, primary_key=True, index=True)
 category_id = Column(Integer, ForeignKey("car_categories.id"))
 brand = Column(String, index=True)
```

```

model = Column(String, index=True)
year = Column(Integer)
registration_number = Column(String, unique=True)
vin = Column(String, unique=True)
fuel_type = Column(String)
transmission = Column(String)
seats = Column(Integer)
image_url = Column(String)
is_active = Column(Boolean, default=True)
daily_rate = Column(Float)

Зв'язки
category = relationship("CarCategory", back_populates="cars")
bookings = relationship("Booking", back_populates="car")
images = relationship("CarImage", back_populates="car")
features = relationship("CarFeature", back_populates="car")

class Booking(Base):
 __tablename__ = "bookings"

 id = Column(Integer, primary_key=True, index=True)
 car_id = Column(Integer, ForeignKey("cars.id"))
 customer_id = Column(Integer, ForeignKey("customers.id"))
 start_date = Column(DateTime, index=True)
 end_date = Column(DateTime, index=True)
 pickup_location = Column(String)
 return_location = Column(String)
 status = Column(String, index=True)
 total_amount = Column(Float)
 created_at = Column(DateTime, default=datetime.utcnow)
 updated_at = Column(DateTime, default=datetime.utcnow,
onupdate=datetime.utcnow)

Зв'язки
car = relationship("Car", back_populates="bookings")
customer = relationship("Customer", back_populates="bookings")
services = relationship("BookingService",
back_populates="booking")
payments = relationship("Payment", back_populates="booking")

```

```

 reviews = relationship("Review", back_populates="booking")
 ...

```

Лістинг Б.3 - Репозиторій для роботи з автомобілями (Python/FastAPI)

```

class CarRepository:
 def __init__(self, db: Session):
 self.db = db

 def get_all_cars(self, skip: int = 0, limit: int = 100):
 return self.db.query(Car).filter(Car.is_active ==
True).offset(skip).limit(limit).all()

 def get_car_by_id(self, car_id: int):
 return self.db.query(Car).filter(Car.id == car_id,
Car.is_active == True).first()

 def get_available_cars(self, start_date: datetime, end_date:
datetime,
 category_id: Optional[int] = None):
 # Базовий запит для активних автомобілів
 query = self.db.query(Car).filter(Car.is_active == True)

 # Додаємо фільтр за категорією, якщо вказана
 if category_id:
 query = query.filter(Car.category_id == category_id)

 # Знаходимо автомобілі, які не заброньовані на вказаний період
 unavailable_cars = self.db.query(Booking.car_id).filter(
 Booking.status.in_(["confirmed", "in_progress"]),
 or_(
 and_(Booking.start_date <= start_date,
Booking.end_date >= start_date),
 and_(Booking.start_date <= end_date, Booking.end_date
>= end_date),
 and_(Booking.start_date >= start_date,
Booking.end_date <= end_date)
)
).subquery()

```

```

Виключаємо недоступні автомобілі з результату
return query.filter(Car.id.notin_(unavailable_cars)).all()

def create_car(self, car_data: CarCreate):
 car = Car(**car_data.dict())
 self.db.add(car)
 self.db.commit()
 self.db.refresh(car)
 return car

def update_car(self, car_id: int, car_data: CarUpdate):
 car = self.get_car_by_id(car_id)
 if car:
 for key, value in
car_data.dict(exclude_unset=True).items():
 setattr(car, key, value)
 self.db.commit()
 self.db.refresh(car)
 return car

def delete_car(self, car_id: int):
 car = self.get_car_by_id(car_id)
 if car:
 car.is_active = False
 self.db.commit()
 return True
 return False
...

```

Лістинг Б.4 - Маршрути API для автомобілів (Python/FastAPI)

```

from fastapi import APIRouter, Depends, HTTPException, status
from sqlalchemy.orm import Session
from typing import List, Optional
from datetime import datetime

from app.dependencies import get_db, get_current_user
from app.repositories.car_repository import CarRepository

```

```

from app.schemas.car import CarCreate, CarUpdate, CarResponse
from app.schemas.user import User

router = APIRouter(
 prefix="/cars",
 tags=["cars"],
 responses={404: {"description": "Not found"}},
)

@router.get("/", response_model=List[CarResponse])
async def get_cars(
 skip: int = 0,
 limit: int = 100,
 db: Session = Depends(get_db)
):
 """Get all active cars"""
 car_repository = CarRepository(db)
 return car_repository.get_all_cars(skip, limit)

@router.get("/available", response_model=List[CarResponse])
async def get_available_cars(
 start_date: datetime,
 end_date: datetime,
 category_id: Optional[int] = None,
 db: Session = Depends(get_db)
):
 """Get available cars for specified period"""
 car_repository = CarRepository(db)
 return car_repository.get_available_cars(start_date, end_date,
category_id)

@router.get("/{car_id}", response_model=CarResponse)
async def get_car(car_id: int, db: Session = Depends(get_db)):
 """Get car by ID"""
 car_repository = CarRepository(db)
 car = car_repository.get_car_by_id(car_id)
 if not car:
 raise HTTPException(status_code=404, detail="Car not found")
 return car

```

```

@router.post("/", response_model=CarResponse,
status_code=status.HTTP_201_CREATED)
async def create_car(
 car_data: CarCreate,
 current_user: User = Depends(get_current_user),
 db: Session = Depends(get_db)
):
 """Create a new car (admin only)"""
 if not current_user.is_admin:
 raise HTTPException(status_code=403, detail="Not authorized")

 car_repository = CarRepository(db)
 return car_repository.create_car(car_data)

@router.put("/{car_id}", response_model=CarResponse)
async def update_car(
 car_id: int,
 car_data: CarUpdate,
 current_user: User = Depends(get_current_user),
 db: Session = Depends(get_db)
):
 """Update car data (admin only)"""
 if not current_user.is_admin:
 raise HTTPException(status_code=403, detail="Not authorized")

 car_repository = CarRepository(db)
 car = car_repository.update_car(car_id, car_data)
 if not car:
 raise HTTPException(status_code=404, detail="Car not found")
 return car

@router.delete("/{car_id}", status_code=status.HTTP_204_NO_CONTENT)
async def delete_car(
 car_id: int,
 current_user: User = Depends(get_current_user),
 db: Session = Depends(get_db)
):
 """Delete car (admin only)"""

```

```

 if not current_user.is_admin:
 raise HTTPException(status_code=403, detail="Not authorized")

 car_repository = CarRepository(db)
 success = car_repository.delete_car(car_id)
 if not success:
 raise HTTPException(status_code=404, detail="Car not found")
 ...

```

Лістинг Б.5 - Компонент пошуку автомобілів (React/TypeScript)

```

import React, { useState, useEffect } from 'react';
import { useDispatch } from 'react-redux';
import { format } from 'date-fns';
import styled from 'styled-components';
import DatePicker from 'react-datepicker';
import { toast } from 'react-toastify';
import 'react-datepicker/dist/react-datepicker.css';

import { setSearchParams } from '../redux/slices/searchSlice';
import { useNavigate } from 'react-router-dom';
import { api } from '../services/api';
import { Category } from '../types/category';

const SearchForm: React.FC = () => {
 const [startDate, setStartDate] = useState<Date | null>(null);
 const [endDate, setEndDate] = useState<Date | null>(null);
 const [pickupLocation, setPickupLocation] = useState<string>('');
 const [returnLocation, setReturnLocation] = useState<string>('');
 const [category, setCategory] = useState<string>('');
 const [categories, setCategories] = useState<Category[]>([]);

 const dispatch = useDispatch();
 const navigate = useNavigate();

 useEffect(() => {
 const fetchCategories = async () => {
 try {
 const response = await api.getCarCategories();

```

```

 setCategories(response.data);
 } catch (error) {
 console.error('Error fetching categories:', error);
 }
};

fetchCategories();
}, []);

const handleSubmit = (e: React.FormEvent) => {
 e.preventDefault();

 if (!startDate || !endDate || !pickupLocation) {
 toast.error('Будь ласка, заповніть всі обов\'язкові поля');
 return;
 }

 const searchParams = {
 startDate: format(startDate, 'yyyy-MM-dd'),
 endDate: format(endDate, 'yyyy-MM-dd'),
 pickupLocation,
 returnLocation: returnLocation || pickupLocation,
 category: category || undefined
 };

 dispatch(setSearchParams(searchParams));
 navigate('/search-results');
};

return (
 <StyledSearchForm onSubmit={handleSubmit}>
 <FormGroup>
 <Label>Дата початку*</Label>
 <DatePickerWrapper>
 <StyledDatePicker
 selected={startDate}
 onChange={(date: Date) => setStartDate(date)}
 selectsStart
 startDate={startDate}

```

```

 endDate={endDate}
 minDate={new Date()}
 placeholderText="Оберіть дату початку"
 dateFormat="dd.MM.yyyy"
 required
 />
 </DatePickerWrapper>
 </FormGroup>

 <FormGroup>
 <Label>Дата закінчення*</Label>
 <DatePickerWrapper>
 <StyledDatePicker
 selected={endDate}
 onChange={(date: Date) => setEndDate(date)}
 selectsEnd
 startDate={startDate}
 endDate={endDate}
 minDate={startDate || new Date()}
 placeholderText="Оберіть дату закінчення"
 dateFormat="dd.MM.yyyy"
 required
 />
 </DatePickerWrapper>
 </FormGroup>

 <FormGroup>
 <Label>Місце отримання*</Label>
 <StyledInput
 type="text"
 value={pickupLocation}
 onChange={(e) => setPickupLocation(e.target.value)}
 placeholder="Введіть місце отримання"
 required
 />
 </FormGroup>

 <FormGroup>
 <Label>Місце повернення</Label>

```

```

 <StyledInput
 type="text"
 value={returnLocation}
 onChange={(e) => setReturnLocation(e.target.value)}
 placeholder="Те саме що й місце отримання"
 />
 </FormGroup>

 <FormGroup>
 <Label>Категорія автомобіля</Label>
 <StyledSelect
 value={category}
 onChange={(e) => setCategory(e.target.value)}
 >
 <option value="">Всі категорії</option>
 {categories.map((cat) => (
 <option key={cat.id} value={cat.id}>
 {cat.name}
 </option>
))}
 </StyledSelect>
 </FormGroup>

 <SubmitButton type="submit">Знайти автомобіль</SubmitButton>
</StyledSearchForm>
);
};

// Styled Components
const StyledSearchForm = styled.form`
 display: flex;
 flex-wrap: wrap;
 gap: 1rem;
 padding: 1.5rem;
 background-color: #ECF0F1;
 border-radius: 8px;
 box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);

 @media (max-width: 768px) {

```

```

 flex-direction: column;
 }
`;

const FormGroup = styled.div`
 display: flex;
 flex-direction: column;
 flex: 1;
 min-width: 200px;
`;

const Label = styled.label`
 font-size: 14px;
 font-weight: 500;
 margin-bottom: 0.5rem;
 color: #2C3E50;
`;

const StyledInput = styled.input`
 height: 48px;
 padding: 0 1rem;
 border: 1px solid #ddd;
 border-radius: 4px;
 font-size: 16px;

 &:focus {
 outline: none;
 border-color: #E67E22;
 box-shadow: 0 0 0 2px rgba(230, 126, 34, 0.2);
 }
`;

const StyledSelect = styled.select`
 height: 48px;
 padding: 0 1rem;
 border: 1px solid #ddd;
 border-radius: 4px;
 font-size: 16px;
 background-color: white;

```

```

 &:focus {
 outline: none;
 border-color: #E67E22;
 box-shadow: 0 0 0 2px rgba(230, 126, 34, 0.2);
 }
 `;

```

```

const DatePickerWrapper = styled.div`
 width: 100%;
`;

```

```

const StyledDatePicker = styled(DatePicker)`
 height: 48px;
 width: 100%;
 padding: 0 1rem;
 border: 1px solid #ddd;
 border-radius: 4px;
 font-size: 16px;

 &:focus {
 outline: none;
 border-color: #E67E22;
 box-shadow: 0 0 0 2px rgba(230, 126, 34, 0.2);
 }
`;

```

```

const SubmitButton = styled.button`
 flex: 1;
 height: 48px;
 background-color: #E67E22;
 color: white;
 border: none;
 border-radius: 4px;
 font-size: 16px;
 font-weight: 600;
 cursor: pointer;
 transition: background-color 0.2s;
 margin-top: 1.5rem;

```

```

 &:hover {
 background-color: #D35400;
 }

 @media (max-width: 768px) {
 margin-top: 0.5rem;
 }
 `;

```

```

export default SearchForm;
```

```

Лістинг Б.6 - Сервіс для взаємодії з API (React/TypeScript)

```

import axios, { AxiosInstance, AxiosRequestConfig } from 'axios';

const API_URL = process.env.REACT_APP_API_URL ||
'http://localhost:8000/api';

class ApiService {
  private api: AxiosInstance;

  constructor() {
    this.api = axios.create({
      baseURL: API_URL,
      headers: {
        'Content-Type': 'application/json',
      },
    });
  }

  // Додаємо інтерцептор для автоматичного додавання токenu
  авторизації
  this.api.interceptors.request.use(
    (config) => {
      const token = localStorage.getItem('token');
      if (token && config.headers) {
        config.headers.Authorization = `Bearer ${token}`;
      }
    }
  );
}

```

```

        return config;
    },
    (error) => Promise.reject(error)
);

// Додаємо інтерцептор для обробки помилок
this.api.interceptors.response.use(
    (response) => response,
    (error) => {
        // Обробка закінчення терміну дії токена
        if (error.response && error.response.status === 401) {
            localStorage.removeItem('token');
            window.location.href = '/login';
        }
        return Promise.reject(error);
    }
);
}

// Методи для роботи з автомобілями
public getCarCategories() {
    return this.api.get('/car-categories');
}

public getAllCars(params?: any) {
    return this.api.get('/cars', { params });
}

public getAvailableCars(startDate: string, endDate: string,
category?: string) {
    return this.api.get('/cars/available', {
        params: { start_date: startDate, end_date: endDate, category_id:
category }
    });
}

public getCarById(id: number) {
    return this.api.get(`/cars/${id}`);
}

```

```

// Методи для роботи з бронюваннями
public createBooking(bookingData: any) {
    return this.api.post('/bookings', bookingData);
}

public getBookingById(id: number) {
    return this.api.get(`/bookings/${id}`);
}

public getUserBookings() {
    return this.api.get('/bookings/user');
}

public cancelBooking(id: number) {
    return this.api.patch(`/bookings/${id}/cancel`);
}

// Методи для роботи з платежами
public createPaymentIntent(paymentData: any) {
    return this.api.post('/payments/intent', paymentData);
}

// Методи для автентифікації
public login(credentials: { email: string; password: string }) {
    return this.api.post('/auth/login', credentials);
}

public register(userData: any) {
    return this.api.post('/auth/register', userData);
}

public logout() {
    localStorage.removeItem('token');
}

public getMe() {
    return this.api.get('/users/me');
}

```

```

}

export const api = new ApiService();
```

```

Лістинг Б.7 - Redux slice для управління пошуком (React/Redux)

```

import { createSlice, PayloadAction } from '@reduxjs/toolkit';
import { SearchParams, SearchState } from '../types/search';

const initialState: SearchState = {
 params: {
 startDate: '',
 endDate: '',
 pickupLocation: '',
 returnLocation: '',
 category: undefined
 },
 results: [],
 isLoading: false,
 error: null
};

const searchSlice = createSlice({
 name: 'search',
 initialState,
 reducers: {
 setSearchParams: (state, action: PayloadAction<SearchParams>) => {
 state.params = action.payload;
 },
 searchStart: (state) => {
 state.isLoading = true;
 state.error = null;
 },
 searchSuccess: (state, action: PayloadAction<any[]>) => {
 state.results = action.payload;
 state.isLoading = false;
 },
 searchFailure: (state, action: PayloadAction<string>) => {

```

```

 state.isLoading = false;
 state.error = action.payload;
 },
 clearSearchResults: (state) => {
 state.results = [];
 }
}
});

```

```

export const {
 setSearchParams,
 searchStart,
 searchSuccess,
 searchFailure,
 clearSearchResults
} = searchSlice.actions;

```

```

export default searchSlice.reducer;
...

```

Лістинг Б.8 - Компонент адміністративної панелі управління автомобілями (React/TypeScript)

```

import React, { useState, useEffect } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import {
 Box,
 Button,
 Paper,
 Table,
 TableBody,
 TableCell,
 TableContainer,
 TableHead,
 TableRow,
 Typography,
 IconButton,
 Tooltip,
 Dialog,

```

```

 DialogTitle,
 DialogContent,
 DialogActions
 } from '@material-ui/core';
import { Edit, Delete, Visibility, Add } from '@material-ui/icons';
import { RootState } from '../../redux/store';
import { fetchCars, deleteCar } from '../../redux/slices/adminCarsSlice';
import { formatCurrency } from '../../utils/formatters';
import { Car } from '../../types/car';
import CarEditForm from './CarEditForm';
import AlertMessage from '../../common/AlertMessage';

const CarsManagement: React.FC = () => {
 const dispatch = useDispatch();
 const { cars, loading, error } = useSelector((state: RootState) =>
state.adminCars);

 const [selectedCar, setSelectedCar] = useState<Car | null>(null);
 const [isEditDialogOpen, setIsEditDialogOpen] = useState(false);
 const [isDeleteDialogOpen, setIsDeleteDialogOpen] = useState(false);
 const [alertMessage, setAlertMessage] = useState('');
 const [alertType, setAlertType] = useState<'success' | 'error' |
''>('');

 useEffect(() => {
 dispatch(fetchCars());
 }, [dispatch]);

 const handleEditClick = (car: Car) => {
 setSelectedCar(car);
 setIsEditDialogOpen(true);
 };

 const handleDeleteClick = (car: Car) => {
 setSelectedCar(car);
 setIsDeleteDialogOpen(true);
 };

```

```

const handleDeleteConfirm = () => {
 if (selectedCar) {
 dispatch(deleteCar(selectedCar.id))
 .then(() => {
 setAlertMessage('Автомобіль успішно видалено');
 setAlertType('success');
 })
 .catch(() => {
 setAlertMessage('Помилка при видаленні автомобіля');
 setAlertType('error');
 });
 setIsDeleteDialogOpen(false);
 }
};

const handleAddNewCar = () => {
 setSelectedCar(null);
 setIsEditDialogOpen(true);
};

const handleEditDialogClose = () => {
 setIsEditDialogOpen(false);
};

const handleDeleteDialogClose = () => {
 setIsDeleteDialogOpen(false);
};

return (
 <Box>
 <Box display="flex" justifyContent="space-between"
alignItems="center" mb={3}>
 <Typography variant="h5" component="h1">
 Управління автопарком
 </Typography>
 <Button
 variant="contained"
 color="primary"
 startIcon={<Add />}

```

```

 onClick={handleAddNewCar}
 >
 Додати автомобіль
 </Button>
</Box>

{alertType && (
 <AlertMessage
 message={alertMessage}
 severity={alertType}
 onClose={() => setAlertType('')}
 />
)}

{loading ? (
 <Typography>Завантаження...</Typography>
) : error ? (
 <Typography color="error">Помилка: {error}</Typography>
) : (
 <TableContainer component={Paper}>
 <Table>
 <TableHead>
 <TableRow>
 <TableCell>ID</TableCell>
 <TableCell>Фото</TableCell>
 <TableCell>Марка</TableCell>
 <TableCell>Модель</TableCell>
 <TableCell>Реєстраційний номер</TableCell>
 <TableCell>Рік</TableCell>
 <TableCell>Паливо</TableCell>
 <TableCell>Трансмисія</TableCell>
 <TableCell>Місць</TableCell>
 <TableCell>Статус</TableCell>
 <TableCell>Тариф за добу</TableCell>
 <TableCell>Дії</TableCell>
 </TableRow>
 </TableHead>
 <TableBody>
 {cars.map((car) => (

```

```

<TableRow key={car.id}>
 <TableCell>{car.id}</TableCell>
 <TableCell>
 <img
 src={car.image_url || '/placeholder-car.png'}
 alt={`${car.brand} ${car.model}`}
 style={{ width: 60, height: 40, objectFit:
'cover' }}
 />
 </TableCell>
 <TableCell>{car.brand}</TableCell>
 <TableCell>{car.model}</TableCell>
 <TableCell>{car.registration_number}</TableCell>
 <TableCell>{car.year}</TableCell>
 <TableCell>{car.fuel_type}</TableCell>
 <TableCell>{car.transmission}</TableCell>
 <TableCell>{car.seats}</TableCell>
 <TableCell>{car.is_active ? 'Активний' :
'Неактивний'}</TableCell>

 <TableCell>{formatCurrency(car.daily_rate)}</TableCell>
 <TableCell>
 <Tooltip title="Переглянути">
 <IconButton size="small" color="primary"
onClick={() => {}}>
 <Visibility />
 </IconButton>
 </Tooltip>
 <Tooltip title="Редагувати">
 <IconButton size="small" color="primary"
onClick={() => handleEditClick(car)}>
 <Edit />
 </IconButton>
 </Tooltip>
 <Tooltip title="Видалити">
 <IconButton size="small" color="secondary"
onClick={() => handleDeleteClick(car)}>
 <Delete />
 </IconButton>

```

```

 </Tooltip>
 </TableCell>
</TableRow>
)}
</TableBody>
</Table>
</TableContainer>
)}

{/* Діалог для редагування/додавання автомобіля */}
<Dialog
 open={isEditDialogOpen}
 onClose={handleEditDialogClose}
 maxWidth="md"
 fullWidth
>
 <DialogTitle>
 {selectedCar ? 'Редагувати автомобіль' : 'Додати новий
автомобіль'}
 </DialogTitle>
 <DialogContent>
 <CarEditForm
 car={selectedCar}
 onSubmitSuccess={() => {
 setIsEditDialogOpen(false);
 setAlertMessage(
 selectedCar
 ? 'Автомобіль успішно оновлено'
 : 'Автомобіль успішно додано'
);
 setAlertType('success');
 dispatch(fetchCars());
 }}
 />
 </DialogContent>
</Dialog>

```

**ДОДАТОК В**  
**Слайди презентації**



Рисунок В.1 – Слайд 1

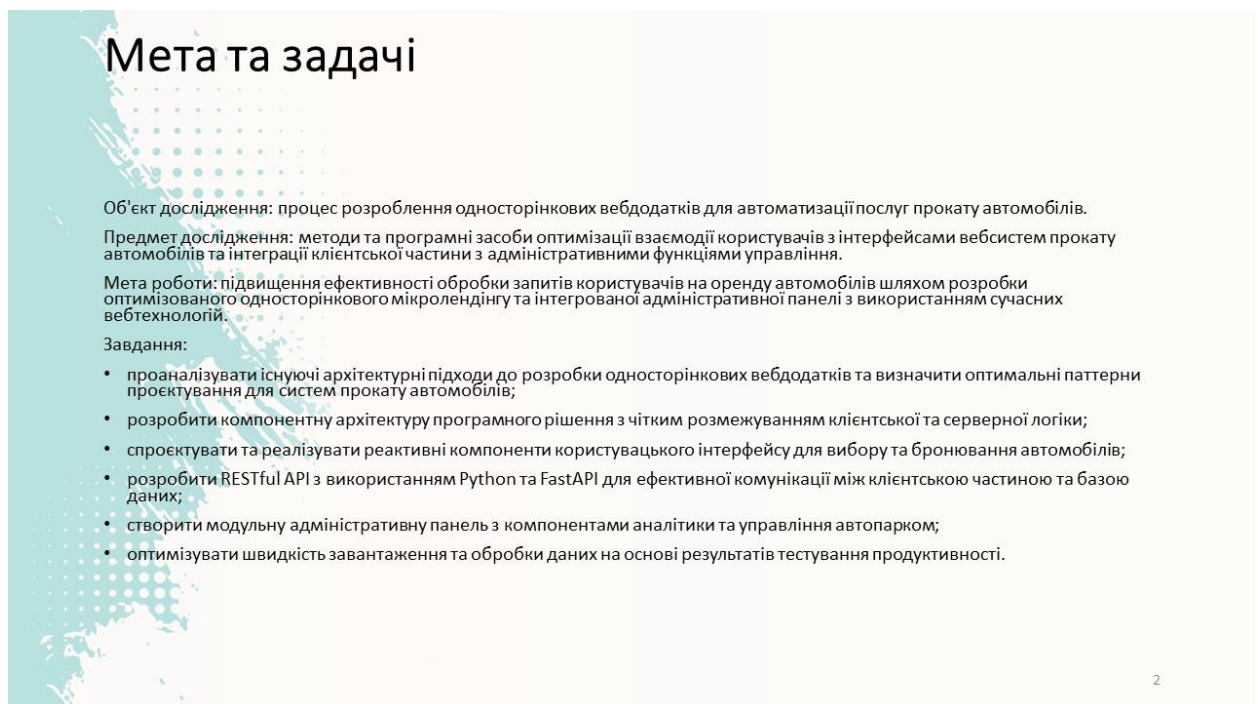
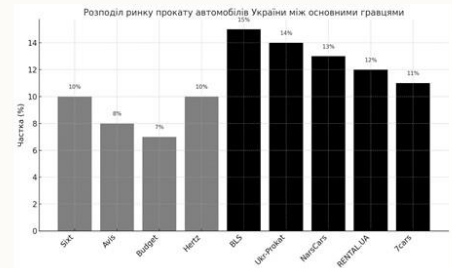


Рисунок В.2 – Слайд 2

## Порівняльний аналіз функціональних можливостей веб-сервісів прокату автомобілів

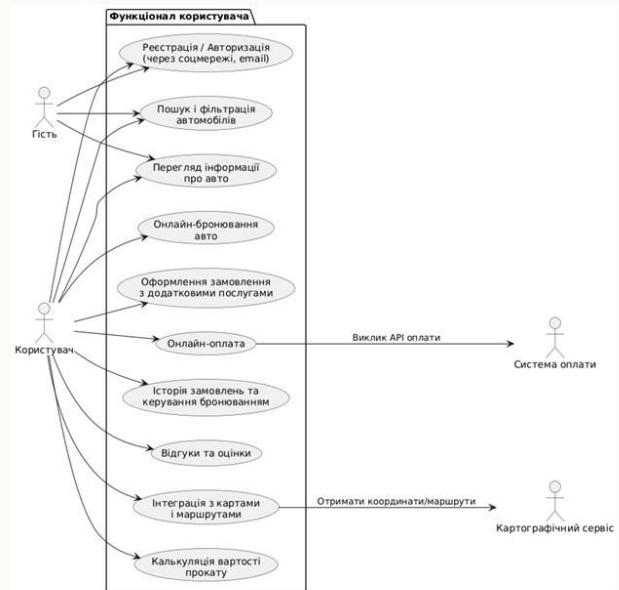
Функціональна можливість	Hertz	Enterprise	Europcar	Turo	Getaround
Онлайн-бронювання	Так	Так	Так	Так	Так
Мобільний додаток	Так	Так	Так	Так	Так
Система рейтингів	Ні	Ні	Ні	Так	Так
Розширений пошук	Так	Так	Так	Так	Так
P2P модель прокату	Ні	Ні	Ні	Так	Так
Інтеграція з GPS	Так	Так	Так	Ні	Так
Безключовий доступ	Ні	Ні	Ні	Ні	Так
Функція порівняння авто	Так	Ні	Так	Так	Ні
Програма лояльності	Так	Так	Так	Ні	Ні
API для партнерів	Так	Так	Так	Так	Так



3

Рисунок В.3 – Слайд 3

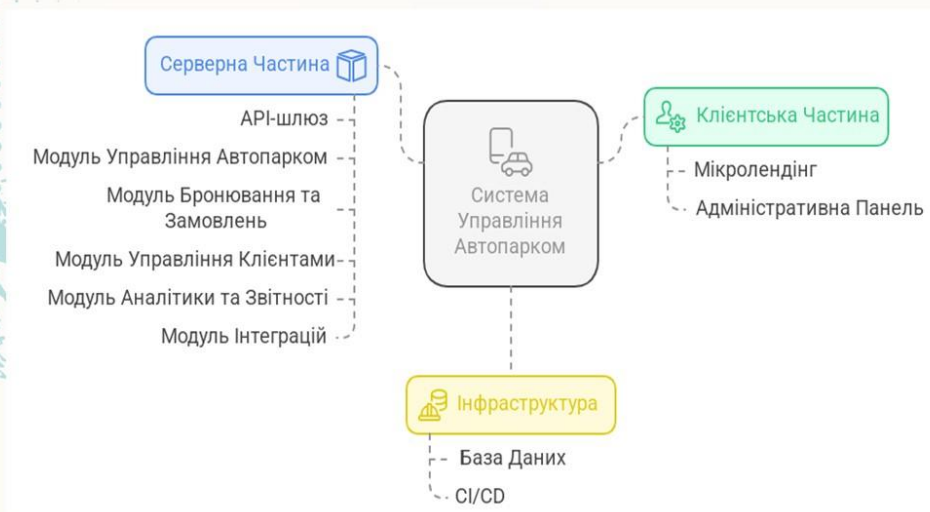
## Взаємодія в системі



4

Рисунок В.4 – Слайд 4

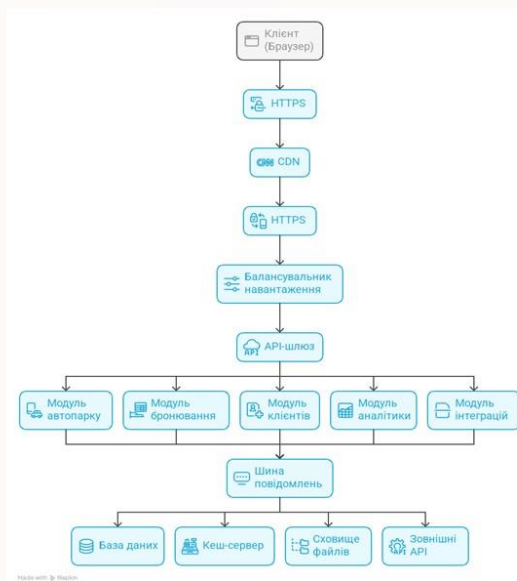
## Компонентна структура



5

Рисунок В.5 – Слайд 5

## Архітектура програмного забезпечення



6

Рисунок В.6 – Слайд 6

## Вибір мов програмування

Критерій	React	Angular	Vue.js
Розмір бібліотек (gziped)	~40KB	~130KB	~30KB
Продуктивність	Висока	Середня	Висока
Крива навчання	Помірна	Висока	Низька
Статична типізація	Опціонально (з TypeScript)	Вбудована (TypeScript)	Опціонально (з TypeScript)
Екосистема	Дуже розвинена	Розвинена	Зростаюча
Підтримка компаній	Facebook	Google	Спільнота
Популярність (GitHub stars)	201K	87K	205K
Швидкість розробки	Висока	Середня	Висока
SEO-оптимізація	Потребує додаткових рішень	Потребує додаткових рішень	Потребує додаткових рішень

Критерій	Node.js + Express.js	Python + FastAPI	Java + Spring Boot
Продуктивність	Висока для ІО-операцій	Висока з асинхронністю	Висока для обчислень
Масштабованість	Добра	Добра	Відмінна
Статична типізація	Опціонально (з TypeScript)	Вбудована (з анотаціями)	Вбудована (з анотаціями)
Екосистема	Дуже розвинена	Розвинена	Дуже розвинена
Крива навчання	Низька	Низька	Висока
Автоматична документація	Потребує додаткових рішень	Вбудована (Swagger/OpenAPI)	Потребує додаткових рішень
Асинхронність	Вбудована	Вбудована	Потребує додаткових рішень
Зрілість технології	Зріла	Нова, але зростаюча	Дуже зріла
Швидкість розробки	Висока	Дуже висока	Середня

7

Рисунок В.7 – Слайд 7

## Порівняння систем управління базами даних

Критерій	PostgreSQL	MySQL	MongoDB
Тип СУБД	Об'єктно-реляційна	Реляційна	Документно-орієнтована (NoSQL)
Підтримка транзакцій	Повна (ACID)	Залежить від движка	Обмежена
Масштабованість	Висока	Середня	Дуже висока
Підтримка JSON	Розширена	Базова	Вбудована
Цілісність даних	Сильна	Сильна	Обмежена
Гнучкість схеми	Обмежена	Обмежена	Висока
Вимоги до ресурсів	Високі	Середні	Середні
Швидкість читання	Висока	Дуже висока	Висока
Швидкість запису	Висока	Середня	Дуже висока
Підтримка спільноту	Активна	Активна	Активна

8

Рисунок В.8 – Слайд 8

```

 erDiagram
 bookings ||--o{ cars : "car_id"
 bookings ||--o{ customers : "customer_id"
 bookings ||--o{ booking_services : "service_id"
 bookings ||--o{ payments : "payment_id"
 bookings ||--o{ reviews : "review_id"
 cars ||--o{ car_images : "car_id"
 cars ||--o{ additional_services : "car_id"
 cars ||--o{ car_features : "car_id"
 cars ||--o{ promotion_categories : "category_id"
 cars ||--o{ promotions : "promotion_id"
 cars ||--o{ car_categories : "category_id"
 customers ||--o{ promotion_categories : "category_id"
 customers ||--o{ promotions : "promotion_id"
 employees ||--o{ employee_roles : "employee_id"
 employees ||--o{ role_permissions : "employee_id"
 roles ||--o{ role_permissions : "role_id"
 roles ||--o{ permissions : "permission_id"

```

The diagram illustrates a database schema for a car rental system. It includes tables for bookings, cars, customers, payments, reviews, booking services, car images, additional services, car features, promotion categories, promotions, car categories, employees, employee roles, role permissions, and permissions. Each table is represented by a box containing its attributes and data types. Lines connect tables to show foreign key relationships, with labels indicating the specific fields involved in the relationships.

Рисунок В.9 – Слайд 9

- 

**Додаток**  
Головний компонент, який відображає все інше.
- 

**Шапка**  
Містить логотип, навігацію та контактну інформацію.
- 

**Герой**  
Включає заголовок, підзаголовок, заклик до дії та форму.
- 

**Переваги**  
Відображає картини, що описують переваги.
- 

**Каталог**  
Показує фільтри, сортування, картини автомобілів та пагінацію.
- 

**Деталі автомобіля**  
Відображає зображення, характеристики, дію та форму бронювання.
- 

**Відгуки**  
Містить карусель відгуків клієнтів.
- 

**Питання-Відповіді**  
Анecdotes з частото задаваними питаннями та відповідями.
- 

**Підвал**  
Включає контактну інформацію, соціальні мережі, юридичні документи та карту сайту.

Бічне меню

Включає дашборд, автопарк, замовлення, клієнтів, тарифи, аналітику, контент та налаштування.

Основна область контенту

Відображає заголовки сторінки, фільтри, контент, таблиці, форми та графіки.

## Підвал

Показує версію системи, копірайт та посилання на документацію

Рисунок В.10 – Слайд 10

## Інтерфейс користувача

The screenshot displays a car rental website interface. On the left, there's a search bar with fields for location, dates, and car type. Below it, a grid shows four car listings: BMW X1, Audi A7, Toyota Camry, and Mercedes C-Class. Each listing includes a photo, model name, and price. To the right, a sidebar titled 'ФІЛЬТРИ' (Filters) allows users to select car brand, type, transmission, and rental period. The main content area features a 'СПЕЦІАЛЬНІ ПРОПОЗИЦІЇ' (Special Offers) section with two highlighted deals, followed by a 'ВІДГУКИ НАШИХ КЛІЄНТІВ' (Client Reviews) section with four testimonials. At the bottom, there's a 'ЧОМУ ОБИРАЮТЬ САМЕ НАС' (Why choose us) section with three icons representing fast service, 24/7 support, and competitive prices.

11

Рисунок В.11 – Слайд 11

## Інтерфейс користувача

The screenshot displays a detailed car rental page. On the left, a 'ТЕХНІЧНІ ХАРАКТЕРИСТИКИ' (Technical Characteristics) table lists details for a 2022 car, including engine type, transmission, number of doors, engine power, fuel consumption, and color. Below this, the 'УМОВИ ОРЕНДИ' (Rental Conditions) section outlines terms like vehicle age, mileage, and insurance. The 'ВІДГУКИ КЛІЄНТІВ' (Client Reviews) section shows a 5-star rating from a user named Olexandr M. On the right, a 'КАЛЬКУЛЯТОР ВАРТІСТІ' (Value Calculator) allows users to input dates, location, and car type to calculate the total rental cost, which is shown as 6450 UAH. Below the calculator, the 'ШВИДКИЙ ЗАПИТ' (Quick Request) form includes fields for name, phone number, and email, with a 'НАДІСЛАТИ ЗАПИТ' (Send Request) button.

12

Рисунок В.12 – Слайд 12



## Оптимізація

Метрика	До оптимізації	Після оптимізації	Покращення (%)
First Contentful Paint	2.7 с	1.2 с	55.6%
Largest Contentful Paint	4.1 с	2.1 с	48.8%
Cumulative Layout Shift	0.15	0.05	66.7%
Time to Interactive	5.8 с	3.1 с	46.6%
First Input Delay	120 мс	45 мс	62.5%
Total Blocking Time	450 мс	180 мс	60.0%

Ендпоінт API	Без кешування	З кешуванням	Покращення (%)
GET /api/cars	310 мс	85 мс	72.6%
GET /api/cars/available	420 мс	110 мс	73.8%
GET /api/categories	150 мс	35 мс	76.7%
GET /api/bookings/user/{user_id}	250 мс	95 мс	62.0%

15

Рисунок В.15 – Слайд 15

## Висновки

- Розроблено програмне забезпечення прокату автомобілів як односторінковий вебзастосунок з адміністративною панеллю.
  - Створено інтуїтивний користувацький інтерфейс мікролендінгу з адаптивним дизайном для різних пристроїв.
  - Розроблено RESTful API з використанням Python та FastAPI, що забезпечує високу продуктивність та надійну валідацію даних.
  - Впроваджено модульну адміністративну панель управління автопарком.
  - Оптимізовано продуктивність системи: час завантаження сторінок зменшено на 58%, час відгуку API — на 67%.
- Програмне забезпечення повністю відповідає поставленим вимогам та автоматизує всі ключові бізнес-процеси компанії з прокату автомобілів.

16

Рисунок В.16 – Слайд 16