МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА CRM ДЛЯ УПРАВЛІННЯ СТАНЦІЄЮ ТЕХНІЧНОГО ОБСЛУГОВУВАННЯ»**

Виконав: студент 4 курсу, групи 6.1212-пі-1
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
І.В. Красніков
(ініціали та прізвище)

Керівник завідувач кафедри програмної інженерії, доцент, к.ф.-м.н. Лісняк А.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Краснікову Івану Олексійовичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка CRM для управління станцією технічного обслуговування

керівник роботи Лісняк Андрій Олександрович, к.ф.-м.н., доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 02 » грудня 2025 року № 1741-с

2. Строк подання студентом роботи 01.06.2026 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Визначення вимог до системи.
2. Проєктування системи.
3. Програмна реалізація інформаційної системи.

4. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 09.12.2025 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	19.12.2025	
2.	Збір вихідних даних.	20.01.2026	
3.	Обробка методичних та теоретичних	10.02.2026	
	джерел.		
4.	Розробка першого та другого розділу.	01.04.2026	
5.	Розробка третього розділу.	15.05.2026	
6.	Оформлення та нормоконтроль	25.05.2026	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	15.06.2026	

Студент _____ І.О. Красніков
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.О. Лісняк
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка CRM для управління станцією технічного обслуговування»: 89 с., 35 рис., 1 табл., 25 джерел, 2 додатки.

АВТОМОБІЛЬ, БАЗА ДАНИХ, БРОНЮВАННЯ, ВЕБЗАСТОСУНОК, ІНТЕРФЕЙС КОРИСТУВАЧА, ІНФОРМАЦІЙНА СИСТЕМА, КЛІЄНТ, ОНЛАЙН-ЗАПИС, ОНЛАЙН-ОПЛАТА, СТАНЦІЯ ТЕХНІЧНОГО ОБСЛУГОВУВАННЯ, УПРАВЛІННЯ ЗАМОВЛЕННЯМИ, УПРАВЛІННЯ КЛІЄНТАМИ.

Об’єкт дослідження – процес автоматизації діяльності станції технічного обслуговування з використанням веборієнтованих інформаційних систем.

Мета роботи – розробити вебзастосунок CRM для управління процесами станції технічного обслуговування, який забезпечує реєстрацію клієнтів, онлайн-запис, формування замовлень, облік наданих послуг та можливість онлайн-оплати.

Методи дослідження – аналіз предметної області, проєктування бази даних, об’єктно-орієнтоване програмування, моделювання програмної архітектури, використання фреймворків Laravel і Vue.js.

У роботі розглянуто підходи до автоматизації роботи станцій технічного обслуговування та описано процес розробки CRM-системи для управління клієнтами, автомобілями, послугами та замовленнями. Створений вебзастосунок забезпечує можливість онлайн-запису клієнтів, призначення робіт майстрам, ведення історії обслуговувань та облік оплат. Реалізовано рольову систему доступу для адміністратора, менеджера, майстра і клієнта.

Розроблена система дозволяє підвищити ефективність роботи СТО, скоротити час обробки замовлень і покращити взаємодію з клієнтами.

SUMMARY

Bachelor's qualifying paper "Development of a CRM System for Managing a Vehicle Service Station": 89 pages, 35 figures, 1 table, 25 references, 2 supplements.

AUTOMOBILE, DATABASE, BOOKING, WEB APPLICATION, INFORMATION SYSTEM, USER INTERFACE, CLIENT, ONLINE BOOKING, ONLINE PAYMENT, SERVICE STATION, ORDER MANAGEMENT, CLIENT MANAGEMENT.

The object of the study is the process of automating the activities of a car service station using web-based information systems.

The aim of the study is to develop a CRM web application for managing the business processes of a car service station, providing client registration, online booking, order management, service accounting, and online payment functionality.

The methods of research are subject area analysis, database design, object-oriented programming, software architecture modeling, and the use of Laravel and Vue.js frameworks.

The paper describes approaches to automating car service management and presents the development of a CRM system for managing clients, vehicles, services, and orders. The created web application allows clients to book services online, managers to assign work to mechanics, and administrators to monitor payments and performance. A role-based access control system was implemented for different types of users.

The developed system improves the efficiency of the service station, reduces order processing time, and enhances client interaction quality.

ЗМІСТ

Завдання на кваліфікаційну роботу……..…..2
Реферат 4
Summary 5
Вступ..... 7
1 Визначення вимог до системи 9
1.1 Опис предметної області 9
1.2 Огляд програмного забезпечення..... 10
1.3 Аналіз вимог 12
2 Проєктування системи..... 15
2.1 Концептуальне проєктування 15
2.2 Етап логічного проєктування..... 24
2.3 Етап фізичного проєктування..... 28
3 Програмна реалізація системи..... 32
3.1 Архітектура та підхід до реалізації 32
3.2 Реалізація основних функціональних модулів..... 33
3.3 Реалізація інтерфейсу 41
Висновки 49
Перелік посилань..... 51
Додаток А Програмна реалізація Front-End частини 53
Додаток Б Програмна реалізація Back-End частини 83

ВСТУП

У сучасному світі цифрові технології активно проникають у всі сфери діяльності, зокрема й у автомобільний сервіс. Автомобіль є невід'ємною частиною життя людини, тому попит на швидке й якісне технічне обслуговування постійно зростає. Для ефективної організації роботи станцій технічного обслуговування (СТО) актуальним є впровадження сучасних CRM-систем (Customer Relationship Management), які автоматизують основні бізнес-процеси.

CRM для СТО дозволяє об'єднати управління клієнтами, замовленнями, послугами та оплатами в єдиній вебсистемі. Клієнт може записатися онлайн і оплатити послуги, а менеджери – контролювати замовлення та планувати роботу майстрів. Це підвищує ефективність обслуговування, покращує взаємодію з клієнтами та сприяє цифровій трансформації підприємства.

Актуальність теми визначається потребою в автоматизації процесів СТО та підвищенні якості сервісу шляхом використання веборієнтованих CRM-рішень.

Метою дипломної роботи є розробка вебзастосунку CRM для управління процесами станції технічного обслуговування, що забезпечує реєстрацію клієнтів, онлайн-запис, формування замовлень, облік послуг та можливість онлайн-оплати.

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

- дослідити предметну область діяльності станцій технічного обслуговування;
- проаналізувати існуючі рішення у сфері CRM-систем для СТО;
- визначити вимоги до функціоналу майбутньої системи;
- спроєктувати архітектуру вебзастосунку та структуру бази даних;
- реалізувати клієнтську та серверну частини системи з використанням фреймворків Laravel і Vue.js;

– забезпечити інтеграцію модуля онлайн-оплати;

– протестувати працездатність і надійність роботи системи.

Об’єкт дослідження – процес автоматизації діяльності станції технічного обслуговування автомобілів із використанням сучасних вебтехнологій.

Предмет дослідження – методи та засоби реалізації CRM-системи для управління клієнтами, послугами, замовленнями та оплатами.

Методи дослідження – аналіз предметної області, методи об’єктно-орієнтованого програмування, модульне проєктування архітектури системи, застосування сучасних фреймворків Laravel і Vue.js, а також тестування розробленого програмного забезпечення.

Структурно дипломна робота складається зі вступу, трьох розділів, висновків, переліку посилань і двох додатків. У першому розділі проведено аналіз предметної області та постановку завдання. У другому розділі описано процес проєктування архітектури системи, бази даних і користувацького інтерфейсу. У третьому розділі наведено опис реалізації вебзастосунку, тестування функціональності та оцінку ефективності розробленої системи. У додатках наведено фрагменти коду Front-End та Back-End частини.

1 ВИЗНАЧЕННЯ ВИМОГ ДО СИСТЕМИ

1.1 Опис предметної області

У межах даної дипломної роботи розробляється CRM-система для управління діяльністю станції технічного обслуговування автомобілів (СТО), що відповідає сучасним підходам до управління взаємовідносинами з клієнтами [25]. Система призначена для автоматизації основних бізнес-процесів, таких як ведення бази клієнтів, управління автомобілями, облік послуг, формування замовлень та контроль виконання робіт. Вона покликана замінити ручне ведення записів, використання таблиць або паперових журналів на сучасний вебзастосунок із зручним інтерфейсом і централізованим зберіганням даних.

У CRM-системі передбачено кілька типів користувачів: адміністратор, менеджер, майстер та клієнт.

Клієнт – фізична особа, яка звертається до СТО для отримання послуг з ремонту чи технічного обслуговування автомобіля. Клієнт має можливість зареєструватися в системі, додати свої транспортні засоби, переглядати доступні послуги та здійснювати онлайн-запис на зручну дату й час. Також користувач може переглядати історію замовлень, отримувати повідомлення про стан виконання робіт і оплачувати послуги онлайн.

Менеджер – співробітник СТО, який відповідає за прийом заявок від клієнтів, формування замовлень, призначення виконавців і контроль виконання робіт. Менеджер має доступ до панелі керування, де може переглядати всі записи клієнтів, змінювати деталі замовлення, переносити час обслуговування, змінювати статус замовлення (нове, у роботі, завершено, скасовано) та формувати звіти по замовленнях.

Майстер – технічний працівник, який виконує безпосередньо ремонтні або діагностичні роботи. У системі майстер може переглядати свої активні

замовлення, відзначати етапи виконання робіт і залишати коментарі щодо виконаних послуг чи використаних запчастин.

Адміністратор – користувач із найвищими правами доступу, який здійснює управління обліковими записами користувачів, налаштовує список послуг, контролює аналітику та переглядає статистику діяльності станції технічного обслуговування.

Основні задачі, які вирішує розроблювана CRM-система:

– автоматизація процесів управління клієнтами, замовленнями та послугами;
– спрощення взаємодії між клієнтами, менеджерами та майстрами;
– облік виконаних робіт і витрачених матеріалів;
– забезпечення можливості онлайн-запису та оплати послуг;
– формування аналітичних звітів для адміністрації СТО.

На ринку існують окремі рішення для управління клієнтськими записами або обліку послуг, проте більшість із них є дорогими, орієнтованими на великі дилерські центри або мають складний інтерфейс. Запропонована система націлена на невеликі та середні станції технічного обслуговування, для яких важливими є простота використання, доступність, швидке впровадження та гнучкість у налаштуванні.

Таким чином, розробка CRM-системи для СТО дозволить оптимізувати роботу персоналу, покращити обслуговування клієнтів і підвищити загальну ефективність управління підприємством за рахунок автоматизації ключових процесів, що підтверджується принципами автоматизації бізнес-процесів, описаними у [19, 20].

1.2 Огляд програмного забезпечення

Перед розробкою власної CRM-системи було проведено аналіз існуючих програмних рішень для управління станціями технічного обслуговування

автомобілів. Розглянуто найбільш поширені системи, що використовуються на ринку.

Bitrix24. Універсальна CRM-платформа з широким набором інструментів: управління клієнтами, завданнями, угодами та комунікаціями. Має модуль онлайн-запису та інтеграцію з платіжними системами.

Переваги: розвинена екосистема інтеграцій, наявність мобільного застосунку, хмарне розгортання.

Недоліки: відсутність спеціалізованого модуля для СТО, надмірна складність налаштування для невеликих сервісів, висока вартість корпоративних тарифів, інтерфейс перевантажений зайвим функціоналом.

АвтоДилер. Спеціалізоване програмне забезпечення для автосервісів і автосалонів, що охоплює управління замовленнями, складом запчастин та фінансовою звітністю.

Переваги: повна автоматизація процесів СТО, облік запчастин і матеріалів, формування нарядів.

Недоліки: десктопна архітектура без веборієнтованого доступу, відсутність клієнтського вебінтерфейсу для онлайн-запису, висока вартість ліцензії, складність впровадження.

1С: управління автосервісом. Галузеве рішення на платформі 1С для комплексної автоматизації автосервісу: від прийому автомобіля до видачі та розрахунку.

Переваги: глибока інтеграція з бухгалтерією, детальний облік робіт і матеріалів.

Недоліки: повна залежність від екосистеми 1С, відсутність сучасного вебінтерфейсу, значні витрати на впровадження та обслуговування, складність налаштування без спеціаліста.

У таблиці 1.1 наведено порівняння розглянутих систем за ключовими критеріями.

За результатами аналізу встановлено, що жодне з розглянутих рішень не забезпечує повного поєднання веборієнтованого доступу, онлайн-запису

клієнтів, обліку автомобілів із технічними характеристиками, кабінету майстра та вбудованої онлайн-оплати в єдиній доступній системі.

Таблиця 1.1 – Порівняння існуючих аналогів

Критерій	**Bitrix24**	**АвтоДилер**	**1С**	**Розроблювана система**
Онлайн-запис клієнта	Так	Ні	Ні	Так
Облік автомобілів (VIN)	Ні	Так	Так	Так
Веборієнтований доступ	Так	Ні	Ні	Так
Онлайн-оплата	Так	Ні	Ні	Так
Кабінет майстра	Ні	Так	Так	Так
Генерація PDF-рахунку	Обмежено	Так	Так	Так
Безкоштовне використання	Обмежено	Ні	Ні	Так
Адаптація під конкретний бізнес	Обмежено	Ні	Ні	Так

У зв’язку з недоліками кожного із існуючих аналогів прийнято рішення розробити власну CRM-систему з використанням стеку технологій Laravel + Vue.js + MySQL [1, 2, 5], який входить до числа найпопулярніших серед розробників [6], що дозволяє реалізувати весь необхідний функціонал із урахуванням специфіки роботи станції технічного обслуговування.

1.3 Аналіз вимог

На етапі аналізу вимог було визначено основні функціональні та нефункціональні вимоги до CRM-системи управління станцією технічного обслуговування автомобілів. Аналіз проводився з урахуванням потреб клієнтів, менеджерів і адміністратора системи, а також особливостей бізнес-процесів, характерних для сервісних центрів, з урахуванням загальноприйнятих підходів до розробки програмного забезпечення [19, 20].

Розглянемо **функціональні вимоги** до розроблюваної системи.

Управління користувачами (CRM-модуль):

– система повинна підтримувати реєстрацію, авторизацію та розмежування ролей (клієнт, менеджер, адміністратор);
– адміністратор має можливість додавати, редагувати та деактивувати користувачів.

Управління клієнтами:

– ведення бази клієнтів із контактними даними та історією замовлень;
– пошук і фільтрація клієнтів за ПІБ, телефоном або номером авто;
– перегляд статистики відвідувань і фінансової активності клієнта.

Управління автомобілями:

– можливість додавання інформації про автомобіль (марка, модель, рік випуску, тип палива, об’єм двигуна);
– зв’язок автомобіля з власником у базі клієнтів;
– перегляд історії технічного обслуговування кожного автомобіля.

Онлайн-запис / бронювання:

– клієнт має змогу створити запис на обслуговування, обравши авто, послуги, дату та час;
– система повинна автоматично блокувати зайняті часові слоти;
– можливість перенесення або скасування запису клієнтом;
– менеджер може редагувати записи, змінювати статуси (заплановано, в процесі, завершено, скасовано).

Управління послугами:

– адміністратор або менеджер може створювати, редагувати та видаляти послуги;
– кожна послуга має опис, базову ціну та категорію;
– підтримується динамічний розрахунок вартості залежно від параметрів автомобіля.

Модуль замовлень / нарядів:

– система формує замовлення на основі запису клієнта;

– передбачено зміну статусу замовлення, додавання коментарів та позначення виконаних робіт;
– автоматичне формування чеку із деталізацією вартості послуг.

Онлайн-оплата:

– клієнт має можливість оплатити послуги онлайн через інтегровану платіжну систему [7];
– інформація про оплату зберігається у базі даних.

Звіти та аналітика:

– система повинна надавати можливість перегляду статистики (кількість замовлень, популярність послуг, активність клієнтів, фінансові показники);
– можливість експорту звітів у форматах CSV або PDF.

Нефункціональні вимоги:

– *зручність використання*: інтерфейс має бути інтуїтивно зрозумілим як для клієнтів, так і для персоналу СТО;
– *швидкодія*: середній час відповіді сервера на запит не повинен перевищувати 2 секунд;
– *безпека*: дані користувачів, паролі та платіжна інформація мають бути захищені за допомогою шифрування й токен-автентифікації [22];
– *масштабованість*: система повинна дозволяти розширення функціональності (додавання нових модулів, інтеграція з API платіжних сервісів або систем лояльності), що відповідає рекомендаціям щодо проєктування REST API [24];
– *сумісність*: вебзастосунок повинен коректно працювати на сучасних браузерах і різних пристроях (ПК, планшет, смартфон).

2 ПРОЄКТУВАННЯ СИСТЕМИ

Розділ «Проєктування системи» описує концептуальну й логічну архітектуру CRM-застосунку для управління станцією технічного обслуговування. Мета розділу – деталізувати функціональні сценарії взаємодії користувачів із системою, визначити основні сутності й їхні зв’язки, а також представити діаграми, які служать основою для подальшої реалізації (ER-діаграма, use-case, компонувальна схема, діаграми послідовності).

2.1 Концептуальне проєктування

У межах даного проєкту розробляється веборієнтована CRM-система, призначена для автоматизації роботи станції технічного обслуговування автомобілів. Платформа забезпечує повний цикл взаємодії з клієнтами: від онлайн-запису на послуги до формування нарядів, управління автомобілями, перегляду історії обслуговування та проведення онлайн-оплат.

Система включає два основні типи користувачів: клієнт та менеджер СТО.

Після авторизації користувачі отримують доступ до відповідних функціональних модулів CRM:

- **клієнти** можуть створювати записи на обслуговування, управляти власними автомобілями, переглядати історію нарядів, отримувати рахунки та здійснювати онлайн-оплату;
- **менеджери** мають можливість керувати клієнтами, автомобілями, переліком послуг, формувати та редагувати наряди, адмініструвати записи, контролювати статуси виконання та переглядати оплату.

Платформа забезпечує зручну навігацію між основними модулями, що підвищує ефективність обслуговування та якість внутрішніх бізнес-процесів СТО.

На рисунках 2.1 – 2.4 наведено діаграми варіантів використання, яка демонструє ключові сценарії взаємодії користувачів із системою.

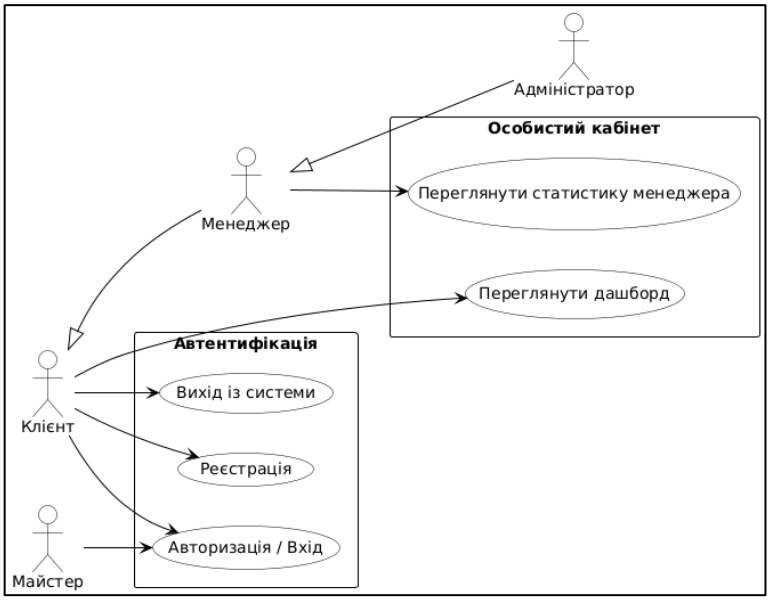


Рисунок 2.1 – Автентифікація та особистий кабінет

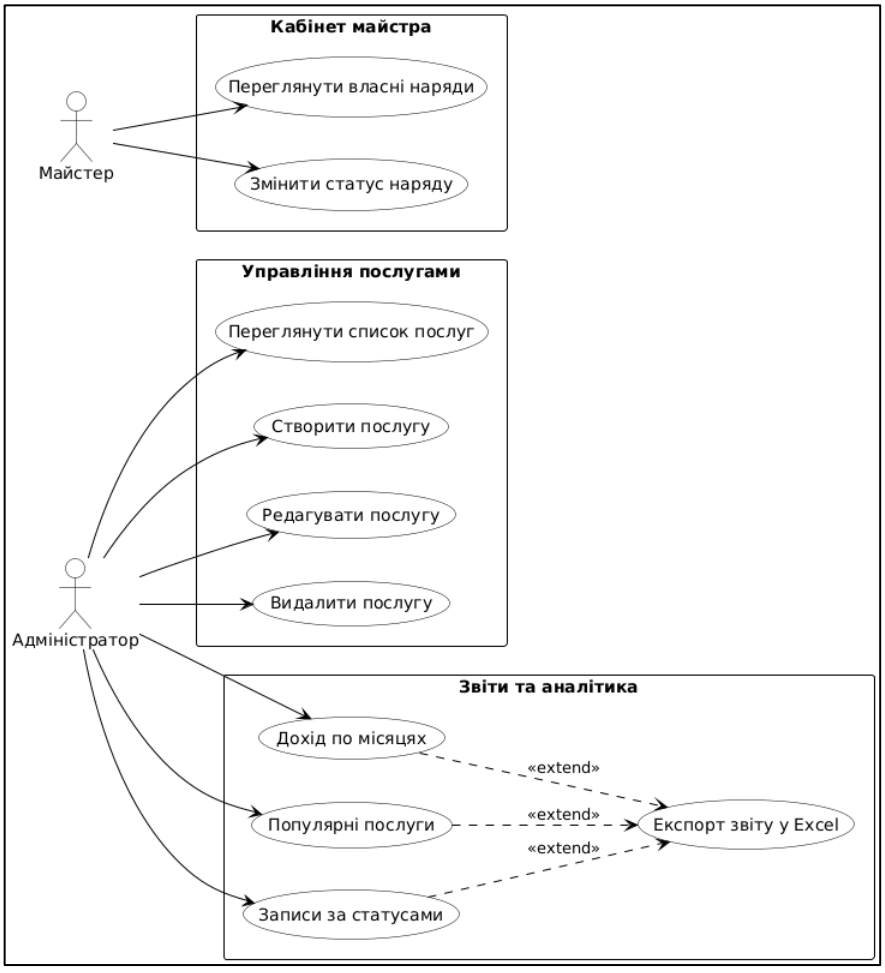


Рисунок 2.2 – Управління послугами, звіти та кабінет майстра

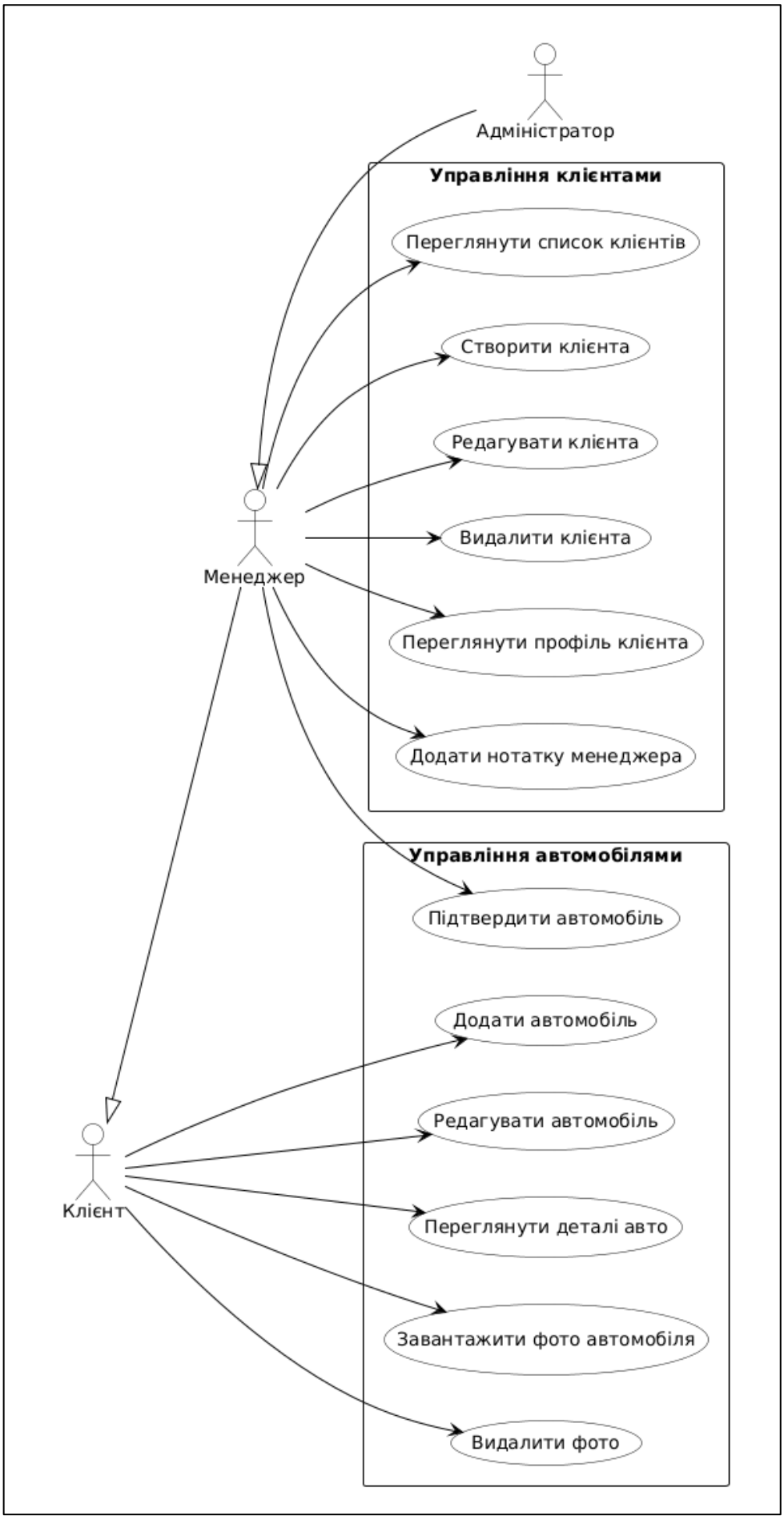


Рисунок 2.3 – Управління клієнтами та автомобілями



Рисунок 2.4 – Управління записами та оплата

Розглянемо акторів системи.

Клієнт може:

– зареєструватися та авторизуватися у системі;

– додати власні автомобілі з технічними характеристиками та фотографіями;

– створити запис на СТО, обравши послуги та дату обслуговування;
– переглянути динамічно сформований чек на основі обраних послуг та року автомобіля;
– сплатити онлайн за послуги через інтегровану платіжну систему;
– переглянути історію власних записів та їх поточні статуси;
– скасувати запис із зазначенням причини у коментарі;
– отримати рахунок у форматі PDF.

Менеджер може (додатково до клієнта):

– авторизуватися як співробітник СТО;
– управляти клієнтами: переглядати, створювати, редагувати, видаляти та додавати нотатки;
– управляти автомобілями клієнтів: переглядати, редагувати та підтверджувати;
– переглядати всі записи клієнтів у табличному вигляді;
– підтверджувати, редагувати та скасовувати записи;
– змінювати статуси запису: новий → підтверджено → в роботі → очікує оплати → завершено → скасовано;
– додавати коментарі до записів;
– підтверджувати факт оплати замовлення;
– переглядати майбутні записи та статистику.

Майстер може:

– авторизуватися як співробітник СТО;
– переглядати список записів, призначених особисто йому;
– переглядати деталі запису: автомобіль, перелік послуг, дату та час;
– змінювати статус запису: в роботі → очікує оплати;
– позначати запис як завершено після підтвердження оплати менеджером;
– додавати технічні коментарі до запису (примітки про виконані роботи);
– переглядати свій робочий розклад на день / тиждень.

Адміністратор може (додатково до менеджера):

– створювати, редагувати та видаляти послуги СТО;
– переглядати аналітичні звіти: дохід по місяцях, популярні послуги, записи за статусами;
– експортувати звіти у формат Excel.

Після виконання будь-якого варіанту використання система оновлює дані у базі та повідомляє користувача про результат.

Наведемо опис декількох з прецедентів системи. Опис прецедентів складено відповідно до загальноприйнятих підходів у розробці програмного забезпечення [19, 20].

Прецедент: «Запис клієнта на обслуговування».

Актор: клієнт.

Передумова: клієнт зареєстрований та увійшов у систему.

Основний потік:

а) клієнт переходить до сторінки створення запису;
б) обирає один зі своїх автомобілів або додає новий;
в) обирає необхідні послуги зі списку;
г) на основі типу автомобіля та обраних послуг система формує чек з детальною ціною;
д) клієнт обирає доступну дату та час обслуговування;
е) підтверджує запис та, за потреби, одразу здійснює онлайн-оплату.

Альтернативний потік: якщо обраний час зайнято, система пропонує доступні часові слоти.

Прецедент: «Формування наряду менеджером».

Актор: менеджер.

Передумова: менеджер авторизований у системі.

Основний потік:

а) менеджер відкриває список записів;
б) обирає запис клієнта та створює на його основі наряд;
в) додає або змінює перелік робіт і матеріалів;

г) система автоматично оновлює суму наряду;

д) менеджер встановлює статус в роботі → очікує оплати → завершено.

Альтернативний потік: якщо клієнт скасовує запис, наряд не формується або позначається як «скасовано».

Прецедент: «Онлайн-оплата послуг».

Актор: клієнт.

Передумова: сформовано рахунок або чек.

Основний потік:

а) клієнт відкриває свій запис чи наряд;

б) натискає «Сплатити онлайн»;

в) переходить на інтегровану платіжну платформу [7];

г) після успішної оплати система автоматично оновлює статус запису або наряду.

Альтернативний потік: у разі невдалої оплати система повідомляє користувача та дозволяє повторити транзакцію.

Прецедент: «Адміністрування записів».

Актор: менеджер.

Основний потік:

а) менеджер переглядає всі записи в єдиній панелі керування;

б) обирає потрібний запис;

в) може змінити дату й час, перелік послуг або відповідального майстра;

г) система перераховує чек при внесенні змін.

Прецедент: «Виконання роботи майстром».

Актор: майстер.

Передумова: менеджер підтвердив запис та призначив майстра.

Основний потік:

а) майстер відкриває список призначених записів;

б) обирає поточний запис та переглядає перелік послуг;

в) встановлює статус «В роботі»;

г) після завершення робіт встановлює статус «Очікує оплати» та додає коментар.

Альтернативний потік: якщо обсяг робіт змінився – майстер додає коментар, менеджер редагує перелік послуг і система перераховує чек.

Діаграми послідовності. На рисунках 2.5 – 2.8 наведено діаграми послідовності для ключових сценаріїв взаємодії користувачів із системою. Діаграми відображають порядок обміну повідомленнями між клієнтською частиною (Vue.js SPA), серверним API (Laravel) та базою даних [1, 2, 24].

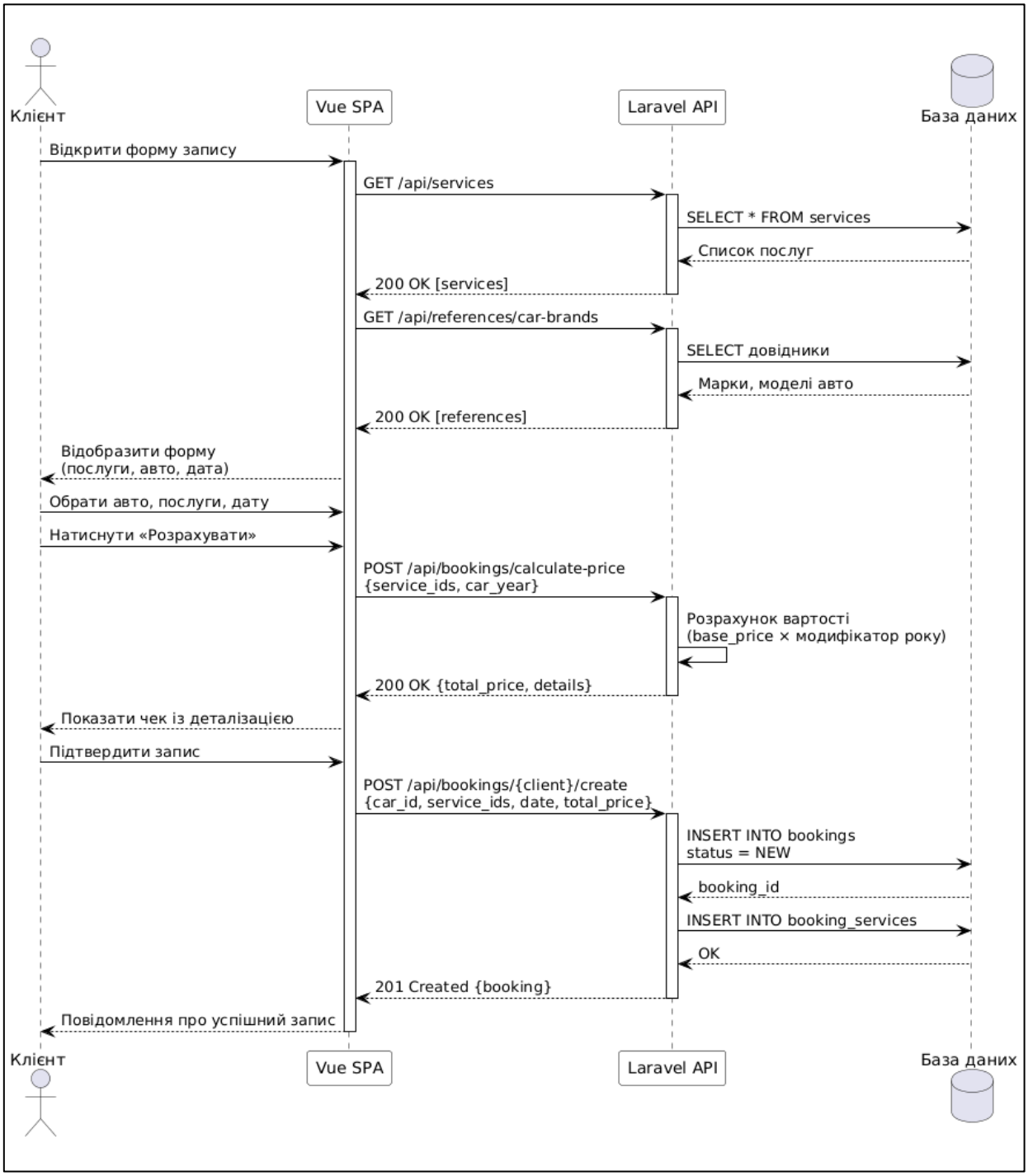


Рисунок 2.5 – Запис клієнта на обслуговування

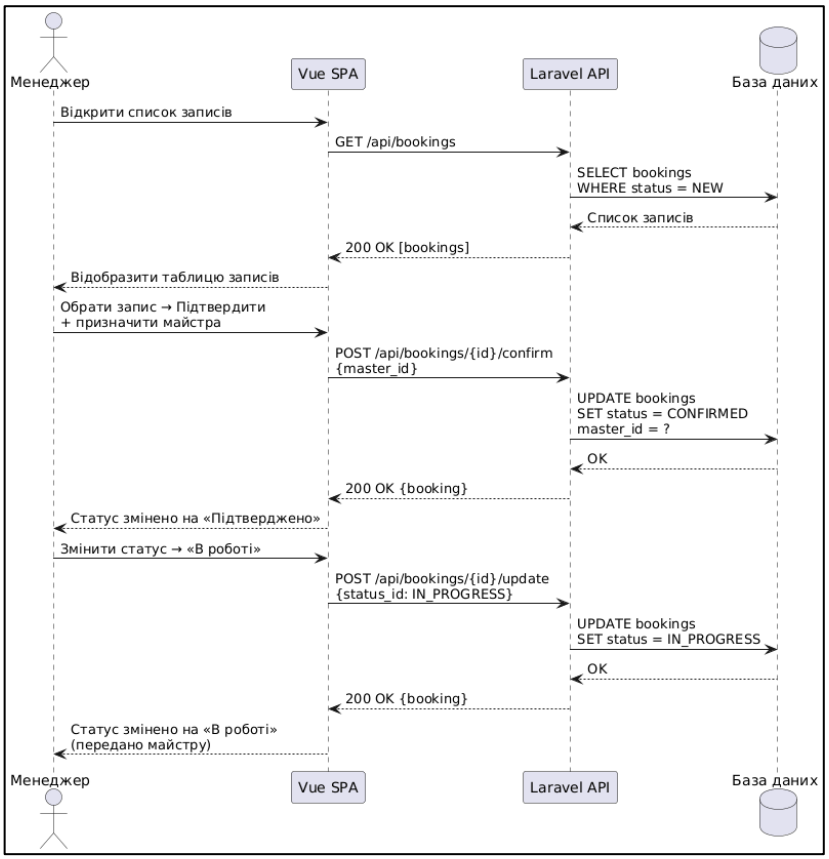


Рисунок 2.6 – Управління нарядом менеджером

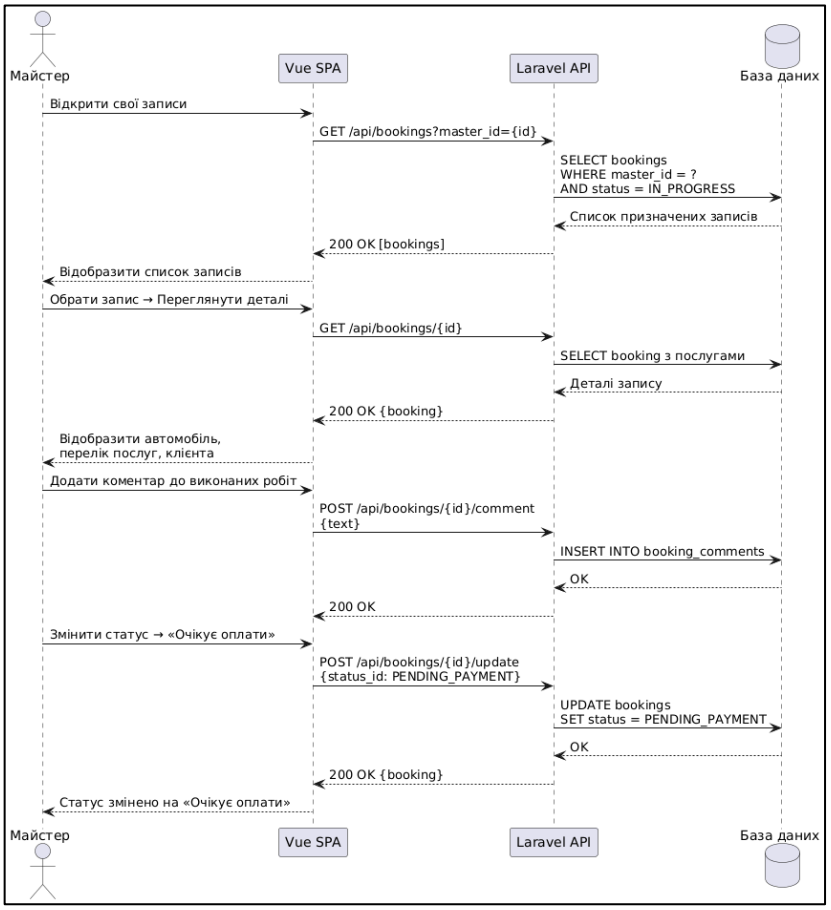


Рисунок 2.7 – Управління нарядом майстром

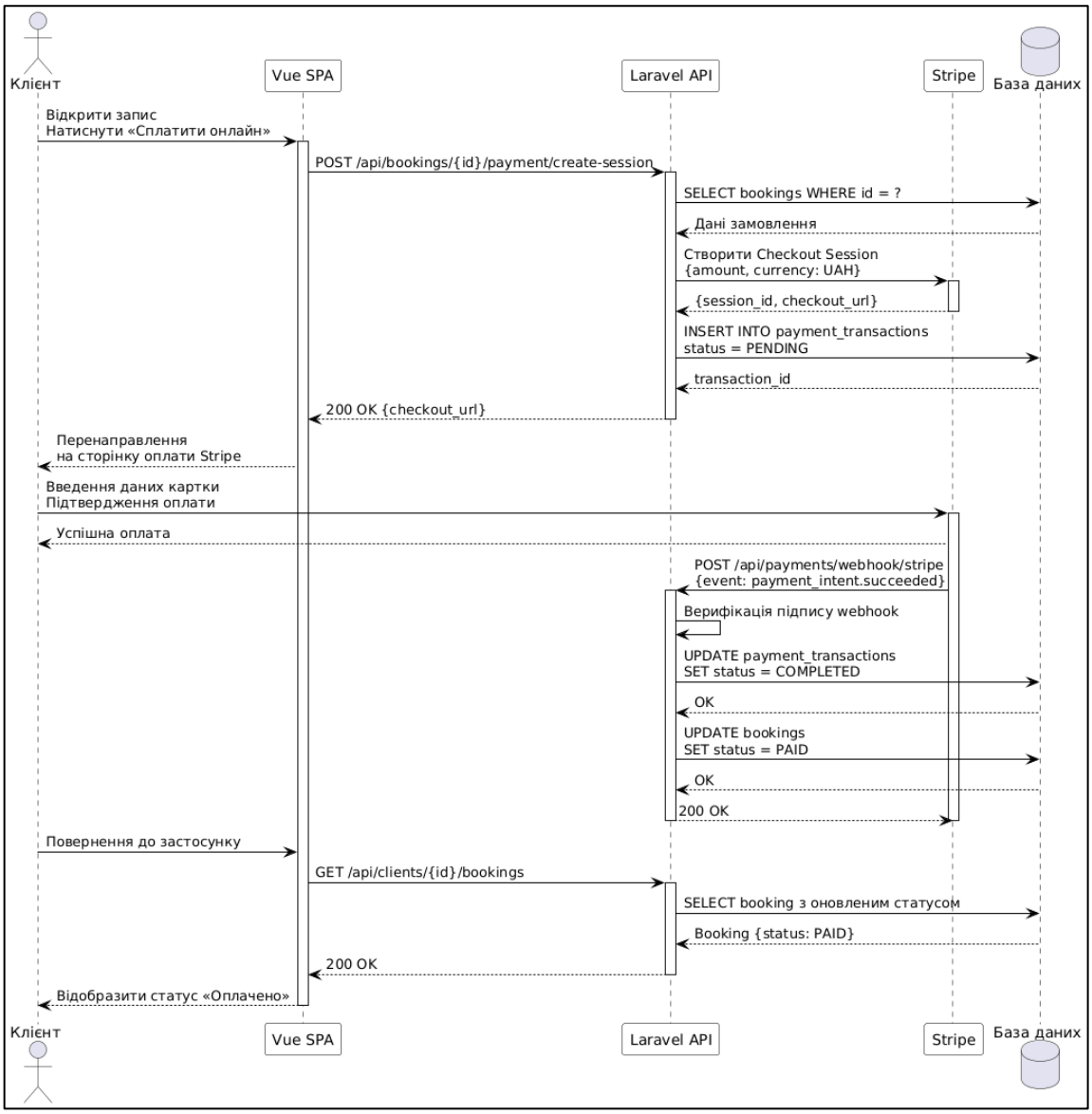


Рисунок 2.8 – Онлайн-оплата послуг

Наведені діаграми наочно показують, як саме відбувається обмін даними між користувачем та системою на кожному етапі роботи. Завдяки чіткому розмежуванню між інтерфейсом, сервером та базою даних кожна дія користувача призводить до передбачуваного та контрольованого результату.

2.2 Етап логічного проєктування

На етапі логічного проєктування було сформовано логічну модель даних для CRM-системи управління станцією технічного обслуговування

автомобілів. Концептуальну модель системи було деталізовано та перетворено у логічну структуру сутностей, яка відображає інформаційні об’єкти, їх атрибути та взаємозв’язки.

Основна увага приділялася сутностям, що забезпечують логіку запису клієнтів на СТО, управління транспортними засобами, обліку послуг та формування замовлень-нарядів. Логічна схема була розроблена з урахуванням вимог масштабованості, розширюваності та нормалізації даних (до третьої нормальної форми) [17, 5].

Статичне представлення структури даних відображено на ER-діаграмі (рис. 2.9). На ній наведено основні таблиці бази даних та їхні зв’язки.

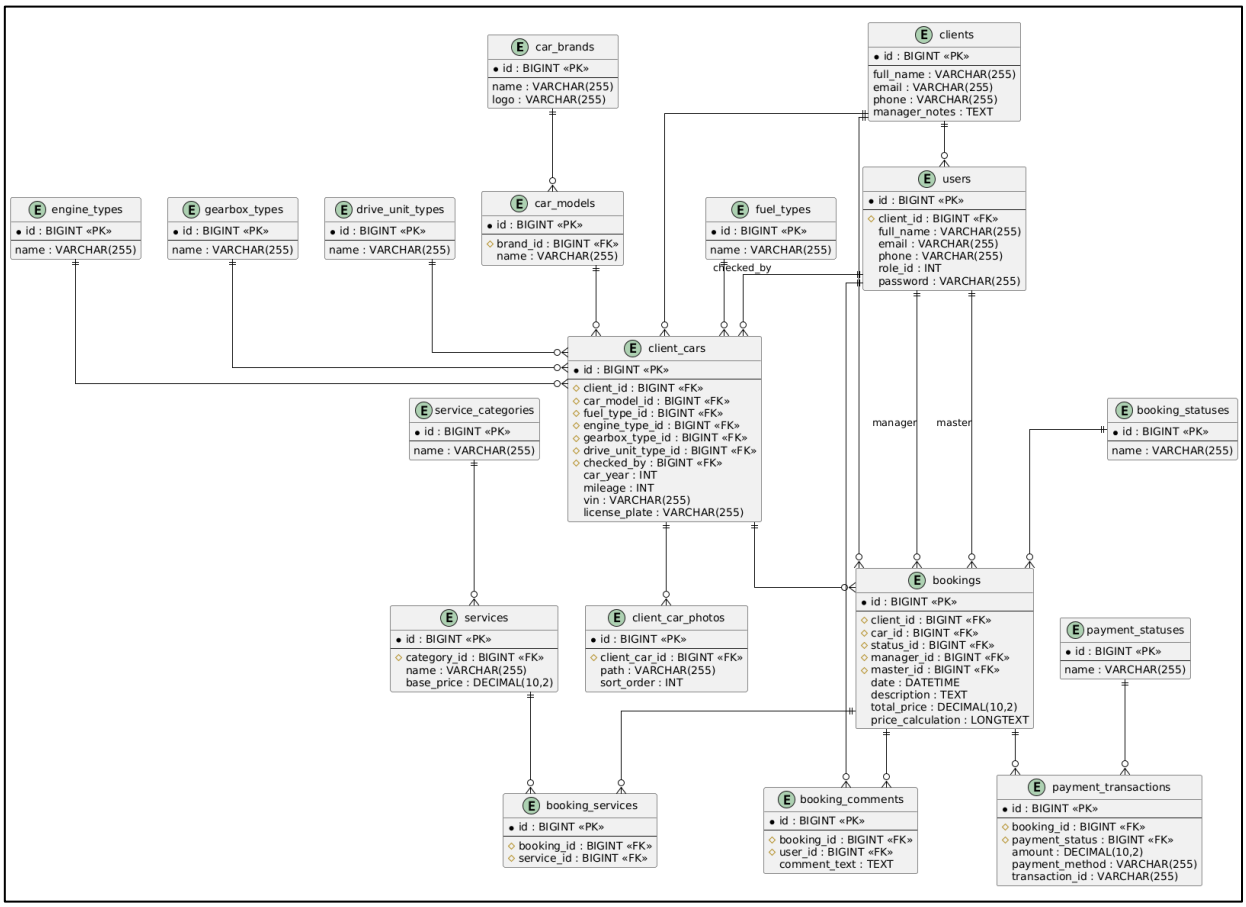


Рисунок 2.9 – ER-модель

Наведемо коротку характеристику логічних зв’язків між сутностями.

Clients – центральна сутність клієнта. Пов’язана з users (обліковий запис), client_cars (автомобілі) та bookings (записи на обслуговування).

Users – виконують дії залежно від ролі (клієнт, менеджер, адміністратор). Пов’язані з bookings двічі: як менеджер (manager_id) та як майстер (master_id). Також фіксують підтвердження автомобіля (checked_by у client_cars).

Bookings пов’язуються з:

– клієнтом (clients) та його автомобілем (client_cars);
– менеджером і майстром (users);
– статусом запису (booking_statuses);
– послугами (services) через зв’язок багато-до-багатьох (booking_services).

Booking comments – містить коментарі клієнта або менеджера (перенесення, скасування, уточнення).

Payment transactions – зберігає дані онлайн-оплати із посиланням на статус із довідника payment_statuses.

У моделі даних було виділено такі основні сутності:

– *users* – користувачі системи (full_name, email, phone, password, role_id, client_id);
– *clients* – незалежний від облікового запису профіль клієнта (full_name, email, phone, manager_notes);
– *car_brands*, *car_models* – довідники марок та моделей авто;
– *fuel_types*, *engine_types*, *gearbox_types*, *drive_unit_types* – довідники характеристик автомобіля;
– *client_cars* – автомобілі клієнтів із повними тех. характеристиками (client_id, car_model_id, car_year, mileage, vin, license_plate, checked_by);
– *client_car_photos* – фото автомобілів клієнта (client_car_id, path, sort_order);
– *services* – перелік послуг СТО (name, base_price, category_id);
– *bookings* – основна сутність запису на СТО (client_id, car_id, status_id, manager_id, master_id, date, description, total_price, price_calculation);

– *booking_services* – зв’язкова таблиця «запис – послуги» (booking_id, service_id);
– *booking_statuses* – довідник статусів запису (новий, підтверджено, в роботі, очікує оплати, завершено, скасовано, оплачено);
– *booking_comments* – коментарі до запису (booking_id, user_id, comment_text);
– *payment_transactions* – інформація про онлайн-оплати (booking_id, amount, payment_method, payment_status, transaction_id);
– *payment_statuses* – довідник статусів транзакції.

Структура взаємозв’язків відповідає принципам проєктування, описаним у [21, 23].

Діаграма класів. На діаграмі класів (рис. 2.10 – 2.11) показано логічні зв’язки між сутностями.

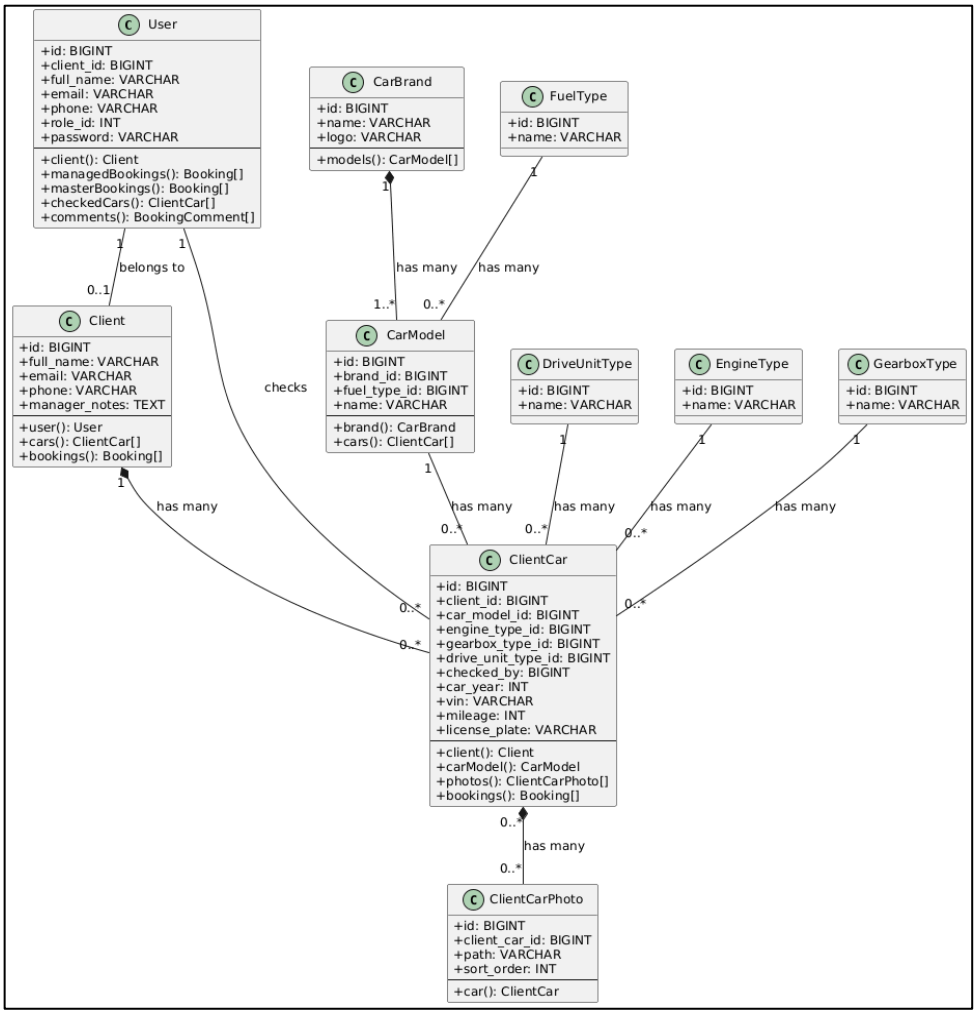


Рисунок 2.10 – Діаграма класів (частина 1)

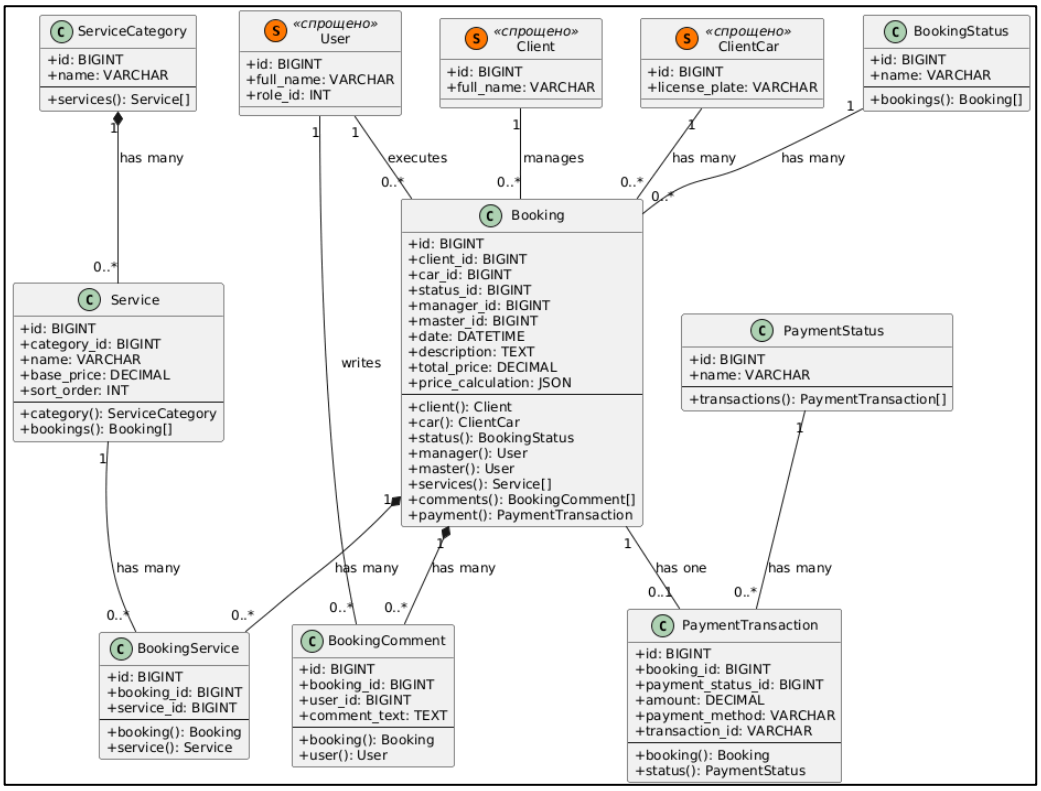


Рисунок 2.11 – Діаграма класів (частина 2)

Основні взаємодії:

– User пов’язаний з Client за принципом один-до-одного (необов’язково) та виступає менеджером або майстром у Booking;
– Client має багато Booking та багато ClientCar;
– Booking пов’язаний з: Client та ClientCar, BookingStatus, User, BookingService, BookingComment та PaymentTransaction;
– ClientCar належить до Client та CarModel, пов’язана з довідниками характеристик (FuelType, EngineType, GearboxType, DriveUnitType) та має колекцію фотографій ClientCarPhoto;
– CarModel належить до CarBrand;
– Service належить до ServiceCategory;
– PaymentTransaction пов’язана з довідником PaymentStatus;
– усі довідники мають зв’язки один-до-багатьох із основними сутностями.

Логічна модель даних охоплює всі ключові аспекти роботи станції технічного обслуговування – від реєстрації клієнтів та обліку автомобілів до

управління записами й обробки платежів. Структура сутностей приведена до третьої нормальної форми, що виключає надлишковість і спрощує подальшу підтримку системи. Виділення окремих довідників для статусів, характеристик автомобілів та категорій послуг забезпечує гнучкість моделі та можливість її розширення без змін в основній схемі.

2.3 Етап фізичного проєктування

На етапі фізичного проєктування вирішуються питання, пов’язані з оптимізацією роботи інформаційної системи, визначаються структури зберігання даних, реалізуються компоненти, що забезпечують взаємодію між клієнтською частиною, серверною логікою та базою даних. Особлива увага приділяється вибору архітектурних рішень, що дозволяють забезпечити масштабованість, продуктивність та надійність роботи вебплатформи.

Діаграма компонентів. На рисунку 2.12 представлена діаграма компонентів, яка демонструє структурні елементи системи та їх зв’язки.

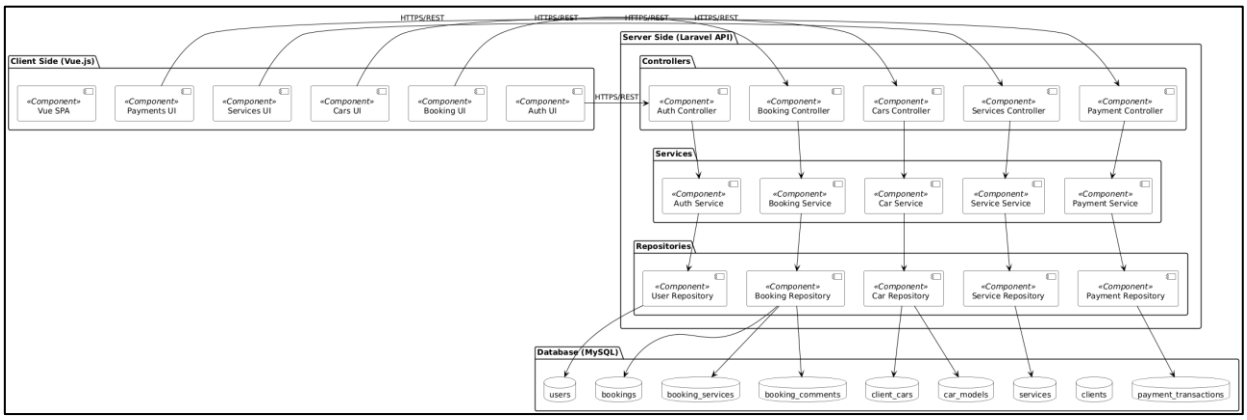


Рисунок 2.12 – Діаграма компонентів

Клієнтська частина, розроблена на Vue.js [2, 13], відповідає за відображення інтерфейсу, формування запитів та взаємодію з користувачем.

Серверна частина на Laravel містить контролери, сервіси та репозиторії [1, 12, 21], які реалізують бізнес-логіку та доступ до даних.

Усі дані зберігаються у реляційній базі даних MySQL, де налаштовані оптимізовані індекси та структури таблиць для швидкої обробки запитів від серверу [5, 18].

Діаграма розгортання. На рисунку 2.13 наведена діаграма розгортання, що демонструє фізичну інфраструктуру системи.

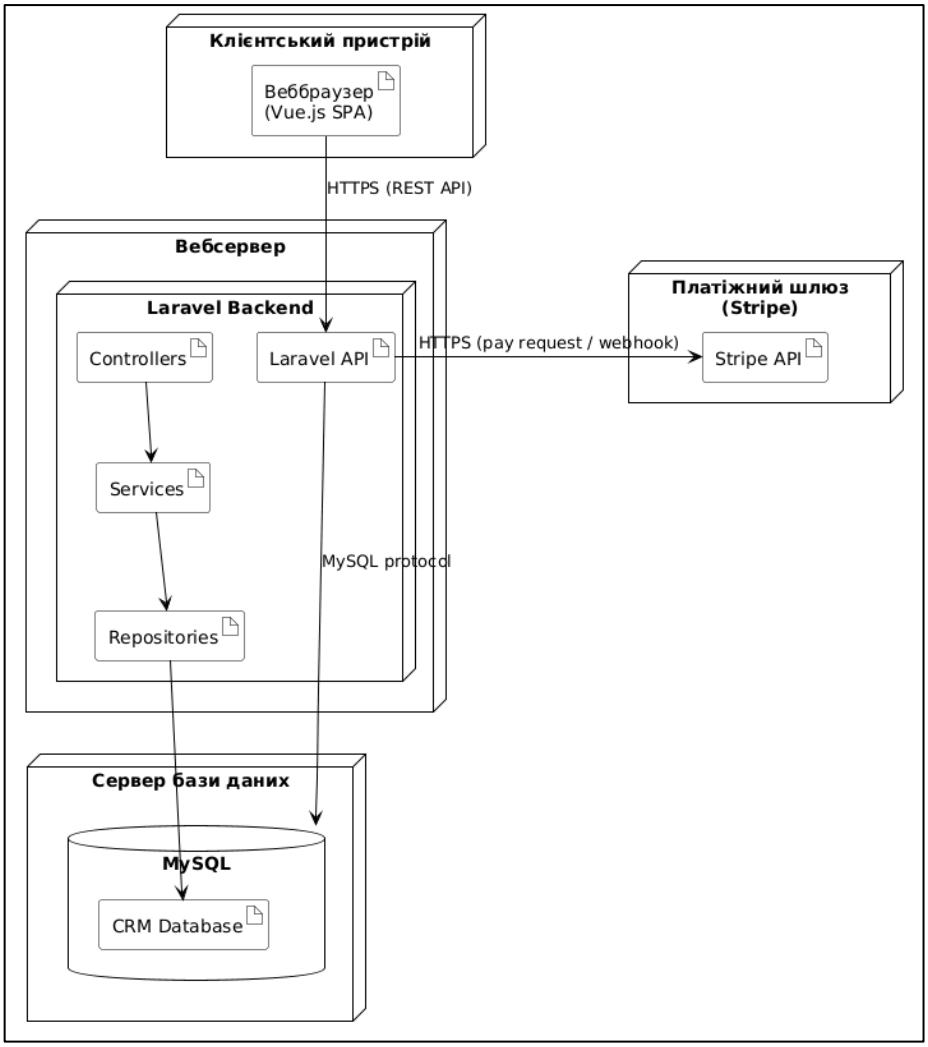


Рисунок 2.13 – Діаграма розгортання

Розгортання передбачає:

– *клієнтський пристрій* (комп’ютер або смартфон) з веббраузером, що завантажує SPA-додаток;
– *вебсервер,* на якому працює Laravel API, що виконує бізнес-логіку, валідацію даних, роботу зі статусами записів та взаємодію зі службами оплати;

– *сервер бази даних*, який зберігає інформацію про клієнтів, автомобілі, записи, послуги, платежі;
– *платіжний шлюз Stripe [7],* який використовується для обробки онлайн-оплат через захищений API.

Взаємодія компонентів здійснюється через захищені канали HTTPS та SSL/TLS.

Фізична архітектура системи побудована за принципом чіткого розподілу відповідальності між клієнтською частиною, серверним API та базою даних, що відповідає принципам чистої архітектури [22]. Використання окремого платіжного шлюзу Stripe знімає з системи відповідальність за обробку фінансових даних і підвищує рівень безпеки. Захищена взаємодія компонентів через HTTPS та SSL/TLS забезпечує надійність передачі даних у виробничому середовищі.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

Розділ «Програмна реалізація системи» описує практичне втілення спроєктованої CRM-системи для управління станцією технічного обслуговування автомобілів. Мета розділу – висвітлити архітектурні рішення та технологічний стек, що використовувались у процесі розробки, продемонструвати реалізацію ключових функціональних модулів системи з наведенням фрагментів програмного коду, а також представити реалізований інтерфейс користувача для всіх категорій користувачів.

3.1 Архітектура та підхід до реалізації

Система реалізована за клієнт-серверною архітектурою з розподілом відповідальності. Клієнтська частина є односторінковим застосунком (SPA), що взаємодіє з сервером виключно через REST API. Такий підхід забезпечує незалежність фронтенду від бекенду та спрощує масштабування.

Серверна частина побудована на фреймворку Laravel 12 за патерном MVC [1, 12, 16]. Усі запити від клієнта обробляються відповідними контролерами, які делегують бізнес-логіку сервісним класам та звертаються до бази даних через Eloquent ORM [1]. Маршрутизація API описана у файлі routes/api.php із групуванням за функціональними модулями та застосуванням middleware для перевірки автентифікації з урахуванням практик [4].

Клієнтська частина реалізована на Vue.js 3 з використанням Composition API [2, 13, 14] та централізованим керуванням станом через Vuex 4 [11]. Навігація між сторінками забезпечується Vue Router 4 [10] із захистом маршрутів на основі ролі авторизованого користувача. HTTP-запити до API виконуються через Axios [3, 15] із глобально налаштованим базовим URL та автоматичним підставленням токена у заголовок запиту.

База даних MySQL спроєктована з урахуванням третьої нормальної форми. Структура таблиць керується міграціями Laravel [5, 17], що забезпечує версіонування схеми та можливість відтворення бази на будь-якому середовищі командою php artisan migrate.

3.2 Реалізація основних функціональних модулів

У цьому підрозділі розглянуто ключові технічні рішення, застосовані при реалізації основних модулів системи.

Модуль автентифікації. Реєстрація нового користувача передбачає одночасне створення двох записів: у таблиці clients (профіль клієнта СТО) та у таблиці users (обліковий запис для входу). Якщо клієнт із таким номером телефону вже існує в базі (наприклад, був створений менеджером раніше), система автоматично пов’язує новий обліковий запис із існуючим профілем. Після реєстрації генерується Sanctum-токен, який повертається клієнту.

Фрагмент коду методу register() контролера AuthController зображено на рисунках 3.1 та 3.2.

```
    public function register(RegisterRequest $request): JsonResponse
    {
        // If a client with this phone was already created via the public booking form — link to them
        $client = Client::where('phone', $request->phone)->first();

        if ($client) {
            // Replace the system-generated placeholder email if present
            if ($client->email && str_contains($client->email, '@booking.local')) {
                $client->update([
                    'full_name' => $request->name,
                    'email' => $request->email,
                ]);
            }
        } else {
```

Рисунок 3.1

```
        $client = Client::create([
                'full_name' => $request->name,
                'email' => $request->email,
                'phone' => $request->phone,
            ]);
        }

        $user = User::create([
            'full_name' => $request->name,
            'email' => $request->email,
            'phone' => $request->phone,
            'password' => Hash::make($request->password),
            'role_id' => User::ROLE_CLIENT,
            'client_id' => $client->id,
        ]);

        $token = $user->createToken('auth-token')->plainTextToken;

        return response()->json([
            'message' => 'Користувача успішно зареєстровано',
            'user' => $user,
            'token' => $token,
        ], 201);
```

Рисунок 3.2

Модуль управління записами. Життєвий цикл замовлення реалізовано через послідовну зміну статусів: новий → підтверджено → в роботі → очікує оплати → завершено.

Кожна зміна статусу перевіряється на допустимість відповідно до ролі користувача: менеджер підтверджує та закриває замовлення, майстер переводить у статуси «в роботі» та «очікує оплати».

До кожного запису можна додавати коментарі – це дозволяє фіксувати причини змін, уточнення від клієнта або технічні примітки майстра. Окремо передбачено можливість скасування запису на будь-якому етапі з обов’язковим зазначенням причини у коментарі.

Реалізацію контролера наведено на рисунку 3.3, кабінету майстра – на рисунках 3.4 та 3.5.

```
public function createBooking(int $clientId, array $data): Booking
    {
        $newStatusId = BookingStatus::getNewStatusId();
        $priceDetails = $this->calculatePriceDetails($data['car_id'], $data['service_ids']);
        $totalPrice = $priceDetails['total_price'];

        $booking = Booking::create([
            'client_id' => $clientId,
            'car_id' => $data['car_id'],
            'date' => $data['date'],
            'description' => $data['comment'] ?? null,
            'status_id' => $newStatusId,
            'manager_id' => null,
            'total_price' => $totalPrice,
            'price_calculation' => $priceDetails,
        ]);

        if (!empty($data['service_ids'])) {
            $booking->services()->attach($data['service_ids']);
        }
        $booking->load(['car.carModel.brand', 'services', 'client', 'status']);
        return $booking;
    }

    public function calculateTotalPrice(int $carId, array $serviceIds): float
    {
        if (empty($serviceIds)) {
            return 0;
        }

        $services = Service::whereIn('id', $serviceIds)->get();
        $basePrice = $services->sum('base_price');
        $car = ClientCar::find($carId);
        if (!$car || !$car->car_year) {
            return $basePrice;
        }

        $currentYear = (int) date('Y');
        $carYear = (int) $car->car_year;
        $yearsOld = max(0, $currentYear - $carYear);
        $yearModifierPercent = min(50, $yearsOld * 5);
        $yearModifier = ($basePrice * $yearModifierPercent) / 100;
        return round($basePrice + $yearModifier, 2);
    }
```

Рисунок 3.3 – Контролер управління записами

```
        public function updateStatus(Request $request, Booking $booking): JsonResponse
        {
            if ($booking->master_id !== $request->user()->id) {
                return response()->json(['message' => 'Доступ заборонено'], 403);
            }

            $request->validate([
                'status_id' => ['required', 'integer', 'exists:booking_statuses,id'],
            ]);

            $allowedTransitions = [
                BookingStatus::getConfirmedStatusId()  =>
BookingStatus::getInProgressStatusId(),
                BookingStatus::getInProgressStatusId() =>
BookingStatus::getPendingPaymentStatusId(),
            ];

            $newStatusId = (int) $request->input('status_id');

            $currentStatusId = $booking->status_id;

            if (!isset($allowedTransitions[$currentStatusId]) ||
$allowedTransitions[$currentStatusId] !== $newStatusId) {
                return response()->json(['message' => 'Недозволений перехід статусу'], 422);
            }

            $updated = $this->bookingRepository->updateStatus($booking->id,
$newStatusId);

            // Notify the client when work is done and payment is expected
            if ($newStatusId === BookingStatus::getPendingPaymentStatusId()) {
                $updated->load(['client', 'car.carModel.brand', 'services', 'status']);
                $email = $updated->client?->email;

                if ($email && !str_contains($email, '@booking.local')) {
                    try {
                        Mail::to($email)->send(new BookingCompletedMail($updated));
                    } catch (\Throwable $e) {
                        \Log::warning('Failed to send booking receipt email: ' . $e-
>getMessage());
                    }
                }
            }
```

Рисунок 3.4 – Метод updateStatus() контролеу MasterController.php

```
    public function index(Request $request): JsonResponse
    {
        $masterId = $request->user()->id;
        $bookings = Booking::where('master_id', $masterId)
            ->when($request->input('date'), fn ($q, $date) =>
                $q->whereDate('date', $date)
            )
            ->when($request->input('status_id'), fn ($q, $statusId) =>
                $q->where('status_id', $statusId)
            )
            ->with('car.carModel.brand', 'services', 'status', 'client')
            ->whereNotIn('status_id', [BookingStatus::getNewStatusId()])
            ->orderBy('date')
            ->get();
        return response()->json($bookings);
    }

    public function show(Request $request, Booking $booking): JsonResponse
    {
        if ($booking->master_id !== $request->user()->id) {
            return response()->json(['message' => 'Доступ заборонено'], 403);
        }

        return response()->json(
            $booking->load('car.carModel.brand', 'services', 'status', 'client')
        );
    }
```

Рисунок 3.5 – Методи index() та show() контролеру MasterController.php

Модуль онлайн-оплати. Інтеграція зі Stripe реалізована через офіційний PHP SDK [7]. При ініціації оплати сервер створює Checkout Session із сумою замовлення у гривнях та URL для повернення клієнта. Після завершення оплати Stripe надсилає webhook-подію на ендпоїнт /api/payments/webhook/stripe. Сервер верифікує підпис події та оновлює статус транзакції і замовлення в базі даних.

Такий підхід гарантує надійність: статус оновлюється лише після підтвердження від платіжної системи, а не на основі редиректу від клієнта, що відповідає рекомендаціям щодо безпечної обробки платежів [7]. Контролер оплати зображений на рисунках 3.6 – 3.8.

```
class PaymentController extends Controller
{
    public function __construct()
    {
        Stripe::setApiKey(config('services.stripe.secret_key'));
    }

    public function createCheckoutSession(Booking $booking): JsonResponse
    {
        try {
            $paidTransaction = PaymentTransaction::where('booking_id', $booking->id)
                ->where('payment_status', PaymentTransaction::STATUS_COMPLETED)
                ->first();
            if ($paidTransaction) {
                return response()->json([
                    'message' => 'Цей букінг вже оплачений',
                ], 400);
            }

            $booking->load('client', 'car', 'services');
            $session = Session::create([
                'payment_method_types' => ['card'],
                'line_items' => [[
                    'price_data' => [
                        'product_data' => [
                            'name' => 'Запис #' . $booking->id,
                            'description' => 'Оплата за послуги: ' . $booking->services-
>pluck('name')->join(', '),
                        ],
                        'unit_amount' => (int)($booking->total_price * 100),
                    ],
                    'quantity' => 1,
                ]],
                'mode' => 'payment',
                'success_url' => config('app.frontend_url') .
'/dashboard/bookings?payment=success',
                'cancel_url' => config('app.frontend_url') .
'/dashboard/bookings?payment=cancelled',
                'metadata' => [
                    'booking_id' => $booking->id,
                ],
            ]);
```

Рисунок 3.6

```
        PaymentTransaction::create([
                'booking_id' => $booking->id,
                'amount' => $booking->total_price,
                'payment_method' => 'stripe',
                'payment_status' => PaymentTransaction::STATUS_PENDING,
                'transaction_id' => $session->id,
            ]);

            return response()->json([
                'session_id' => $session->id,
                'url' => $session->url,
            ]);

        } catch (ApiErrorException $e) {
            Log::error('Stripe API Error: ' . $e->getMessage());
            return response()->json([
                'message' => 'Помилка при створенні сесії оплати: ' . $e->getMessage(),
            ], 500);
        } catch (\Exception $e) {
            Log::error('Payment Error: ' . $e->getMessage());
            return response()->json([
                'message' => 'Помилка при створенні сесії оплати',
            ], 500);
        }
    }

    public function handleWebhook(Request $request): JsonResponse
    {
        $payload = $request->getContent();
        $sigHeader = $request->header('Stripe-Signature');
        $endpointSecret = config('services.stripe.webhook_secret');

        try {
            $event = \Stripe\Webhook::constructEvent(
                $payload,
                $sigHeader,
                $endpointSecret
            );
        } catch (\Exception $e) {
            Log::error('Stripe Webhook Error: ' . $e->getMessage());
            return response()->json(['error' => 'Invalid signature'], 400);
        }
```

Рисунок 3.7

```
        switch ($event->type) {
            case 'checkout.session.completed':
                $session = $event->data->object;
                $this->handleCheckoutSessionCompleted($session);
                break;
            case 'payment_intent.succeeded':
                $paymentIntent = $event->data->object;
                $this->handlePaymentIntentSucceeded($paymentIntent);
                break;
            case 'payment_intent.payment_failed':
                $paymentIntent = $event->data->object;
                $this->handlePaymentIntentFailed($paymentIntent);
                break;

            default:
                Log::info('Unhandled Stripe event: ' . $event->type);
        }
        return response()->json(['received' => true]);
    }

    private function handleCheckoutSessionCompleted($session): void
    {
        $bookingId = $session->metadata->booking_id ?? null;

        if (!$bookingId) {
            Log::warning('Stripe session without booking_id: ' . $session->id);
            return;
        }

        $transaction = PaymentTransaction::where('transaction_id', $session->id)->first();

        if ($transaction) {
            $transaction->update([
                'payment_status' => PaymentTransaction::STATUS_COMPLETED,
            ]);

            $booking = Booking::find($bookingId);
            $booking->update([
                'status_id' => BookingStatus::getPaidStatusId(),
            ]);
        }
    }
}
```

Рисунок 3.8

3.3 Реалізація інтерфейсу

Розглянемо *основні сторінки клієнтської частини*.

Головна сторінка (лендінг). Публічна сторінка застосунку, доступна без авторизації. Містить загальну інформацію про СТО та кнопку для переходу до форми авторизації. На рисунку 3.9 зображено зовнішній вигляд головної сторінки. Фрагмент коду можна побачити у Додатку А.1.

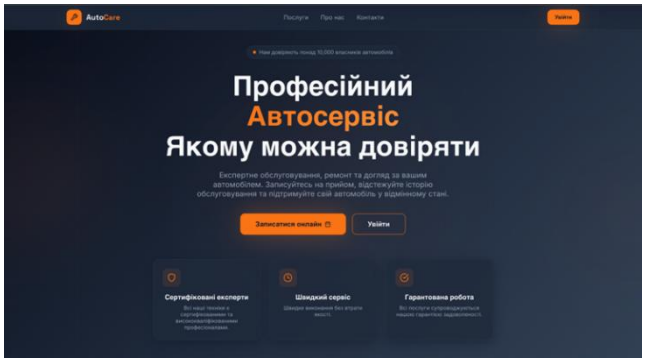


Рисунок 3.9 – Головна сторінка системи

Форма авторизації та реєстрації. Реалізована у вигляді модального вікна, що відкривається при натисканні кнопки входу. Є можливість авторизуватися в існуючий акаунт і зареєструватися. На рисунку 3.10 зображена форма авторизації. Фрагмент коду компонента AuthModal.vue наведено у Додатку А.2, а фрагмент коду серверного контролера AuthController.php – у Додатку Б.1.

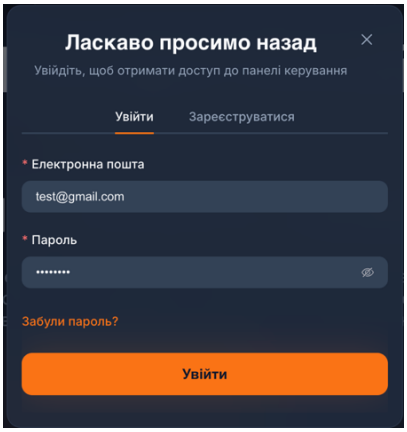


Рисунок 3.10 – Форма авторизації

Особистий кабінет клієнта (дашборд). Після авторизації клієнт потрапляє до особистого кабінету, де відображається зведена статистика: кількість автомобілів, загальна кількість записів та сума витрат за весь час. Окрім статистики, на дашборді відображається скорочений список зареєстрованих автомобілів клієнта та найближчі заплановані записи на обслуговування, що дозволяє швидко орієнтуватися у поточному стані без переходу до окремих розділів. На рисунку 3.11 зображено дашборд клієнта. Фрагмент коду компонента Dashboard.vue наведено у Додатку А.3.

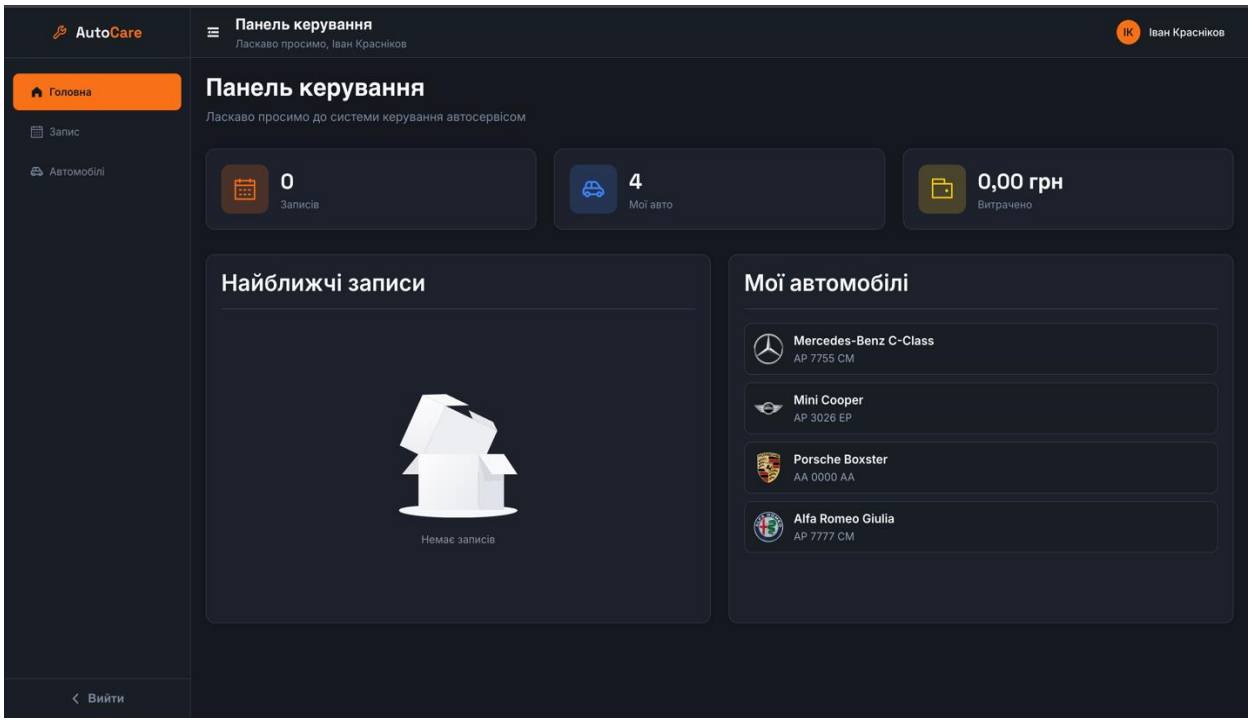


Рисунок 3.11 – Особистий кабінет клієнта

Управління автомобілями клієнта. На цій сторінці клієнт може переглянути список власних автомобілів, додати новий із зазначенням марки, моделі, року випуску, VIN-коду, номерного знаку та технічних характеристик, а також завантажити фотографії. На рисунках 3.12 – 3.13 зображено сторінку автомобілів та форму додавання авто.

Фрагмент коду компонента ClientCars.vue наведено у Додатку А.4, форму створення CreateCarModal.vue – у Додатку А.5, серверний контролер CarsController.php – у Додатку Б.2.

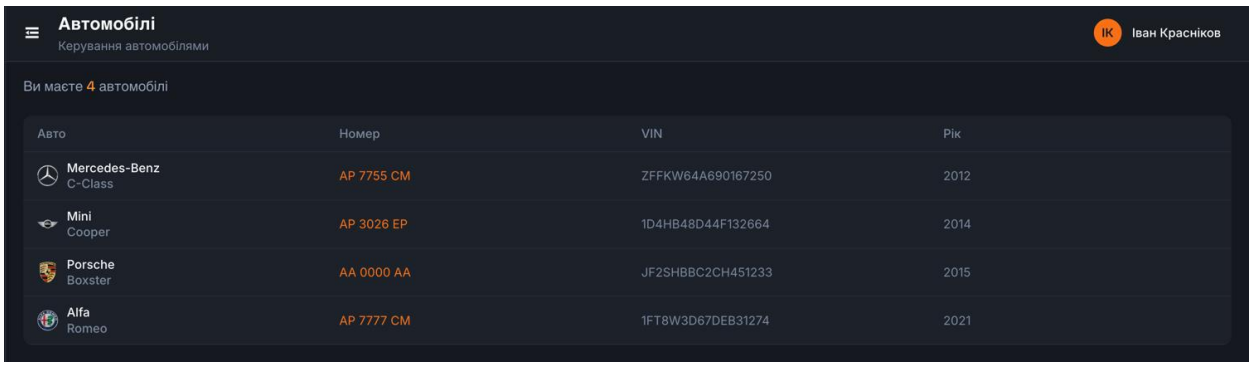


Рисунок 3.12 – Список автомобілів клієнта

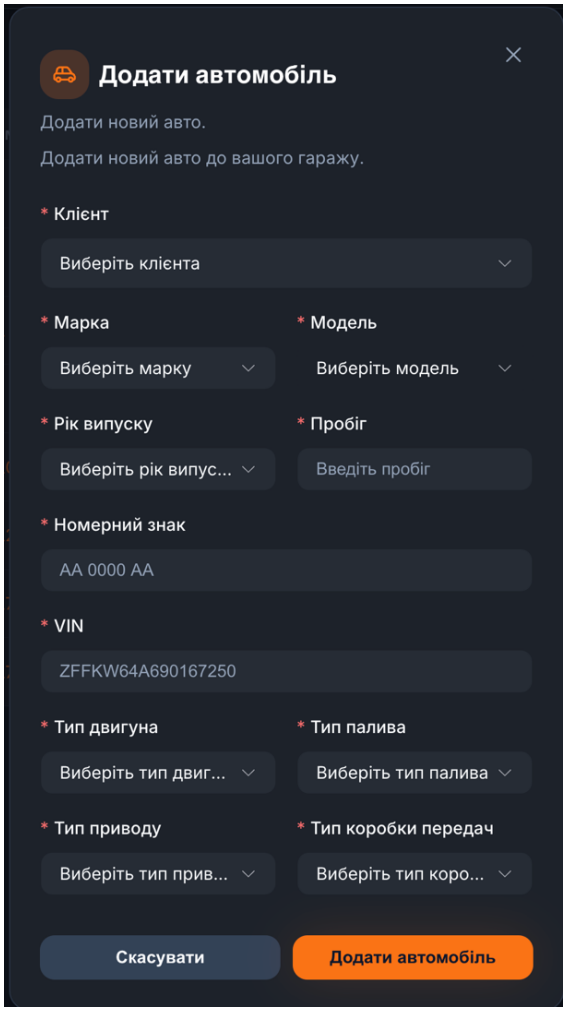


Рисунок 3.13 – Форма додавання автомобіля

Створення запису на СТО. Клієнт може створити новий запис, обравши автомобіль зі свого списку, необхідні послуги та дату обслуговування. Система автоматично розраховує попередню вартість із урахуванням року випуску автомобіля та формує детальний чек. За потреби клієнт може одразу здійснити онлайн-оплату.

На рисунку 3.14 зображено форму створення запису. Фрагмент коду компонента CreateBookingModal.vue наведено у Додатку А.6.

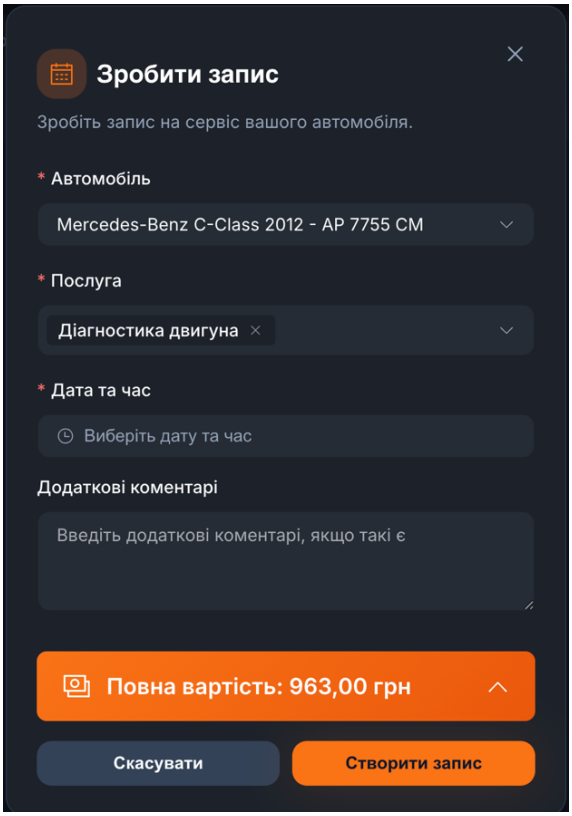


Рисунок 3.14 – Форма запису на СТО

Історія записів клієнта. На цій сторінці клієнт може переглянути всі свої записи з поточними статусами, завантажити PDF-рахунок для кожного замовлення, а також скасувати запис із зазначенням причини у коментарі.

На рисунках 3.15 – 3.16 зображено список записів та завантажений рахунок. Фрагмент коду компонента ClientBookings.vue наведено у Додатку А.7.

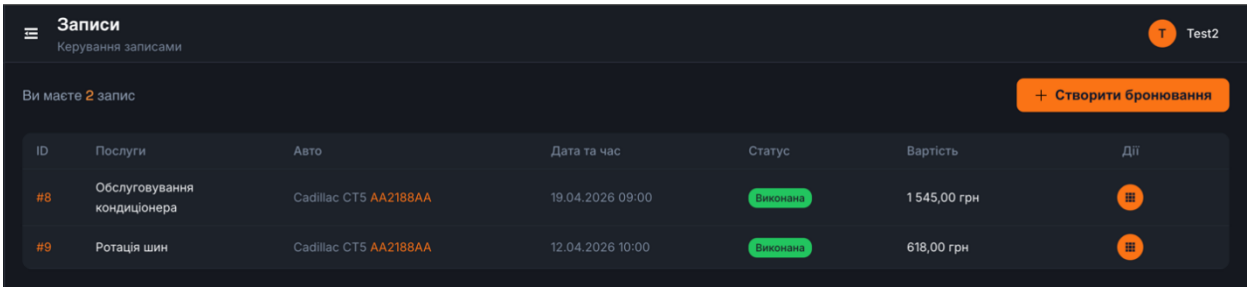


Рисунок 3.15 – Список записів клієнта

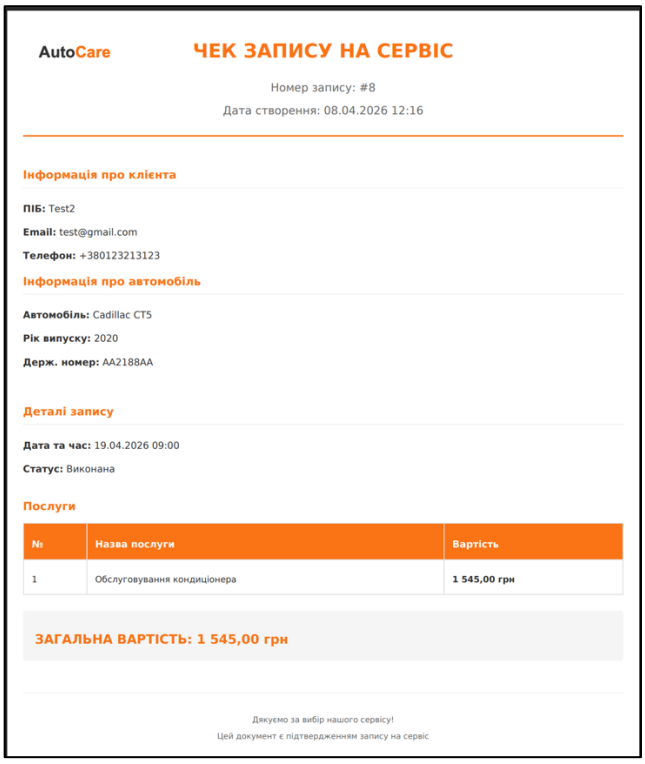


Рисунок 3.16 – Рахунок замовлення

Панель менеджера – управління клієнтами. Менеджер має доступ до повного списку клієнтів із можливістю перегляду профілю, редагування контактних даних та додавання службових нотаток. На рисунку 3.17 зображено сторінку списку клієнтів. Фрагмент коду компонента Clients.vue наведено у Додатку А.8, а серверного контролера ClientController.php – у Додатку Б.3.

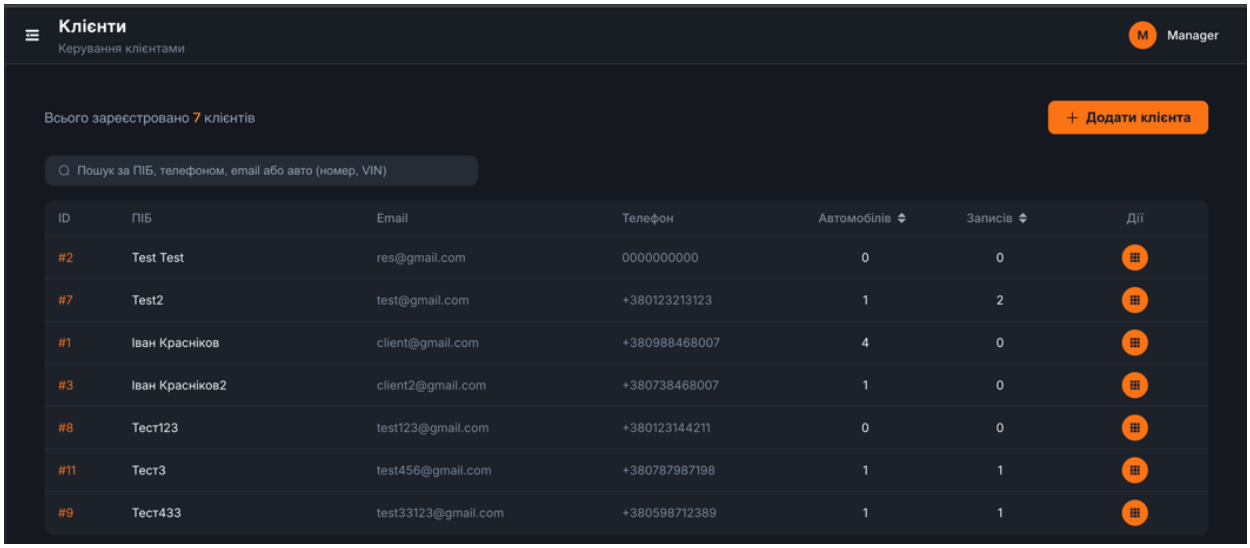


Рисунок 3.17 – Список клієнтів (панель менеджера)

Панель менеджера – управління записами. На цій сторінці менеджер переглядає всі записи клієнтів у табличному вигляді, може підтвердити, відредагувати або скасувати запис, а також змінювати статус замовлення по ланцюжку: *новий → підтверджено → в роботі → очікує оплати → завершено*. На рисунках 3.18 – 3.19 зображено список усіх записів та вікно редагування запису. Фрагмент коду компонента Bookings.vue наведено у Додатку А.9.

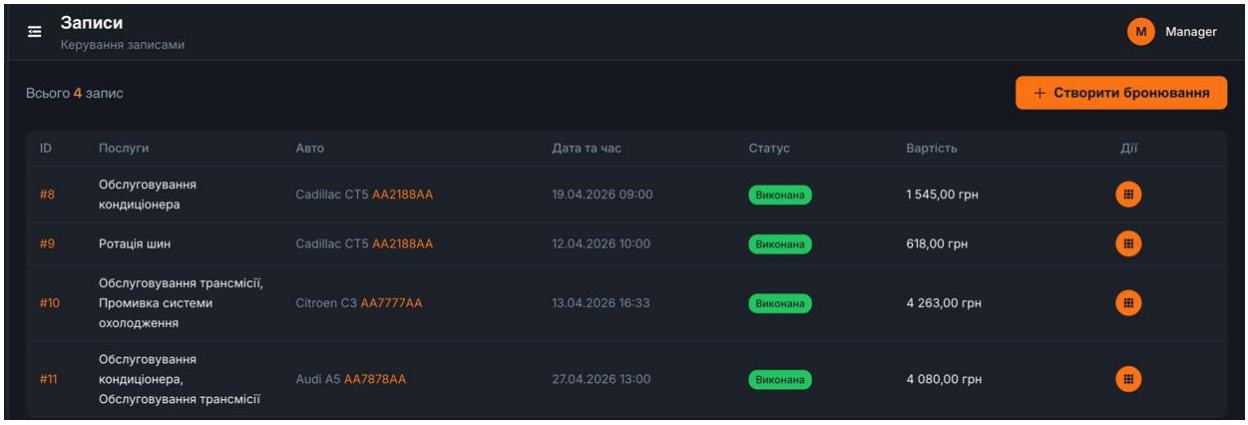


Рисунок 3.18 – Список усіх записів (панель менеджера)

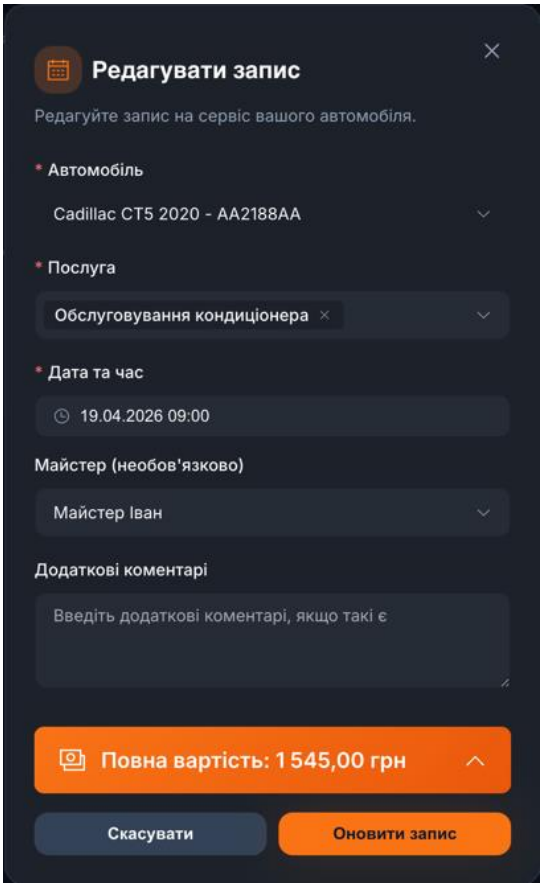


Рисунок 3.19 – Вікно редагування запису

Управління послугами (адміністратор). Адміністратор має доступ до сторінки управління переліком послуг СТО: створення, редагування та видалення послуг із зазначенням базової ціни та категорії. На рисунку 3.20 зображено сторінку управління послугами. Фрагмент коду компонента Services.vue наведено у Додатку А.10, серверного контролера ServiceController.php – у Додатку Б.4.

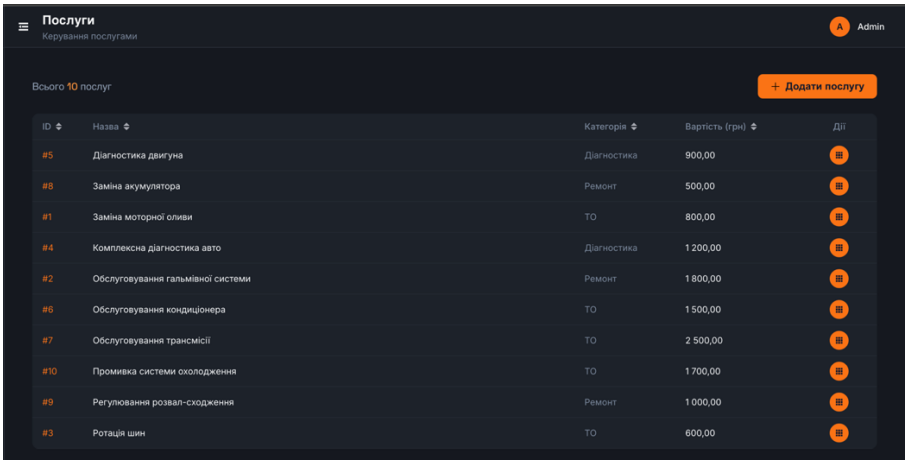


Рисунок 3.20 – Управління послугами (панель адміністратора)

Звіти та аналітика (адміністратор). Розділ аналітики надає адміністратору можливість переглядати три види звітів: дохід по місяцях, популярні послуги та розподіл записів за статусами. Звіти можна експортувати у формат Excel. На рисунку 3.21 зображено сторінку звітів. Фрагмент коду компонента Reports.vue наведено у Додатку А.11, а серверного контролера ReportController.php – у Додатку Б.5.

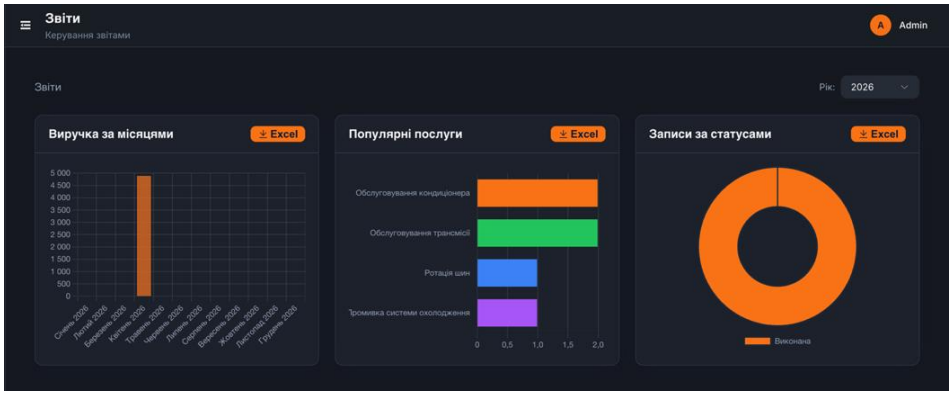


Рисунок 3.21 – Сторінка звітів та аналітики

Кабінет майстра. Майстер має окремий інтерфейс, відокремлений від панелі менеджера. Після авторизації майстер потрапляє до власного дашборду зі списком призначених йому замовлень. На головній сторінці кабінету відображаються картки замовлень з інформацією про клієнта (ім'я, телефон), автомобіль (марка, модель, номерний знак), перелік призначених послуг та загальну вартість. Є можливість фільтрації замовлень за датою та за статусом. Майстер може змінювати статус замовлення у дозволеному ланцюжку: натисканням кнопки «Взяти в роботу» переводить замовлення зі статусу підтверджено у в роботі, а кнопкою «Виконано» – з в роботі у очікує оплати. Подальше підтвердження оплати виконується менеджером. На рисунку 3.22 зображено кабінет майстра зі списком. Фрагмент коду компонента MasterDashboard.vue наведено у Додатку А.12.

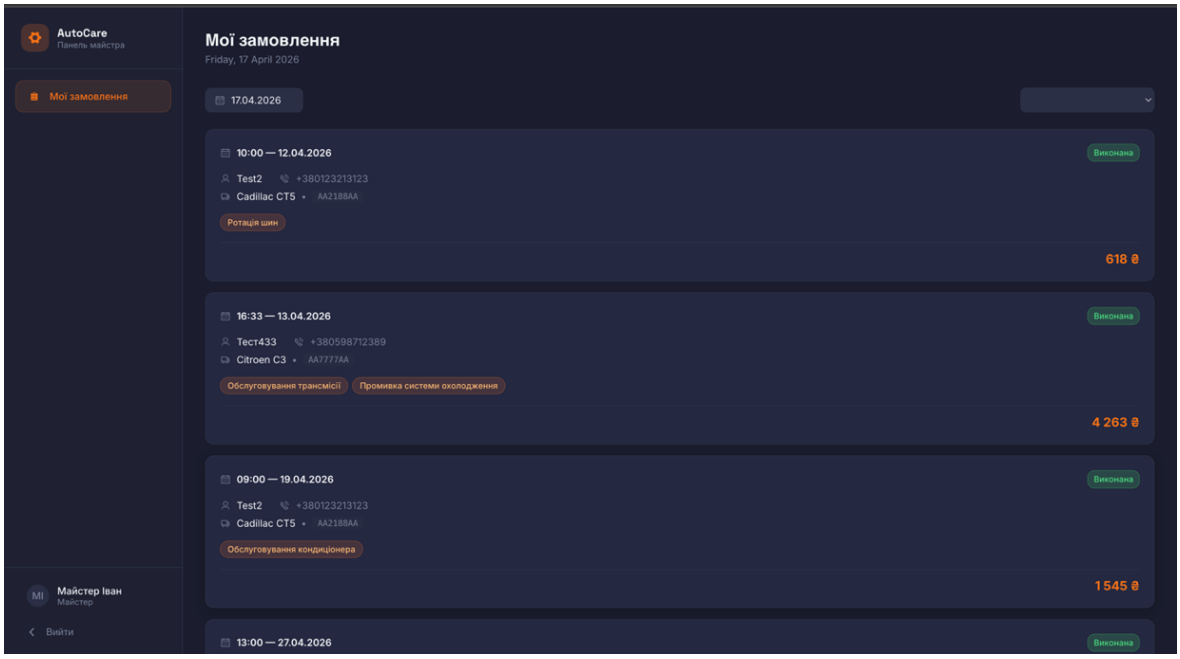


Рисунок 3.22 – Кабінет майстра: список замовлень

У третьому розділі було реалізовано всі ключові модулі системи: від авторизації та управління автомобілями до обробки записів, онлайн-оплати та формування звітів. Розроблені компоненти охоплюють повний цикл роботи кожного актора системи та підтверджують відповідність реалізації сформованим раніше вимогам і проєктним рішенням.

ВИСНОВКИ

У процесі виконання дипломної роботи розроблено повнофункціональну CRM-систему для управління станцією технічного обслуговування автомобілів. Реалізація здійснювалась із використанням сучасного технологічного стеку: Vue.js 3 (клієнтська частина), Laravel 12 (серверна логіка) та MySQL (база даних) [1, 2, 5].

Досліджено предметну область діяльності станцій технічного обслуговування автомобілів, визначено основні бізнес-процеси: прийом клієнтів, облік автомобілів, формування нарядів та контроль виконання робіт. Встановлено потребу в автоматизації цих процесів засобами сучасних CRM-систем [25].

Проаналізовано існуючі програмні рішення у сфері управління СТО: Bitrix24, YCLIENTS, АвтоДилер та 1С: Управління автосервісом. Виявлено, що жодне з них не поєднує повноцінний веборієнтований доступ, онлайн-запис, облік автомобілів та вбудовану оплату в єдиній доступній системі.

Визначено функціональні та нефункціональні вимоги до системи. Сформовано рольову модель із чотирма типами користувачів: клієнт, менеджер, адміністратор та майстер. Побудовано діаграми варіантів використання, послідовності та класів.

Спроєктовано архітектуру вебзастосунку за патерном MVC та структуру реляційної бази даних із чотирнадцятьма таблицями, нормалізованими до третьої нормальної форми [17]. Розроблено ER-модель, діаграму компонентів та діаграму розгортання системи.

Реалізовано клієнтську частину у вигляді SPA-застосунку на Vue.js 3 з використанням Vuex та Element Plus [9], а також серверний REST API на Laravel 12 із авторизацією через Laravel Sanctum [1]. Система забезпечує повний цикл роботи СТО: від реєстрації клієнта до завершення замовлення.

Забезпечено інтеграцію модуля онлайн-оплати через платіжний шлюз Stripe [7]. Реалізовано створення Checkout-сесії, обробку webhook-подій та

автоматичне оновлення статусу замовлення після успішної транзакції. Передбачено генерацію PDF-рахунків для клієнтів.

Проведено тестування основних функціональних модулів системи за допомогою PHPUnit [8]. Розроблено тест-кейси для модулів авторизації, управління послугами та бронювань. Підтверджено коректну обробку як позитивних, так і негативних сценаріїв взаємодії.

ПЕРЕЛІК ПОСИЛАНЬ

1. Laravel Documentation. *Laravel Official Website*. URL: https://laravel.com/docs (дата звернення: 05.01.2026).

2. Vue.js Guide. *Vue.js Official Website*. URL: https://vuejs.org/guide/ (дата звернення: 06.01.2026).

3. MDN Web Docs. *JavaScript & Web Technologies.* URL: https://developer.mozilla.org/ (дата звернення: 17.01.2026)

4. Laravel Tutorials. *DigitalOcean Community*. URL: https://www.digitalocean.com/community/tutorials (дата звернення: 17.01.2026).

5. MySQL Reference Manual. *MySQL Official Documentation.* URL: https://dev.mysql.com/doc/ (дата звернення: 20.02.2026).

6. Stack Overflow Developer Survey 2024. *Stack Overflow.* URL: https://survey.stackoverflow.co/2024/ (дата звернення: 21.02.2026).

7. Stripe Payments Documentation. *Stripe Official Website.* URL: https://stripe.com/docs (дата звернення: 04.03.2026).

8. PHPUnit Manual. *PHPUnit Official Website*. URL: https://phpunit.de/documentation.html (дата звернення: 04.03.2026).

9. Element Plus Component Library. *Element Plus Official Website*. URL: https://element-plus.org/en-US/ (дата звернення: 10.03.2026).

10. Vue Router Documentation. *Vue Router Official Website.* URL: https://router.vuejs.org/ (дата звернення: 10.03.2026).

11. Vuex State Management. *Vuex Official Website.* URL: https://vuex.vuejs.org/ (дата звернення: 10.03.2026).

12. Stauffer M. Laravel: Up & Running : 3rd ed. Sebastopol : O’Reilly Media, 2023. 450 p.

13. Shavin M. Learning Vue. Sebastopol : O’Reilly Media, 2023. 354 p.

14. Hanchett E., Listwon B. Vue.js in Action. Shelter Island : Manning Publications, 2019. 304 p.

15. Flanagan D. JavaScript: The Definitive Guide : 7th ed. Sebastopol : O’Reilly Media, 2020. 706 p.

16. Zandstra M. PHP 8 Objects, Patterns, and Practice : 6th ed. New York : Apress, 2021. 696 p.

17. Welling L., Thomson L. PHP and MySQL Web Development : 5th ed. Boston : Addison-Wesley, 2017. 1008 p.

18. Botros S., Tinley J. High Performance MySQL : 4th ed. Sebastopol : O’Reilly Media, 2022. 558 p.

19. Sommerville I. Software Engineering : 10th ed. London : Pearson Education, 2016. 816 p.

20. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner’s Approach : 8th ed. New York : McGraw-Hill Education, 2015. 976 p.

21. Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley Professional, 2002. 560 p.

22. Martin R. C. Clean Architecture: A Craftsman’s Guide to Software Structure and Design. Upper Saddle River : Prentice Hall, 2017. 432 p.

23. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston : Addison-Wesley, 1994. 395 p.

24. Richardson L., Amundsen M. RESTful Web APIs. Sebastopol : O’Reilly Media, 2013. 408 p.

25. Greenberg P. CRM at the Speed of Light : 4th ed. New York : McGraw-Hill, 2009. 668 p.

ДОДАТОК А

Програмна реалізація Front-End частини

Для розробки клієнтської частини застосунку використовувався Vue.js 3 у поєднанні з Vuex 4 для керування станом та Vue Router 4 для навігації між сторінками. HTTP-запити до серверного API реалізовані за допомогою бібліотеки Axios, а оформлення інтерфейсу – з використанням UI-бібліотеки Element Plus.

Реалізація клієнтської частини включає написання модулів стану (Vuex store), конфігурації маршрутизатора, модальних компонентів для роботи з формами, а також сторінкових компонентів для відображення даних. У додатку А представлені основні фрагменти коду компонентів Vue.js.

А.1 Файл Home.vue

```
<template>
  <div class="home-page">
    <LandingHeader @loginClick="openAuthModal" />
    <HeroSection @getStarted="openAuthModal" />
    <LandingFooter />
    <AuthModal :isOpen="isAuthModalOpen" @close="closeAuthModal" />
  </div>
</template>

<script>
import LandingHeader from '../../layout/LandingHeader.vue';
import LandingFooter from '../../layout/LandingFooter.vue';
import HeroSection from './HeroSection.vue';
import AuthModal from '../../modals/AuthModal.vue';

export default {
  name: 'Home',
  components: {
    LandingHeader,
```

```
            LandingFooter,
            HeroSection,
            AuthModal
        },
        data() {
            return {
                isAuthModalOpen: false
            };
        },
        methods: {
            openAuthModal() {
                this.isAuthModalOpen = true;
            },
            closeAuthModal() {
                this.isAuthModalOpen = false;
            }
        }
    };
    </script>
```

А.2 Фрагмент файлу AuthModal.vue

```
    <template>
        <el-dialog
            v-model="dialogVisible"
            :title="mode === 'login' ? 'Ласкаво просимо назад' : 'Створити акаунт'"
            width="450px"
            :close-on-click-modal="false"
            :close-on-press-escape="true"
            @close="handleClose"
            class="auth-modal-dialog"
        >
            <template #header>
                <div class="auth-modal-header">
                    <h2 class="auth-modal-title">
                        {{ mode === 'login' ? 'Ласкаво просимо назад' : 'Створити акаунт' }}
                    </h2>
                    <p class="auth-modal-subtitle">
                        {{ mode === 'login'
                            ? 'Увійдіть, щоб отримати доступ до панелі керування'
                            : 'Зареєструйтеся, щоб почати керувати обслуговуванням
автомобіля'
```

```
                    }}
                  </p>
                </div>
              </template>

              <el-tabs v-model="mode" class="auth-modal-tabs">
                <el-tab-pane label="Увійти" name="login">
                  <el-form
                    ref="loginFormRef"
                    :model="formData"
                    :rules="loginRules"
                    label-position="top"
                    @submit.prevent="handleSubmit"
                  >
                    <el-form-item label="Електронна пошта" prop="email">
                      <el-input
                        v-model="formData.email"
                        type="email"
                        placeholder="ivan@example.com"
                        size="large"
                      />
                    </el-form-item>

                    <el-form-item label="Пароль" prop="password">
                      <el-input
                        v-model="formData.password"
                        :type="showPassword ? 'text' : 'password'"
                        size="large"
                        @click:append-icon="togglePassword"
                        show-password
                      />
                    </el-form-item>

                    <el-form-item>
                      <div class="text-right">
                        <el-link type="primary" :underline="false"
@click="handleForgotPassword">
                          Забули пароль?
                        </el-link>
                      </div>
                    </el-form-item>

                    <el-form-item>
                      <el-button
                        type="primary"
```

```
                    size="large"
                    class="w-full"
                    native-type="submit"
                    :loading="loading"
                  >
                    Увійти
                  </el-button>
                </el-form-item>
              </el-form>
            </el-tab-pane>

            <el-tab-pane label="Зареєструватися" name="register">
              <el-form
                ref="registerFormRef"
                :model="formData"
                :rules="registerRules"
                label-position="top"
                @submit.prevent="handleSubmit"
              >
                <el-form-item label="Повне ім'я" prop="name">
                  <el-input
                    v-model="formData.name"
                    type="text"
                    placeholder="Іван Іванов"
                    size="large"
                  />
                </el-form-item>

                <el-form-item label="Електронна пошта" prop="email">
                  <el-input
                    v-model="formData.email"
                    type="email"
                    placeholder="ivan@example.com"
                    size="large"
                  />
                </el-form-item>

                <el-form-item label="Телефон" prop="phone">
                  <PhoneInput
                    v-model="formData.phone"
                    size="large"
                    placeholder="+380 (00) 000 00 00"
                  />
                </el-form-item>
```

```
                    <el-form-item label="Пароль" prop="password">
                      <el-input
                        v-model="formData.password"
                        :type="showPassword ? 'text' : 'password'"
                        placeholder="••••••••"
                        size="large"
                        show-password
                      />
                    </el-form-item>

                    <el-form-item label="Підтвердження пароля"
prop="password_confirmation">
                      <el-input
                        v-model="formData.password_confirmation"
                        :type="showPassword ? 'text' : 'password'"
                        placeholder="••••••••"
                        size="large"
                        show-password
                      />
                    </el-form-item>

                    <el-form-item>
                      <el-button
                        type="primary"
                        size="large"
                        class="w-full"
                        native-type="submit"
                        :loading="loading"
                      >
                        Створити акаунт
                      </el-button>
                    </el-form-item>
                  </el-form>
                </el-tab-pane>
              </el-tabs>
            </el-dialog>
          </template>
```

А.3 Фрагмент файлу Dashboard.vue

```
          <template>
            <DashboardLayout>
```

```
            <div class="dashboard-page">
              <div class="page-header">
                <h1 class="page-title">Панель керування</h1>
                <p class="page-subtitle">Ласкаво просимо до системи керування автосервісом</p>
              </div>

              <div class="stats-grid">
                <el-card class="stat-card" v-loading="isLoading">
                  <div class="stat-content">
                    <div class="stat-icon bookings">
                      <el-icon :size="32"><Calendar /></el-icon>
                    </div>
                    <div class="stat-info">
                      <div class="stat-value">{{ isUserClient ? stats.total_bookings : managerStats.total_bookings }}</div>
                      <div class="stat-label">Записів</div>
                    </div>
                  </div>
                </el-card>

                <el-card class="stat-card" v-loading="isLoading">
                  <div class="stat-content">
                    <div class="stat-icon clients">
                      <CarIcon :size="32" />
                    </div>
                    <div class="stat-info">
                      <div class="stat-value">{{ isUserClient ? stats.total_cars : managerStats.total_cars }}</div>
                      <div class="stat-label">{{ isUserClient ? 'Мої авто' : 'Авто' }}</div>
                    </div>
                  </div>
                </el-card>

                <el-card class="stat-card" v-loading="isLoading">
                  <div class="stat-content">
                    <div class="stat-icon revenue">
                      <el-icon :size="32">
                        <Wallet v-if="isUserClient" />
                        <User v-else />
                      </el-icon>
                    </div>
                    <div class="stat-info">
                      <div class="stat-value">{{ isUserClient ? formatCurrency(stats.total_spent) + ' грн' : managerStats.total_clients }}</div>
```

```
                        <div class="stat-label">{{ isUserClient ? 'Витрачено' : 'Клієнти' }}</div>
                    </div>
                </div>
            </el-card>
        </div>

        <div class="dashboard-grid">
            <el-card class="dashboard-card" v-loading="isLoading">
                <template #header>
                    <div class="card-header">
                        <span>Найближчі записи</span>
                    </div>
                </template>
                <div class="dashboard-card__list">
                    <div v-if="bookings.length === 0" class="empty-state">
                        <el-empty description="Немає записів" />
                    </div>
                    <div v-else
                        v-for="booking in bookings.slice(0, 3)"
                        :key="booking.id"
                        class="dashboard-card__item"
                        >
                        <div class="dashboard-card_item-info">
                            <div class="text">
                                <div class="title">{{ booking.services.map(service => service.name).join(', ') }}</div>
                                <div class="meta">
                                    {{ booking.date }} • <span class="orange-text">{{ booking.car?.license_plate }}</span>
                                </div>
                            </div>
                        </div>
                        <span class="status" :class="BOOKING_STATUS_CLASS_MAPPING[booking.status_id]">{{ booking.status?.name || 'Невідомо' }}</span>
                    </div>
                </div>
            </el-card>
            <el-card class="dashboard-card" v-loading="isLoading" v-if="!isManagerOrAdmin">
                <template #header>
                    <div class="card-header">
                        <span>Мої автомобілі</span>
                    </div>
```

```
                    </template>
                    <div class="dashboard-card__list">
                      <div v-if="cars.length === 0" class="empty-state">
                        <el-empty description="Немає авто" />
                      </div>
                      <div v-else
                        v-for="car in cars"
                        :key="car.id"
                        class="dashboard-card__item"
                      >
                        <div class="dashboard-card_item-info">
                          <div class="icon">
                            <img :src="car.brand_logo" alt="Car Brand Logo">
                          </div>
                          <div class="text">
                            <div class="title">{{ car.full_name }}</div>
                            <div class="meta">
                              {{ car.license_plate }}
                            </div>
                          </div>
                        </div>
                        <span class="add-info" v-if="car.latest_booking">
                          <span     class="label">{{     getBookingLabel(car.latest_booking)
}}</span>
                          <span class="value">{{ car.latest_booking?.date }}</span>
                        </span>
                      </div>
                    </div>
                  </el-card>
                </div>
              </div>
            </DashboardLayout>
          </template>
```

А.4 Фрагмент файлу ClientCars.vue

```
          <template>
            <DashboardLayout>
              <div class="cars-container">
                <div class="top">
                  <h1 class="title">Всього зареєстровано <span class="count">{{ cars.length
}}</span> {{ carsWord }}</h1>
```

```
                <button class="create-car-btn" @click="openCreateCarModal">
                  <el-icon><Plus /></el-icon>
                  <span class="text">Додати автомобіль</span>
                </button>
              </div>
              <CarsTable :cars="cars" :loading="loading" />
            </div>

            <CreateCarModal :isOpen="isCreateCarModalOpen"
@close="closeCreateCarModal" managerMode :editingCarId="editingCarId" />
          </DashboardLayout>
        </template>
```

А.5 Фрагмент файлу CreateCarModal.vue

```
        <template>
          <el-dialog
            v-model="dialogVisible"
            title="Додати автомобіль"
            width="450px"
            :close-on-click-modal="false"
            :close-on-press-escape="true"
            @close="handleClose"
            class="create-form-modal-dialog"
          >
            <template #header>
              <div class="create-form-modal-header">
                <h2 class="create-form-modal-title">
                  <div class="icon">
                    <CarIcon :size="32" />
                  </div>
                  {{ editingCarId ? 'Редагувати автомобіль' : 'Додати автомобіль' }}
                </h2>
                <p class="create-form-modal-description" v-if="!editingCarId &&
managerMode">
                  Додати новий авто.
                </p>
                <p class="create-form-modal-description">
                  {{ editingCarId ? 'Редагувати автомобіль' : 'Додати новий авто до
вашого гаражу.' }}
                </p>
```

```
    </div>
  </template>

  <el-form
    ref="createCarFormRef"
    :model="formData"
    :rules="formRules"
    label-position="top"
    @submit.prevent="handleSubmit"
  >
    <div class="form-row" v-if="managerMode">
      <el-form-item label="Клієнт" prop="client_id">
        <el-select
          v-model="formData.client_id"
          placeholder="Виберіть клієнта"
          clearable
          filterable
        >
          <el-option
            v-for="client in clients"
            :key="client.id"
            :label="getClientLabel(client)"
            :value="client.id"
          >
            <span>{{ getClientLabel(client) }}</span>
          </el-option>
        </el-select>
      </el-form-item>
    </div>

    <div class="form-row">
      <el-form-item label="Марка" prop="brand">
        <el-select
          v-model="formData.brand"
          placeholder="Виберіть марку"
          :options="carBrands"
          :props="{ label: 'name', value: 'id' }"
          clearable
        />
      </el-form-item>
      <el-form-item label="Модель" prop="model">
        <el-select
          v-model="formData.model"
          placeholder="Виберіть модель"
          :options="filteredCarModels"
```

```
                :disabled="!formData.brand"
                :props="{ label: 'name', value: 'id' }"
                clearable
              />
            </el-form-item>
          </div>

          <div class="form-row">
            <el-form-item label="Рік випуску" prop="year">
              <el-select
                v-model="formData.year"
                placeholder="Виберіть рік випуску"
                :options="carYears"
                :props="{ label: 'name', value: 'id' }"
                clearable
              >
                <el-option
                  v-for="year in carYears"
                  :key="year.id"
                  :label="year.name"
                  :value="year.id"
                />
              </el-select>
            </el-form-item>

            <el-form-item label="Пробіг" prop="mileage">
              <el-input
                v-model="formData.mileage"
                placeholder="Введіть пробіг"
                type="number"
                min="0"
                max="1000000"
                clearable
              />
            </el-form-item>
          </div>

          <div class="form-row">
            <el-form-item label="Номерний знак" prop="licence_plate">
              <el-input
                v-model="formData.licence_plate"
                placeholder="AA 0000 AA"
                type="text"
                clearable
                maxlength="10"
```

```
            @input="formatLicencePlateValue"
            style="text-transform: uppercase;"
          />
        </el-form-item>
      </div>

      <div class="form-row">
        <el-form-item label="VIN" prop="vin">
          <el-input
            v-model="formData.vin"
            placeholder="ZFFKW64A690167250"
            type="text"
            clearable
          />
        </el-form-item>
      </div>

      <template v-if="isAddInfoAvailable">
        <div class="form-row">
          <el-form-item label="Тип двигуна" prop="engine_type">
            <el-select
              v-model="formData.engine_type"
              placeholder="Виберіть тип двигуна"
              :options="engineTypes"
              :props="{ label: 'name', value: 'id' }"
            />
          </el-form-item>

          <el-form-item label="Тип палива" prop="fuel_type">
            <el-select
              v-model="formData.fuel_type"
              placeholder="Виберіть тип палива"
              :options="fuelTypes"
              :props="{ label: 'name', value: 'id' }"
            />
          </el-form-item>
        </div>

        <div class="form-row">
          <el-form-item label="Тип приводу" prop="drive_unit_type">
            <el-select
              v-model="formData.drive_unit_type"
              placeholder="Виберіть тип приводу"
              :options="driveUnitTypes"
              :props="{ label: 'name', value: 'id' }"
```

```
                        />
                    </el-form-item>

                    <el-form-item label="Тип коробки передач" prop="gearbox_type">
                        <el-select
                            v-model="formData.gearbox_type"
                            placeholder="Виберіть тип коробки передач"
                            :options="gearboxTypes"
                            :props="{ label: 'name', value: 'id' }"
                        />
                    </el-form-item>
                </div>
            </template>
        </el-form>

        <template #footer>
            <div class="create-form-modal-footer">
                <el-button @click="handleClose">Скасувати</el-button>
                <el-button type="primary" @click="handleSubmit">{{ editingCarId ?
'Оновити автомобіль' : 'Додати автомобіль' }}</el-button>
            </div>
        </template>
    </el-dialog>
</template>
```

А.6 Фрагмент файлу CreateBookingModal.vue

```
<template>
    <el-dialog
        v-model="dialogVisible"
        title="Зробити запис"
        width="450px"
        :close-on-click-modal="false"
        :close-on-press-escape="true"
        @close="handleClose"
        class="create-form-modal-dialog"
    >
        <template #header>
            <div class="create-form-modal-header">
                <h2 class="create-form-modal-title">
                    <div class="icon">
                        <el-icon><Calendar /></el-icon>
```

```
                    </div>
                    {{ editingBookingId ? 'Редагувати запис' : 'Зробити запис' }}
                  </h2>
                  <p class="create-form-modal-description">
                    {{ editingBookingId ? 'Редагуйте запис на сервіс вашого автомобіля.' :
'Зробіть запис на сервіс вашого автомобіля.' }}
                  </p>
                </div>
              </template>

              <el-form
                ref="createBookingFormRef"
                :model="formData"
                :rules="formRules"
                label-position="top"
                @submit.prevent="handleSubmit"
              >
                <div class="form-row">
                  <el-form-item label="Автомобіль" prop="car">
                    <el-select
                      v-model="formData.car"
                      placeholder="Виберіть автомобіль"
                      clearable
                      :disabled="editingBookingId"
                    >
                      <el-option
                        v-for="car in cars"
                        :key="car.id"
                        :label="carOptionLabel(car)"
                        :value="Number(car.id)"
                      />
                    </el-select>
                  </el-form-item>
                </div>

                <div class="form-row">
                  <el-form-item label="Послуга" prop="services">
                    <el-select
                      v-model="formData.services"
                      placeholder="Виберіть послугу"
                      :options="services"
                      :props="{ label: 'name', value: 'id' }"
                      clearable
                      multiple
                    />
```

```
        </el-form-item>
      </div>

      <div class="form-row">
        <el-form-item label="Дата та час" prop="date">
          <el-date-picker
            v-model="formData.date"
            type="datetime"
            placeholder="Виберіть дату та час"
            format="DD.MM.YYYY HH:mm"
            date-format="DD.MM.YYYY"
            time-format="HH:mm"
            value-format="YYYY-MM-DD HH:mm:ss"
            :disabled-date="disabledDate"
            :disabled-time="disabledTime"
          />
        </el-form-item>
      </div>

      <div class="form-row" v-if="isManagerOrAdmin">
        <el-form-item label="Майстер (необов'язково)" prop="master_id">
          <el-select
            v-model="formData.master_id"
            placeholder="Оберіть майстра"
            filterable
            clearable
            @change="checkMasterSlot"
          >
            <el-option
              v-for="m in mastersList"
              :key="m.id"
              :label="m.full_name"
              :value="m.id"
            />
          </el-select>
        </el-form-item>
        <el-alert
          v-if="masterConflict"
          type="warning"
          :closable="false"
          style="margin-top: 4px;"
        >{{ masterConflict }}</el-alert>
      </div>

      <div class="form-row">
```

```
                    <el-form-item label="Додаткові коментарі" prop="comment">
                      <el-input
                        v-model="formData.comment"
                        :rows="3"
                        type="textarea"
                        placeholder="Введіть додаткові коментарі, якщо такі є"
                      />
                    </el-form-item>
                  </div>
                </el-form>

                <div class="price-collapse-container" v-if="priceForServices.total_price > 0">
                  <div class="price-collapse-header" :class="{ 'open': priceCollapseOpen }"
@click="priceCollapseOpen = !priceCollapseOpen">
                    <div class="price-header-left">
                      <el-icon class="price-icon"><Money /></el-icon>
                      <span class="price-header-text">Повна вартість: {{
formatPrice(priceForServices.total_price) }} грн</span>
                    </div>
                    <el-icon class="collapse-icon" :class="{ 'rotated': priceCollapseOpen }">
                      <ArrowUp />
                    </el-icon>
                  </div>
                  <div class="price-collapse-content" v-show="priceCollapseOpen">
                    <div class="price-details-list">
                      <div
                        class="price-detail-item"
                        v-for="service in priceForServices.services"
                        :key="service.id"
                      >
                        <span class="service-name">{{ service.name }}</span>
                        <span class="service-price">{{ formatPrice(service.final_price) }}
грн</span>
                      </div>
                      <div class="price-detail-total">
                        <span class="total-label">Загальна вартість</span>
                        <span class="total-price">{{ formatPrice(priceForServices.total_price)
}} грн</span>
                      </div>
                    </div>
                  </div>
                </div>

                <template #footer>
                  <div class="create-form-modal-footer">
```

```
                <el-button @click="handleClose">Скасувати</el-button>
                <el-button type="primary" @click="handleSubmit">{{ editingBookingId ? 'Оновити запис' : 'Створити запис' }}</el-button>
              </div>
            </template>
          </el-dialog>
        </template>
```

А.7 Фрагмент файлу ClientBookings.vue

```
        <template>
          <DashboardLayout>
            <div class="bookings-container">
              <div class="top">
                <h1 class="title">Всього <span class="count">{{ bookings.length }}</span> {{ bookingsWord }}</h1>

                <button class="create-booking-btn" @click="openCreateBookingModal">
                  <el-icon><Plus /></el-icon>
                  <span class="text">Створити бронювання</span>
                </button>
              </div>
              <BookingsTable :bookings="bookings" :loading="loading" @edit-booking="editBooking" />
            </div>

            <CreateBookingModal :isOpen="isCreateBookingModalOpen" :editingBookingId="editingBookingId" @close="closeCreateBookingModal" />
          </DashboardLayout>
        </template>
```

А.8 Фрагмент файлу ClientsTable.vue

```
        <template>
          <el-table v-loading="loading" :data="clients" class="dash-table">
            <el-table-column prop="id" label="ID" width="60">
              <template #default="scope">
                <span class="orange-text">#{{ scope.row.id }}</span>
```

```
            </template>
          </el-table-column>
          <el-table-column prop="full_name" label="ПІБ" width="200">
            <template #default="scope">
              <span class="white-text">{{ scope.row.full_name }}</span>
            </template>
          </el-table-column>
          <el-table-column prop="email" label="Email" width="200">
            <template #default="scope">
              <span>{{ scope.row.email }}</span>
            </template>
          </el-table-column>
          <el-table-column prop="phone" label="Телефон" width="150">
            <template #default="scope">
              <span>{{ scope.row.phone || '-' }}</span>
            </template>
          </el-table-column>
          <el-table-column prop="cars_count" sortable label="Автомобілів" width="120"
align="center">
            <template #default="scope">
              <span class="white-text">{{ scope.row.cars_count || 0 }}</span>
            </template>
          </el-table-column>
          <el-table-column prop="bookings_count" sortable label="Записів" width="100"
align="center">
            <template #default="scope">
              <span class="white-text">{{ scope.row.bookings_count || 0 }}</span>
            </template>
          </el-table-column>
          <el-table-column label="Дії" width="120" align="center">
            <template #default="scope">
              <el-dropdown placement="bottom">
                <el-button><el-icon><Grid /></el-icon></el-button>
                <template #dropdown>
                  <el-dropdown-menu>
                    <el-dropdown-item @click="viewClient(scope.row.id)">
                      <el-icon><View /></el-icon>
                      Переглянути
                    </el-dropdown-item>
                    <el-dropdown-item @click="deleteClient(scope.row)" class="danger-
item">
                      <el-icon><Delete /></el-icon>
                      Видалити
                    </el-dropdown-item>
                  </el-dropdown-menu>
```

```
                    </template>
                  </el-dropdown>
                </template>
              </el-table-column>
            </el-table>
          </template>
```

А.9 Фрагмент файлу BookingsTable.vue

```
          <template>
            <div>
              <el-table v-loading="loading" :data="bookings" class="dash-table">
                <el-table-column prop="id" label="ID" width="30">
                  <template #default="scope">
                    <span class="orange-text">#{{ scope.row.id }}</span>
                  </template>
                </el-table-column>
                <el-table-column prop="services" label="Послуги" width="100">
                  <template #default="scope">
                    <span class="white-text">{{ scope.row.services.map(service =>
service.name).join(', ') }}</span>
                  </template>
                </el-table-column>
                <el-table-column prop="car" label="Авто" width="130">
                  <template #default="scope">
                    <span>{{ scope.row.car.full_name }}</span>
                    <span class="orange-text">{{ ' ' + scope.row.car.license_plate }}</span>
                  </template>
                </el-table-column>
                <el-table-column prop="date" label="Дата та час" width="100">
                  <template #default="scope">
                    <span>{{ scope.row.date }}</span>
                  </template>
                </el-table-column>
                <el-table-column prop="status_name" label="Статус" width="80">
                  <template #default="scope">
                    <span class="status-badge"
:class="BOOKING_STATUS_CLASS_MAPPING[scope.row.status_id]">{{
scope.row.status_name }}</span>
                  </template>
                </el-table-column>
                <el-table-column prop="total_price" label="Вартість" width="70">
                  <template #default="scope">
```

```
                    <span class="white-text">{{ formatPrice(scope.row.total_price) }} грн</span>
                  </template>
                </el-table-column>
                <el-table-column label="Дії" width="100" align="center">
                  <template #default="scope">
                    <el-dropdown placement="bottom">
                      <el-button> <el-icon><Grid /></el-icon>  </el-button>
                      <template #dropdown>
                        <el-dropdown-menu>
                          <!-- Online payment: only for clients, only when pending payment -->
                          <el-dropdown-item
                            v-if="isClient && isBookingPendingPayment(scope.row)"
                            @click="payBooking(scope.row.id)"
                          ><el-icon><Money /></el-icon> Сплатити онлайн</el-dropdown-item>
                          <!-- Manual paid: only for manager/admin, only when pending payment -->
                          <el-dropdown-item
                            v-if="isManagerOrAdmin && isBookingPendingPayment(scope.row)"
                            @click="markPaid(scope.row)"
                          ><el-icon><Money /></el-icon> Сплачено</el-dropdown-item>
                          <!-- Receipt: only when work is done (pending payment, paid, or completed) -->
                          <el-dropdown-item
                            v-if="isBookingReceiptAvailable(scope.row)"
                            @click="downloadReceipt(scope.row.id)"
                          ><el-icon><Tickets /></el-icon> Скачати чек</el-dropdown-item>
                          <!-- Edit: only for manager/admin -->
                          <el-dropdown-item
                            v-if="isManagerOrAdmin"
                            @click="editBooking(scope.row.id)"
                          ><el-icon><Edit /></el-icon> Редагувати</el-dropdown-item>
                          <!-- Confirm: manager/admin, new bookings only -->
                          <el-dropdown-item
                            v-if="isManagerOrAdmin && isBookingNew(scope.row)"
                            @click="openConfirmDialog(scope.row)"
                          ><el-icon><Check /></el-icon> Підтвердити</el-dropdown-item>
                        </el-dropdown-menu>
                      </template>
                    </el-dropdown>
                  </template>
                </el-table-column>
              </el-table>
```

```
            <!-- Confirm dialog with master selection -->
            <el-dialog
              v-model="confirmDialogVisible"
              title="Підтвердження запису"
              width="460px"
              :close-on-click-modal="false"
              class="create-form-modal-dialog"
            >
              <template #header>
                <div class="create-form-modal-header">
                  <h2 class="create-form-modal-title">
                    <div class="icon"><el-icon><Check /></el-icon></div>
                    Підтвердження запису
                  </h2>
                  <p class="create-form-modal-description">Оберіть майстра та підтвердіть запис.</p>
                </div>
              </template>

              <div v-if="confirmingBooking" style="margin-bottom: 12px;">
                <p style="margin-bottom: 4px;"><strong>Запис #{{ confirmingBooking.id }}</strong></p>
                <p style="color: var(--el-text-color-secondary); font-size: 13px;">
                  {{ confirmingBooking.date }} • {{ confirmingBooking.services?.map(s => s.name).join(', ') }}
                </p>
              </div>

              <el-form label-position="top">
                <el-form-item label="Майстер" required>
                  <el-select
                    v-model="confirmForm.master_id"
                    placeholder="Оберіть майстра"
                    filterable
                    style="width: 100%"
                    @change="checkMasterAvailability"
                  >
                    <el-option
                      v-for="m in mastersList"
                      :key="m.id"
                      :label="m.full_name"
                      :value="m.id"
                    />
                  </el-select>
                </el-form-item>
```

```
                    <el-alert
                      v-if="masterConflict"
                      type="warning"
                      :closable="false"
                      style="margin-bottom: 12px;"
                    >
                      {{ masterConflict }}
                    </el-alert>

                    <el-form-item label="Змінити час (необов'язково)">
                      <el-date-picker
                        v-model="confirmForm.date"
                        type="datetime"
                        placeholder="Залишити поточний"
                        format="DD.MM.YYYY HH:mm"
                        value-format="YYYY-MM-DD HH:mm:ss"
                        style="width: 100%"
                        @change="checkMasterAvailability"
                      />
                    </el-form-item>
                  </el-form>

                  <template #footer>
                    <div class="create-form-modal-footer">
                      <el-button @click="confirmDialogVisible = false">Скасувати</el-
button>
                      <el-button
                        type="primary"
                        :loading="confirmLoading"
                        :disabled="!confirmForm.master_id || !!masterConflict"
                        @click="submitConfirm"
                      >Підтвердити</el-button>
                    </div>
                  </template>
                </el-dialog>
              </div>
            </template>
```

A.10 Фрагмент файлу CreateServiceModal.vue

```
            <template>
              <el-dialog
                v-model="dialogVisible"
```

```
            :title="editingServiceId ? 'Редагувати послугу' : 'Додати послугу'"
            width="450px"
            :close-on-click-modal="false"
            :close-on-press-escape="true"
            @close="handleClose"
            class="create-form-modal-dialog"
          >
            <template #header>
              <div class="create-form-modal-header">
                <h2 class="create-form-modal-title">
                  <div class="icon">
                    <el-icon><Setting /></el-icon>
                  </div>
                  {{ editingServiceId ? 'Редагувати послугу' : 'Додати послугу' }}
                </h2>
                <p class="create-form-modal-description">
                  {{ editingServiceId ? 'Редагуйте інформацію про послугу.' : 'Додайте
нову послугу до списку.' }}
                </p>
              </div>
            </template>

            <el-form
              ref="formRef"
              :model="formData"
              :rules="formRules"
              label-position="top"
              @submit.prevent="handleSubmit"
            >
              <div class="form-row">
                <el-form-item label="Назва послуги" prop="name">
                  <el-input
                    v-model="formData.name"
                    placeholder="Наприклад: Заміна масла"
                    maxlength="255"
                    show-word-limit
                    clearable
                  />
                </el-form-item>
              </div>

              <div class="form-row">
                <el-form-item label="Категорія" prop="category_id">
                  <el-select
                    v-model="formData.category_id"
                    placeholder="Оберіть категорію"
                    clearable
```

```
                    style="width: 100%"
                  >
                    <el-option
                      v-for="cat in categories"
                      :key="cat.id"
                      :label="cat.name"
                      :value="cat.id"
                    />
                  </el-select>
                </el-form-item>
              </div>

              <div class="form-row">
                <el-form-item label="Базова вартість (грн)" prop="base_price">
                  <el-input-number
                    v-model="formData.base_price"
                    :min="0"
                    :step="50"
                    :precision="2"
                    placeholder="0.00"
                    style="width: 100%"
                  />
                </el-form-item>
              </div>
            </el-form>

            <template #footer>
              <div class="create-form-modal-footer">
                <el-button @click="handleClose">Скасувати</el-button>
                <el-button type="primary" :loading="saving" @click="handleSubmit">
                  {{ editingServiceId ? 'Зберегти' : 'Додати' }}
                </el-button>
              </div>
            </template>
          </el-dialog>
        </template>
```

А.11 Фрагмент файлу Reports.vue

```
<template>
  <DashboardLayout>
    <div class="reports-container">
      <div class="reports-header">
```

```
                <h1 class="title">Звіти</h1>
                <div class="year-selector" v-if="revenueData">
                  <label>Рік:</label>
                  <el-select v-model="selectedYear" @change="fetchRevenueData"
class="year-select">
                    <el-option
                      v-for="y in yearOptions"
                      :key="y"
                      :label="y"
                      :value="y"
                    />
                  </el-select>
                </div>
              </div>

              <div class="charts-grid">
                <div class="chart-card">
                  <div class="chart-header">
                    <h2>Виручка за місяцями</h2>
                    <el-button type="primary" size="small" :loading="exportingRevenue"
@click="downloadRevenueExcel">
                      <el-icon><Download /></el-icon>
                      Excel
                    </el-button>
                  </div>
                  <div class="chart-body">
                    <Bar v-if="revenueData" :data="revenueChartData"
:options="revenueChartOptions" />
                    <div v-else class="chart-loading">
                      <el-icon class="is-loading"><Loading /></el-icon>
                      <span>Завантаження...</span>
                    </div>
                  </div>
                </div>

                <div class="chart-card">
                  <div class="chart-header">
                    <h2>Популярні послуги</h2>
                    <el-button type="primary" size="small" :loading="exportingPopular"
@click="downloadPopularExcel">
                      <el-icon><Download /></el-icon>
                      Excel
                    </el-button>
                  </div>
                  <div class="chart-body">
```

```
                        <Bar v-if="popularData" :data="popularChartData"
:options="popularChartOptions" />
                        <div v-else class="chart-loading">
                          <el-icon class="is-loading"><Loading /></el-icon>
                          <span>Завантаження...</span>
                        </div>
                      </div>
                    </div>

                    <div class="chart-card">
                      <div class="chart-header">
                        <h2>Записи за статусами</h2>
                        <el-button type="primary" size="small" :loading="exportingStatus"
@click="downloadStatusExcel">
                          <el-icon><Download /></el-icon>
                          Excel
                        </el-button>
                      </div>
                      <div class="chart-body">
                        <Doughnut v-if="statusData" :data="statusChartData"
:options="statusChartOptions" />
                        <div v-else class="chart-loading">
                          <el-icon class="is-loading"><Loading /></el-icon>
                          <span>Завантаження...</span>
                        </div>
                      </div>
                    </div>
                  </div>
                </DashboardLayout>
              </template>
```

А.12 Фрагмент файлу MasterDashboard.vue

```
              <template>
                <div class="technician-dashboard" style="font-family: 'Inter', sans-serif">
                  <!-- Sidebar -->
                  <aside class="technician-sidebar">
                    <div class="sidebar-top">
                      <div class="sidebar-logo">
                        <div class="logo-icon-wrap">
                          <el-icon :size="20"><Tools /></el-icon>
```

```
            </div>
            <div>
              <h2 class="logo-title">AutoCare</h2>
              <p class="logo-subtitle">Панель майстра</p>
            </div>
          </div>
        </div>

        <nav class="sidebar-nav">
          <router-link
            to="/dashboard/master"
            class="nav-link nav-link--active"
          >
            <el-icon class="nav-icon"><List /></el-icon>
            <span>Мої замовлення</span>
          </router-link>
        </nav>

        <div class="sidebar-footer">
          <div class="user-block">
            <div class="user-avatar">{{ userInitials }}</div>
            <div>
              <p class="user-name">{{ userDisplayName }}</p>
              <p class="user-role">Майстер</p>
            </div>
          </div>
          <button
            type="button"
            class="logout-btn"
            @click="handleLogout"
          >
            <el-icon class="logout-icon"><ArrowLeftBold /></el-icon>
            <span>Вийти</span>
          </button>
        </div>
      </aside>

      <!-- Main Content -->
      <main class="technician-main">
        <div class="main-header">
          <h1 class="main-title">Мої замовлення</h1>
          <p class="main-date">{{ dateLabel }}</p>
        </div>

        <!-- Toolbar -->
```

```
        <div class="toolbar">
          <div class="date-picker-wrap" v-click-outside="closeCalendar">
            <div class="date-trigger" @click="showCalendar = !showCalendar">
              <el-icon class="toolbar-icon"><Calendar /></el-icon>
              <span>{{ dateValue }}</span>
            </div>
            <div v-if="showCalendar" class="calendar-popover">
            <el-date-picker
              v-model="filterDate"
              type="date"
              :placeholder="'Оберіть дату'"
              format="DD.MM.YYYY"
              value-format="YYYY-MM-DD"
              class="calendar-inline"
              @change="onDateChange"
            />
            </div>
          </div>

          <div class="select-wrap">
            <select v-model="filterStatus" class="status-select"
@change="fetchBookings">
              <option value="">Всі статуси</option>
              <option v-for="s in statuses" :key="s.id" :value="s.id">{{ s.name
}}</option>
            </select>
          </div>
        </div>

        <!-- Skeleton -->
        <div v-if="loading" class="orders-list">
          <div v-for="i in 3" :key="i" class="order-card order-card--skeleton">
            <div class="skeleton-row">
              <div class="skeleton skeleton-w40" />
              <div class="skeleton skeleton-w24" />
            </div>
            <div class="skeleton skeleton-w56" />
            <div class="skeleton skeleton-w48" />
            <div class="skeleton-row tags">
              <div class="skeleton skeleton-w20" />
              <div class="skeleton skeleton-w24" />
            </div>
            <div class="skeleton skeleton-w24 align-end" />
          </div>
        </div>
```

```
                <!-- Empty -->
                <div v-else-if="filteredBookings.length === 0" class="empty-state">
                  <div class="empty-circle">
                    <el-icon :size="40" class="empty-icon"><List /></el-icon>
                  </div>
                  <h3 class="empty-title">Замовлень немає</h3>
                  <p class="empty-text">
                    На обрану дату немає призначених замовлень. Оберіть іншу дату або
звернiться до менеджера.
                  </p>
                </div>

                <!-- Orders list -->
                <div v-else class="orders-list">
                  <div
                    v-for="booking in filteredBookings"
                    :key="booking.id"
                    class="order-card"
                  >
                    <div class="card-top">
                      <div class="card-datetime">
                        <el-icon class="card-icon"><Calendar /></el-icon>
                        <span>{{ formatBookingDateTime(booking) }}</span>
                      </div>
                      <div class="card-actions">
                        <span :class="['status-badge', statusClass(booking.status_id)]">
                          {{ booking.status_name }}
                        </span>
                        <button
                          v-if="isConfirmed(booking)"
                          type="button"
                          class="action-btn action-btn--primary"
                          :disabled="updatingId === booking.id"
                          @click="startWork(booking)"
                        >
                          {{ updatingId === booking.id ? '...' : 'Взяти в роботу' }}
                        </button>
                        <button
                          v-if="isInProgress(booking)"
                          type="button"
                          class="action-btn action-btn--complete"
                          :disabled="updatingId === booking.id"
                          @click="complete(booking)"
                        >
```

```
                        {{ updatingId === booking.id ? '...' : 'Виконано' }}
                      </button>
                    </div>
                  </div>

                  <div class="card-body">
                    <div class="info-row">
                      <div class="info-item">
                        <el-icon class="info-icon"><User /></el-icon>
                        <span>{{ booking.client?.full_name }}</span>
                      </div>
                      <div v-if="booking.client?.phone" class="info-item info-item--
muted">
                        <el-icon class="info-icon"><Phone /></el-icon>
                        <span>{{ booking.client.phone }}</span>
                      </div>
                    </div>
                    <div class="info-item car-row">
                      <el-icon class="info-icon"><Van /></el-icon>
                      <span>{{ booking.car?.full_name }}</span>
                      <span class="separator">•</span>
                      <span class="plate-badge">{{ booking.car?.license_plate }}</span>
                    </div>
                    <div class="services-row">
                      <span
                        v-for="service in booking.services"
                        :key="service.id"
                        class="service-tag"
                      >
                        {{ service.name }}
                      </span>
                    </div>
                    <div v-if="booking.description" class="comment-row">
                      <el-icon class="comment-icon"><ChatDotRound /></el-icon>
                      <span>{{ booking.description }}</span>
                    </div>
                  </div>

                  <div class="card-footer">
                    <span class="price">{{ formatPrice(booking.total_price) }} ₴</span>
                  </div>
                </div>
              </div>
            </main>
          </div>
        </template>
```

ДОДАТОК Б

Програмна реалізація Back-End частини

Для серверної частини були написані контроллери та роутери для доступу клієнстької частини до серверу.

Б.1 Файл AuthController.php (login() та register() методи)

```
    public function register(RegisterRequest $request): JsonResponse
    {
        // If a client with this phone was already created via the public booking form — link to them
        $client = Client::where('phone', $request->phone)->first();

        if ($client) {
            // Replace the system-generated placeholder email if present
            if ($client->email && str_contains($client->email, '@booking.local')) {
                $client->update([
                    'full_name' => $request->name,
                    'email' => $request->email,
                ]);
            }
        } else {
            $client = Client::create([
                'full_name' => $request->name,
                'email' => $request->email,
                'phone' => $request->phone,
            ]);
        }

        $user = User::create([
            'full_name' => $request->name,
            'email' => $request->email,
            'phone' => $request->phone,
            'password' => Hash::make($request->password),
            'role_id' => User::ROLE_CLIENT,
            'client_id' => $client->id,
        ]);
```

```
        $token = $user->createToken('auth-token')->plainTextToken;

        return response()->json([
            'message' => 'Користувача успішно зареєстровано',
            'user' => $user,
            'token' => $token,
        ], 201);
    }

    public function login(LoginRequest $request): JsonResponse
    {
        if (!Auth::attempt($request->only('email', 'password'))) {
            return response()->json([
                'message' => 'Невірний email або пароль',
            ], 401);
        }

        $user = Auth::user();
        $token = $user->createToken('auth-token')->plainTextToken;

        return response()->json([
            'message' => 'Успішний вхід',
            'user' => $user,
            'token' => $token,
        ]);
    }
```

Б.2 Файл CarsController.php (createCar(), updateCar())

```
    public function createCar(int $clientId, array $data): ClientCar
    {
        $data['client_id'] = $clientId;
        $car = ClientCar::create([
            'client_id' => $clientId,
            'car_model_id' => $data['model'],
            'car_year' => $data['year'],
            'mileage' => $data['mileage'],
            'vin' => $data['vin'],
            'license_plate' => $data['licence_plate'],
            'engine_type_id' => $data['engine_type'],
            'gearbox_type_id' => $data['gearbox_type'],
```

```
            'drive_unit_type_id' => $data['drive_unit_type'],
            'fuel_type_id' => $data['fuel_type'],
        ]);

        $car->load('carModel.brand');

        return $car;
    }

    public function updateCar(int $carId, array $data): ClientCar
    {
        $car = ClientCar::find($carId);
        $car->update([
            'car_model_id' => $data['model'],
            'car_year' => $data['year'],
            'mileage' => $data['mileage'],
            'vin' => $data['vin'],
            'license_plate' => $data['licence_plate'],
            'engine_type_id' => $data['engine_type'],
            'gearbox_type_id' => $data['gearbox_type'],
            'drive_unit_type_id' => $data['drive_unit_type'],
            'fuel_type_id' => $data['fuel_type'],
        ]);

        $car->load('carModel.brand');

        return $car;
    }
```

Б.3 Фрагмент файлу ClientController.php

```
    public function index(Request $request): JsonResponse
    {
        $query = Client::query()
            ->with('user')
            ->withCount(['cars', 'bookings']);

        $search = $request->input('search');
        if (!empty($search)) {
            $search = '%' . trim($search) . '%';
            $query->where(function ($q) use ($search) {
                $q->where('clients.full_name', 'like', $search)
```

```
                ->orWhere('clients.email', 'like', $search)
                ->orWhere('clients.phone', 'like', $search)
                ->orWhereHas('cars', function ($carQuery) use ($search) {
                    $carQuery->where('license_plate', 'like', $search)
                        ->orWhere('vin', 'like', $search);
                });
            });
        }

        $clients = $query->orderBy('full_name')->get();

        return response()->json($clients);
    }

    public function store(Request $request): JsonResponse
    {
        $validated = $request->validate([
            'full_name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'email', 'max:255', 'unique:clients,email'],
            'phone' => ['nullable', 'string', 'max:255'],
        ]);

        $client = Client::create($validated);
        $client->load('user')->loadCount(['cars', 'bookings']);

        return response()->json([
            'message' => 'Клієнта створено',
            'client' => $client,
        ], 201);
    }

    public function update(Request $request, Client $client): JsonResponse
    {
        $validated = $request->validate([
            'full_name'      => 'sometimes|string|max:255',
            'email'          => 'sometimes|email|max:255|unique:clients,email,' . $client->id,
            'phone'          => 'sometimes|nullable|string|max:255',
            'manager_notes'  => 'sometimes|nullable|string|max:5000',
        ]);

        $client->update($validated);
        $client->load([
            'user',
            'cars.carModel.brand',
            'bookings.services',
```

```
            'bookings.status',
            'bookings.car',
        ]);
        $client->loadCount(['cars', 'bookings']);

        return response()->json($client);
    }
```

Б.4 Фрагмент файлу ServiceController.php

```
    public function index(): JsonResponse
    {
        $services = Service::with('category')->orderBy('name')->get();

        return response()->json($services);
    }

    public function store(Request $request): JsonResponse
    {
        $validated = $request->validate([
            'name' => ['required', 'string', 'max:255'],
            'base_price' => ['required', 'numeric', 'min:0'],
            'category_id' => ['nullable', 'integer', 'exists:service_categories,id'],
        ]);

        $service = Service::create($validated);
        $service->load('category');

        return response()->json([
            'message' => 'Послугу створено',
            'service' => $service,
        ], 201);
    }

    public function update(Request $request, Service $service): JsonResponse
    {
        $validated = $request->validate([
            'name' => ['required', 'string', 'max:255'],
            'base_price' => ['required', 'numeric', 'min:0'],
            'category_id' => ['nullable', 'integer', 'exists:service_categories,id'],
        ]);
```

```
        $service->update($validated);
        $service->load('category');

        return response()->json([
            'message' => 'Послугу оновлено',
            'service' => $service,
        ]);
    }

    public function destroy(Service $service): JsonResponse
    {
        $service->delete();

        return response()->json(['message' => 'Послугу видалено']);
    }
```

Б.5 Фрагмент файлу ReportController.php

```
    public function revenueByMonth(Request $request): JsonResponse
    {
        $year = $request->input('year', now()->year);
        $months = $request->input('months', 12);

        $data = PaymentTransaction::query()
            ->where('payment_status', PaymentTransaction::STATUS_COMPLETED)
            ->whereYear('created_at', $year)
            ->selectRaw('MONTH(created_at) as month, SUM(amount) as total')
            ->groupBy('month')
            ->orderBy('month')
            ->get()
            ->keyBy('month');

        $labels = [];
        $values = [];
        $monthNames = [
            1 => 'Січень', 2 => 'Лютий', 3 => 'Березень', 4 => 'Квітень',
            5 => 'Травень', 6 => 'Червень', 7 => 'Липень', 8 => 'Серпень',
            9 => 'Вересень', 10 => 'Жовтень', 11 => 'Листопад', 12 => 'Грудень'
        ];

        for ($m = 1; $m <= 12; $m++) {
            $labels[] = $monthNames[$m] . ' ' . $year;
```

```
            $values[] = (float) ($data->get($m)?->total ?? 0);
        }

        return response()->json([
            'labels' => $labels,
            'values' => $values,
            'year' => (int) $year,
        ]);
    }

    public function exportRevenueExcel(Request $request): BinaryFileResponse
    {
        $year = $request->input('year', now()->year);

        return Excel::download(
            new RevenueByMonthExport($year),
            "vuruchka-{$year}.xlsx",
            \Maatwebsite\Excel\Excel::XLSX
        );
    }

    public function exportPopularServicesExcel(): BinaryFileResponse
    {
        return Excel::download(
            new PopularServicesExport(),
            'populyarni-posluhy.xlsx',
            \Maatwebsite\Excel\Excel::XLSX
        );
    }

    public function exportBookingsByStatusExcel(): BinaryFileResponse
    {
        return Excel::download(
            new BookingsByStatusExport(),
            'zapysy-za-statusamy.xlsx',
            \Maatwebsite\Excel\Excel::XLSX
        );
    }
```