НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО **Декан факультету**	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ** **Завідувач кафедри інформаційних систем і технологій**
_____ **Ігор БОЛБОТ** (підпис)	_____ **Михайло ШВИДЕНКО** (підпис)
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка системи бронювання автомобілів у сервісі прокату**»**

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Гарант освітньої програми

д.е.н., професор _____ **Роман РУДЕНСЬКИЙ**

(підпис)

Керівник бакалаврської кваліфікаційної роботи _____ **Роман ЗОЛОТУХА**

д.ф. з к.н. (підпис)

Виконав _____ **Віктор БІЛОНОЖКО**

(підпис)

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри інформаційних систем і технологій

к.е.н., доцент _____ Михайло ШВИДЕНКО
(підпис)

«___» _____ 2026 р.

ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ
Білоножку Віктору Станіславовичу

Спеціальність 122 – «Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Тема бакалаврської кваліфікаційної роботи «Розробка системи бронювання автомобілів у сервісі прокату» затверджена наказом від «06» січня 2026 р.№46 «С»

Термін подання завершеної роботи на кафедру «22» травня 2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту): надання інформації про доступні для оренди автомобілі, дати бронювання.

Перелік питань, що підлягають дослідженню:

- Аналіз проблемної області
- Моделювання предметної області
- Проєктування програмної системи
- Впровадження та експлуатація системи

Дата видачі завдання «06» січня 2026 р.

Керівник бакалаврської кваліфікаційної роботи _____ (підпис) **Роман ЗОЛОТУХА**

Завдання взяв до виконання _____ (підпис) **Віктор БІЛОНОЖКО**

ЗМІСТ

ВСТУП

Швидкий розвиток інформаційних технологій та зростання цифрової економіки суттєво трансформували сферу послуг, зокрема індустрію прокату автомобілів. Сьогодні клієнти очікують швидких, зручних та безпечних онлайн-сервісів, які дозволяють їм знаходити, порівнювати та бронювати транспортні засоби дистанційно без зайвої паперової роботи. Водночас постачальникам послуг прокату потрібні ефективні інструменти для управління списками транспортних засобів, обробки запитів на бронювання та спілкування з клієнтами. Тому розробка веб-системи бронювання автомобілів є актуальним та своєчасним завданням, яке сприяє цифровізації та автоматизації послуг прокату.

Актуальність дослідження обумовлена зростаючим попитом на платформи онлайн-бронювання, необхідністю оптимізації бізнес-процесів у компаніях з прокату та важливістю покращення взаємодії з користувачами завдяки автоматизації та взаємодії в режимі реального часу. Впровадження спеціалізованого веб-додатку дозволяє зменшити ручні операції, мінімізувати помилки, підвищити прозорість процедур бронювання та забезпечити ефективну комунікацію між постачальниками послуг прокату та орендарями.

Метою бакалаврської роботи є розробка та впровадження веб-системи бронювання автомобілів, що забезпечує автоматизацію процесів публікації оголошень про прокат, пошуку транспортних засобів, подання заявок на оренду та комунікації між орендарями й орендодавцями.

Об'єктом дослідження є процес організації та управління послугами прокату автомобілів з використанням сучасних веб-технологій.

Предметом дослідження є методи, моделі та програмні засоби, що використовуються для проєктування та впровадження веб-системи оренди автомобілів на базі технологій PHP, Symfony та MySQL.

Для досягнення поставленої мети визначено такі **завдання**:

1. проаналізувати поточний стан онлайн-сервісів прокату автомобілів та визначити їх основні особливості та обмеження;
2. визначити функціональні та нефункціональні вимоги до системи;
3. спроєктувати архітектуру системи та структуру бази даних;
4. розробити веб-додаток з використанням PHP, Symfony та MySQL;
5. реалізувати ключові функціональні можливості, такі як реєстрація та авторизація користувачів, розміщення оголошень про прокат, пошук автомобілів за критеріями, подання заявок на прокат та спілкування в режимі реального часу через чат.

Тестування програмного застосунку проводилося шляхом функціонального та інтеграційного тестування для перевірки відповідності заданим вимогам.

Результати бакалаврської роботи апробовано у тезах доповіді на V Всеукраїнській науково-практичній конференції «Сучасні інформаційні технології та програмне забезпечення», що відбулася у 2026 році.

Структура записки: бакалаврська робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. Загальний обсяг роботи становить 76 сторінок. Робота містить 22 рисунки, 2 таблиці, 35 джерел та 2 додатки.

У першому розділі проаналізовано предметну область та існуючі рішення на ринку онлайн-сервісів прокату автомобілів, сформульовано постановку завдання, а також визначено функціональні й нефункціональні вимоги та вимоги до інтерфейсу користувача. У другому розділі проведено об’єктне та функціональне моделювання предметної області з використанням інструментарію UML (діаграми прецедентів, послідовності, активності, класів) та визначено ключові абстракції системи. У третьому розділі описано проєктування програмної системи, що включає розробку логічної моделі даних, вибір та реалізацію системи управління базами даних (СУБД), побудову загальної архітектури й організаційної структури програмного забезпечення (діаграма пакетів), а також обґрунтовано вибір інструментарію для розробки.

Четвертий розділ присвячено впровадженню та експлуатації системи, зокрема визначено вимоги до апаратного і програмного забезпечення, описано процес тестування розробленого веб-застосунку та наведено детальний склад інсталяційного пакету. У висновках підсумовано результати дослідження та окреслено можливі напрямки подальшого розвитку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Ринок прокату автомобілів зазнав значних трансформацій за останнє десятиліття завдяки швидкому розвитку цифрових технологій та широкому використанню Інтернету. Традиційні моделі прокату автомобілів, які значною мірою спиралися на офлайн-спілкування, паперову документацію та ручну координацію між клієнтами та постачальниками послуг, поступово замінюються онлайн-платформами, що автоматизують основні бізнес-процеси. У цьому контексті розробка системи оренди автомобілів для служби прокату є логічною відповіддю на зростаючий попит на цифрові рішення, що підвищують ефективність, прозорість та доступність операцій з прокату.

Розглядається проблемна область, що включає процеси публікації пропозицій прокату, пошуку транспортних засобів, подання заявок на прокат, підтвердження бронювань та організації комунікації між постачальником прокату та орендарем. Ці процеси включають кількох учасників, кожен з яких має різні ролі та інтереси. Постачальники прокату прагнуть максимізувати використання транспортних засобів, мінімізувати час простою та забезпечити надійну взаємодію з клієнтами. Орендарі, у свою чергу, очікують зручних інструментів пошуку, чіткої інформації про транспортні засоби, прозорого ціноутворення та оперативного підтвердження запитів на оренду. Взаємодія між цими сторонами повинна підтримуватися технологічним рішенням, здатним координувати їхні дії в режимі реального часу.

Ключовим викликом є ефективне управління інформацією. Кожен орендований автомобіль характеризується численними параметрами, такими як марка, модель, рік випуску, технічні характеристики, ціна оренди, період доступності та додаткові умови. Система повинна зберігати, обробляти та відображати ці дані структуровано та послідовно. Крім того, вона повинна забезпечувати актуальність інформації про доступність та синхронізацію із

запитами на бронювання, щоб запобігти конфліктам, таким як подвійне бронювання або перекриття періодів оренди.

Ще одним важливим аспектом є автоматизація процедур бронювання. За відсутності спеціалізованої програмної системи запити на оренду часто обробляються вручну, що збільшує ймовірність людських помилок, затримок у відповідях та невідповідностей у комунікації. Впровадження веб-системи бронювання автомобілів дозволяє формалізувати механізми подання запитів, відстеження статусу та сповіщень. Це сприяє більшій надійності та відстежуваності операцій, оскільки кожен запит реєструється в базі даних і може контролюватися протягом усього його життєвого циклу.

Зв'язок між постачальниками послуг оренди та орендарями становить окремий вимір проблемної області. У багатьох існуючих сценаріях оренди зв'язок здійснюється через зовнішні канали, такі як телефонні дзвінки або сторонні програми обміну повідомленнями. Такий підхід ускладнює ведення обліку та знижує прозорість. Інтеграція функції чату безпосередньо в систему дозволяє обом сторонам обмінюватися повідомленнями в контрольованому середовищі, зберігаючи історію спілкування та пов'язуючи її з конкретним запитом на оренду. Це покращує координацію та зменшує непорозуміння, пов'язані з деталями бронювання.

З технологічної точки зору, проблемна область також включає питання безпеки та захисту даних. Оскільки система обробляє персональні дані користувачів, включаючи реєстраційні дані та історію оренди, вона повинна впроваджувати безпечні механізми автентифікації та авторизації. Доступ до функціональності має бути чітко розмежований відповідно до ролей користувачів, гарантуючи, що лише авторизовані користувачі можуть створювати оголошення, подавати запити або отримувати доступ до конфіденційної інформації. Крім того, система повинна захищати від поширених веб-вразливостей, таких як несанкціонований доступ, маніпулювання даними та атаки ін'єкцій.

Масштабованість та продуктивність є додатковими міркуваннями в проблемній області. Зі збільшенням кількості користувачів та оголошень про оренду система повинна підтримувати стабільну продуктивність та швидкість реагування. Це вимагає добре продуманої архітектури, оптимізованих запитів до бази даних та ефективної обробки одночасних запитів. Вибір сучасного веб-фреймворку та реляційної системи управління базами даних підтримує структуровану розробку такого рішення.

Аналіз існуючих онлайн-платформ прокату автомобілів показує, що хоча багато систем пропонують базові функції бронювання, не всі вони забезпечують гнучкі механізми фільтрації, інтегровані інструменти комунікації або чіткий розподіл ролей між постачальниками та орендарями. Деякі рішення орієнтовані переважно на великі компанії з прокату та можуть не підходити для малих або незалежних постачальників. Тому існує потреба в системі, яка поєднує зручність використання, модульну архітектуру та адаптивність до різних сценаріїв прокату.

Предметом цього дослідження є організаційні, інформаційні та технологічні процеси, пов'язані з наданням послуг прокату автомобілів через веб-систему бронювання. Послуги прокату автомобілів є сегментом транспортної галузі, що зосереджений на тимчасовому наданні транспортних засобів окремим особам або організаціям за заздалегідь визначених умов. Перехід таких послуг на цифрові платформи суттєво змінив спосіб управління, координації та контролю операцій з прокату.

У традиційній моделі прокату автомобілів взаємодія між постачальником послуг прокату та клієнтом здійснюється переважно офлайн. Клієнти відвідують фізичний офіс, переглядають доступні транспортні засоби, обговорюють умови та підписують паперові угоди. Такий підхід вимагає значних адміністративних зусиль, обмежує географічну доступність та ускладнює управління великими автопарками або кількома запитами на прокат одночасно. Цифровізація послуг прокату усуває ці обмеження шляхом впровадження централізованих

інформаційних систем, які автоматизують основні процеси та забезпечують віддалений доступ до послуг.

У предметній області можна виділити дві основні категорії користувачів: постачальники послуг прокату та орендарі. Постачальники послуг прокату відповідають за створення та ведення списків транспортних засобів, уточнення умов прокату, встановлення цін та управління запитами на бронювання. Їхньою метою є забезпечення високого рівня використання транспортних засобів, мінімізація періодів простою та контроль над операціями з прокату. Орендарі, з іншого боку, шукають зручні інструменти для пошуку транспортних засобів, які відповідають їхнім потребам з точки зору ціни, наявності та технічних характеристик. Їм потрібна чітка та достовірна інформація, а також простий механізм подання заявок на оренду та зв'язку з постачальниками.

Комунікація між учасниками також є невід'ємною частиною предметної області. Ефективна координація часто вимагає уточнення умов оренди, деталей отримання та повернення або додаткових вимог. Інтеграція інструментів комунікації в систему бронювання гарантує, що всі взаємодії пов'язані з конкретними транзакціями оренди, тим самим підвищуючи прозорість та підзвітність.

Технологічний вимір предметної області включає використання сучасних фреймворків веб-розробки, систем управління базами даних та клієнтських технологій. Додаток повинен підтримувати безпечну автентифікацію, контроль доступу на основі ролей та захист від поширених веб-загроз. Крім того, значну роль відіграють міркування зручності використання, оскільки система повинна забезпечувати інтуїтивно зрозумілий інтерфейс, який спрощує взаємодію для користувачів з різним рівнем технічної підготовки.

Детальний опис класів та атрибутів предметної області наведено у таблиці 1.1.

Таблиця 1.1

Опис атрибутів класів предметної області

Клас предметної області	Атрибут	Опис
Користувач	ПІБ	Повне ім'я користувача системи
	Телефон	Контактний номер для зв'язку
	Email	Електронна пошта для авторизації та комунікації
	Роль	Визначає права доступу користувача
Оголошення (включає дані авто)	Заголовок	Коротка назва пропозиції прокату
	Опис	Текстовий опис умов та переваг
	Характеристики авто	Тип кузова та параметри двигуна
	Ціна за добу	Вартість оренди автомобіля за один день
	Період доступності	Дати, з якої і по яку автомобіль вільний для оренди
	Фотографії	Зображення, прив'язані до конкретного оголошення
Заявка на оренду	Дата початку та кінця	Вибраний клієнтом період оренди
	Повідомлення	Додатковий коментар орендаря до заявки
	Статус	Стан розгляду (очікує, схвалено, відхилено)
Повідомлення (Чат)	Текст повідомлення	Зміст повідомлення між користувачами
	Статус прочитання	Індикатор того, чи було повідомлення прочитане

1.2 Огляд існуючих рішень

Сфера онлайн-прокату автомобілів активно розвивається під впливом цифрової трансформації транспортних послуг та зростання попиту на гнучкі моделі мобільності. На сучасному ринку функціонують як міжнародні платформи-агрегатори, так і корпоративні системи окремих прокатних компаній. Аналіз існуючих рішень дозволяє визначити їх функціональні можливості, переваги та обмеження, що є необхідним для формування обґрунтованих вимог до розроблюваної системи.

Rentalcars.com — це одна з найбільших у світі онлайн-платформ для порівняння і бронювання прокату автомобілів, що дозволяє мандрівникам знаходити та замовляти транспортні засоби у партнерських компаній у багатьох країнах світу. За даними профілю компанії, сервіс працює у понад 160 країнах, співпрацює з більш ніж 900 прокатними компаніями та пропонує доступ до більш ніж 60 000 пунктів прокату, забезпечуючи глобальне покриття для мандрівників та локальних операторів ринку. Платформа доступна на багатьох мовах і підтримує мультивалютні транзакції, а також пропонує цілодобову підтримку клієнтів через телефон чи електронну пошту (Рис. 1.1) [4].

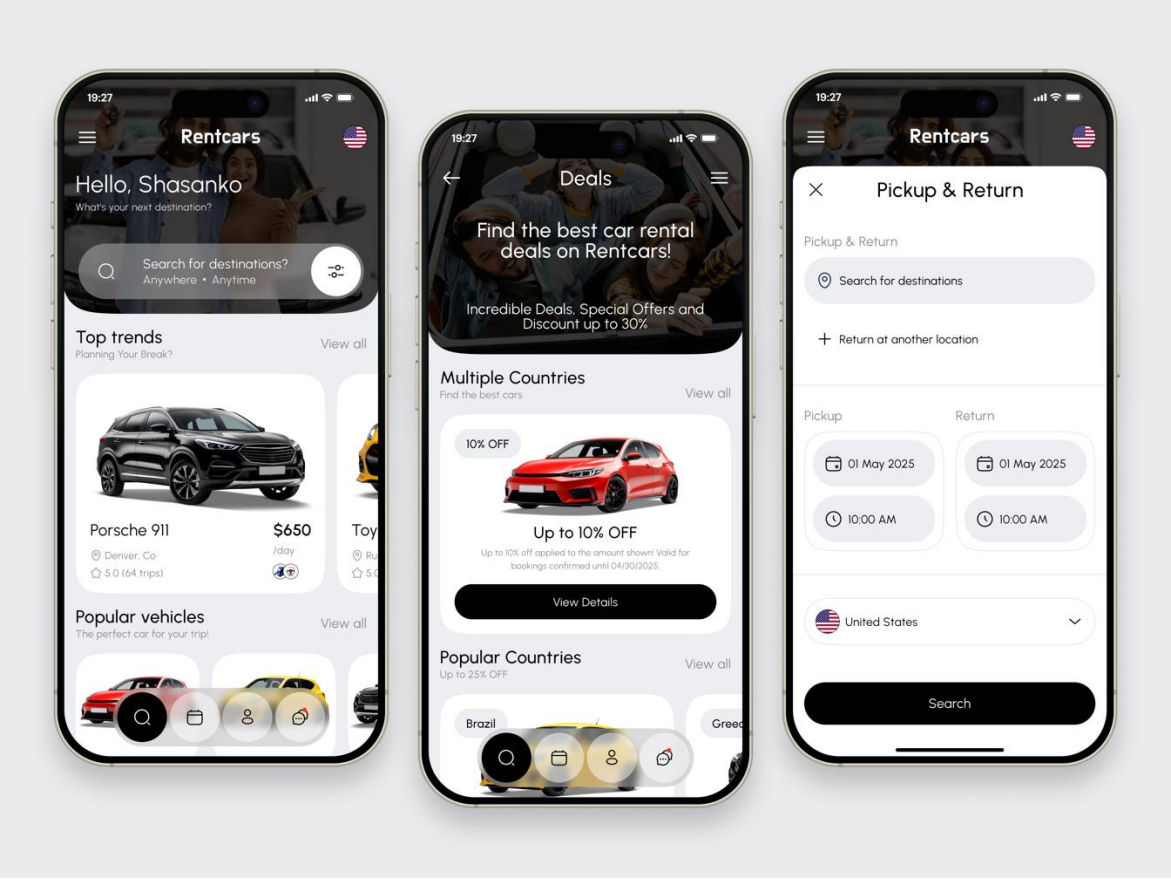

Рис. 1.1 Інформаційна система “RentalCars”

Основна функціональна мета сервісу полягає у порівнянні пропозицій різних прокатних компаній за обраними параметрами (розміщення, дати, категорія автомобіля тощо) та наданні користувачам можливості бронювання автомобіля онлайн. Це дозволяє користувачам побачити ціни від різних постачальників, вибрати найбільш відповідний варіант та забронювати автомобіль без необхідності безпосередньо звертатися до кожної компанії окремо.

Платформа також пропонує мобільний додаток, що дозволяє здійснювати пошук і бронювання автомобілів з мобільного пристрою, зберігати інформацію про бронювання та використовувати електронні ваучери під час отримання транспортного засобу без необхідності друку документів. Такий підхід покращує зручність використання для мандрівників, що часто перебувають у русі.

З технічної точки зору, Rentalcars.com інтегрується з інвентарем багатьох локальних і міжнародних постачальників прокатних послуг, обробляє великий обсяг запитів та ціноутворення в режимі реального часу. До того ж сервіс є частиною великої міжнародної групи Booking Holdings, що включає декілька інших відомих онлайн-платформ для подорожей.

Водночас у відкритих джерелах можна знайти й різні відгуки користувачів, деякі з яких вказують на проблеми з обслуговуванням, ціноутворенням або підтримкою клієнтів у окремих випадках. Такі відгуки можуть відображати індивідуальний досвід окремих користувачів і не є офіційною оцінкою сервісу, але вони показують, що залучення посередника може створювати певні нюанси під час отримання послуги.

Розглянемо інше програмне рішення на ринку.

Avis — це один із найстаріших і найвідоміших гравців на світовому ринку прокату автомобілів. Компанія була заснована в 1946 році Уорреном Авісом, який відкрив першу станцію прокату прямо в аеропорту Willow Run у Детройті (США), що стало першим у світі таким сервісом у форматі оренди авто в аеропорту. Сьогодні Avis працює більш ніж у 165 країнах світу та має тисячі

прокатних точок на всіх континентах, забезпечуючи широкий географічний доступ до своїх послуг (Рис. 1.2) [5].

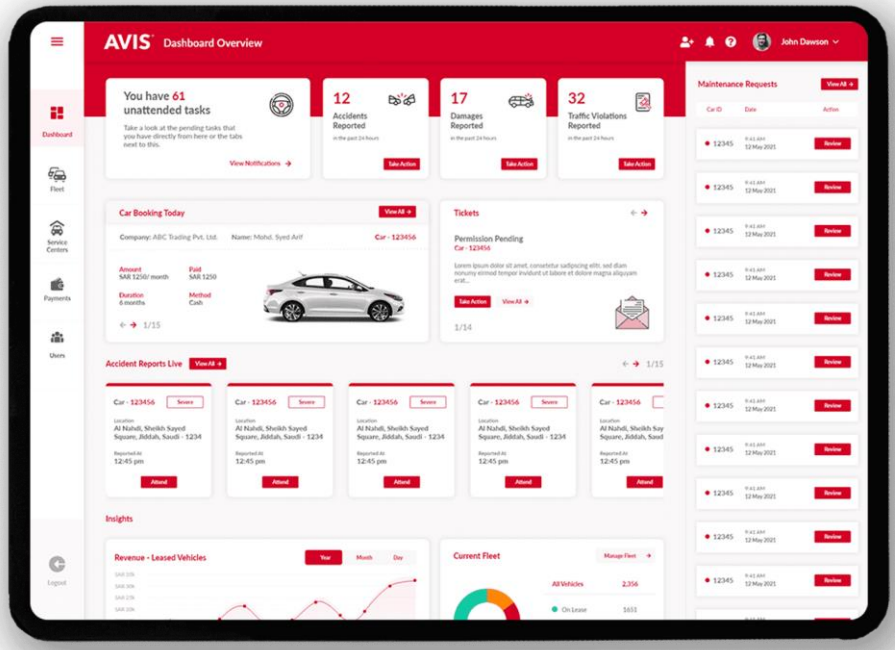

Рис. 1.2 Інформаційна система “Avis”

Компанія входить до складу Avis Budget Group, Inc., яка об’єднує кілька брендів у сфері прокату автомобілів і управління автопарком. Компанія здобула широку популярність завдяки своїй репутації надійного сервісу, інноваційним підходам до обслуговування клієнтів та сильній присутності як у корпоративному, так і в роздрібному сегменті ринку.

Avis надає послуги прокату автомобілів різних категорій — від економ- і середнього класу до преміальних моделей, позашляховиків та спеціальних транспортних засобів. Платформа дозволяє клієнтам здійснювати бронювання автомобіля онлайн через офіційний веб-сайт або мобільні додатки, обирати параметри оренди (дати, місце отримання та повернення, додаткові послуги) і отримувати підтвердження замовлення у режимі реального часу. Бізнес-модель

охоплює як короткострокову оренду для туристичних або ділових поїздок, так і довгострокові рішення для корпоративних клієнтів.

Важливою особливістю Avis є надання високого рівня сервісу та підтримки клієнтів, включно з технічним обслуговуванням транспортних засобів, страхуванням, опціями додаткового обладнання та можливістю отримати автомобіль у зручному для користувача місці. Компанія також активно розвиває цифрові сервіси, які спрощують процес бронювання та взаємодії з клієнтами, зокрема цифрову реєстрацію та мобільні інструменти управління замовленнями.

Таблиця 1.2

Порівняльний аналіз існуючих систем прокату автомобілів

Критерій	**Rentalcars.com**	**Avis**	**Власна система**
Тип системи	Онлайн-агрегатор прокатних сервісів	Корпоративна система прокатної компанії	Веб-система для автоматизації прокату автомобілів
Основні функції	Пошук і порівняння автомобілів, онлайн-бронювання, мобільний додаток	Онлайн-бронювання, управління автопарком, корпоративні сервіси	Реєстрація користувачів, публікація оголошень, пошук авто, заявки на оренду, чат
Переваги	Велика кількість партнерів, міжнародне покриття, зручне порівняння цін	Надійність бренду, високий рівень сервісу, підтримка корпоративних клієнтів	Простота використання, інтегрований чат, орієнтація на локальний ринок
Недоліки	Залежність від сторонніх постачальників, можливі проблеми з підтримкою	Висока вартість оренди, закритість системи для приватних орендодавців	Відсутність мобільного застосунку та інтеграції з платіжними системами
Що використано у власній системі	Онлайн-пошук та система бронювання	Організація структури бронювання та управління заявками	Реалізовано власний підхід до взаємодії користувачів
Відмінності	Орієнтація не на агрегування компаній, а на	Підтримка розміщення оголошень приватними	Наявність чату та спрощеної взаємодії між

власної системи	взаємодію користувачів	користувачами	орендарем і орендодавцем

У загальному підсумку, Avis представляє собою класичний приклад глобальної car rental платформи з багаторічною історією, широкою мережею операторських точок та розвиненою системою бронювання, яка може бути корисною як для приватних орендарів, так і для корпоративних клієнтів, включно з подорожами, бізнес-поїздками чи іншими транспортними потребами.

1.3 Постановка завдання

Організація системи бронювання автомобілів для служби прокату базується на необхідності забезпечення ефективної координації між постачальниками послуг прокату та орендарями, а також структурованого управління доступністю транспортних засобів, запитами на бронювання та взаємодією з користувачами. Система функціонує як централізоване цифрове середовище, в якому всі ключові процеси прокату формалізовані, реєструються та контролюються в режимі реального часу.

Основним обов'язком постачальника послуг прокату в системі є управління списками транспортних засобів та контроль операційних показників, що характеризують поточний стан бізнесу прокату. Ці показники відображають як доступність транспортних засобів, так і динаміку взаємодії з клієнтами. Зокрема, система повинна забезпечувати моніторинг наступних параметрів:

1. стан доступності кожного транспортного засобу на вибрані періоди прокату;
2. вартість прокату та додаткові умови, пов'язані з бронюванням;
3. кількість активних запитів на бронювання та підтверджених прокатів;
4. історія завершених прокатів;
5. оцінки користувачів, залишені після завершення прокату.

Програмний додаток підтримує структуровану базу даних, яка зберігає детальну інформацію про транспортні засоби, користувачів, запити на бронювання та історію зв'язку. Узгодженість та цілісність даних є критично важливими вимогами, оскільки система повинна запобігати конфліктам, таким як перекриття бронювань та несанкціоновані зміни умов прокату.

Процес бронювання організований за чітко визначеним життєвим циклом. Коли орендар вибирає транспортний засіб та вказує бажані дати оренди, генерується запит на бронювання та зберігається в базі даних. Постачальник послуг оренди отримує повідомлення про запит та переглядає запропоновані умови. Залежно від наявності та інших факторів, постачальник може підтвердити або відхилити запит. Кожна зміна статусу реєструється в системі, що забезпечує прозорість та відстеження дій.

Інтегрований механізм комунікації відіграє значну роль у конфігурації системи. Функціональність чату дозволяє обом сторонам обмінюватися повідомленнями безпосередньо в додатку, уточнювати деталі оренди та вирішувати потенційні проблеми. Вся кореспонденція пов'язана з обліковими записами користувачів і може бути пов'язана з конкретними запитами на бронювання, що покращує підзвітність та спрощує ведення обліку.

Інтерфейс користувача структурований для підтримки інтуїтивної навігації по основних функціональних модулях системи. Окремі розділи передбачені для реєстрації та автентифікації користувачів, управління списками транспортних засобів, пошуку та фільтрації транспортних засобів, подання запитів на бронювання та комунікації. Такий логічний розподіл функціональності зменшує складність та підвищує зручність використання.

З архітектурної точки зору, система розроблена за багаторівневою моделлю, яка розділяє компоненти презентації, бізнес-логіки та управління даними. Такий підхід підвищує зручність обслуговування, масштабованість та безпеку. Бекенд обробляє вхідні запити та забезпечує дотримання бізнес-правил, тоді як фронтенд надає доступний веб-інтерфейс для взаємодії з користувачем.

Завдяки цій структурованій конфігурації система бронювання автомобілів підтримує ефективне управління орендою, прозору комунікацію та надійний контроль доступності транспортних засобів. Впроваджена організаційна модель створює сприятливі умови для оптимізації операцій з оренди та покращення загальної якості послуг, що надаються користувачам.

1.4 Функціональні та нефункціональні вимоги

Для забезпечення ефективної роботи системи бронювання автомобілів для прокату було визначено набір функціональних та нефункціональних вимог. Ці вимоги слугують основою для проєктування, впровадження та тестування системи. Вони відображають потреби як постачальників послуг прокату, так і орендарів, водночас гарантуючи, що система залишається надійною, безпечною та зручною для користувача.

Функціональні вимоги:

1. забезпечити реєстрацію та авторизацію користувачів з різними ролями (орендодавець та орендар);
2. дозволити орендодавцям створювати, редагувати та видаляти оголошення про автомобілі, включаючи інформацію про марку, модель, рік випуску, технічні характеристики, ціну та фотографії;
3. забезпечити можливість пошуку автомобілів орендарями за різними критеріями, такими як ціна, дата доступності, клас авто та інші параметри;
4. реалізувати функціонал подання та обробки заявок на оренду з повідомленням про статус (підтверджено, відхилено, очікує);
5. інтегрувати внутрішню систему обміну повідомленнями (чат) між орендодавцями та орендарями для уточнення умов оренди;
6. забезпечити відображення історії бронювань та відгуків користувачів для покращення прозорості та довіри;

7. реалізувати механізм сповіщень про нові заявки, зміну статусу або зміну доступності автомобілів;
8. забезпечити ведення журналу дій користувачів для можливості контролю;
9. надати інтерфейс для управління профілем користувача та налаштування персональної інформації.

Нефункціональні вимоги:

1. забезпечити безпеку даних користувачів шляхом використання надійної аутентифікації, шифрування паролів та захисту від несанкціонованого доступу;
2. гарантувати стабільну продуктивність системи при великій кількості користувачів та одночасних запитах;
3. забезпечити масштабованість архітектури для підтримки зростання бази користувачів та автопарку;
4. надати високу доступність сервісу, з мінімальними простоями та відмовами;
5. забезпечити зручність користування інтерфейсом завдяки інтуїтивній навігації та зрозумілому дизайну;
6. підтримувати сумісність із сучасними веб-браузерами та мобільними пристроями;
7. реалізувати ефективні механізми резервного копіювання та відновлення даних для запобігання їх втраті.

1.5 Вимоги до інтерфейсу користувача

Інтерфейс користувача (UI) системи бронювання автомобілів відіграє вирішальну роль у забезпеченні ефективної взаємодії між системою та її користувачами, включаючи постачальників послуг прокату та орендарів. Добре розроблений інтерфейс повинен сприяти виконанню ключових завдань,

забезпечувати чіткість та прозорість інформації, а також мінімізувати криву навчання для користувачів з різним рівнем технічного досвіду.

Вимоги до інтерфейсу користувача базуються на принципах зручності використання, доступності та адаптивності. Система повинна забезпечувати інтуїтивно зрозуміле розташування з чітко визначеними елементами навігації та логічно згрупованими функціональними модулями. Основні розділи інтерфейсу повинні включати: реєстрацію та автентифікацію користувачів, управління списками транспортних засобів для постачальників послуг прокату, пошук та фільтрацію транспортних засобів для орендарів, подання та відстеження запитів на оренду, а також інтегровану систему обміну повідомленнями для зв'язку між користувачами.

Основні вимоги до інтерфейсу користувача включають:

- забезпечити зручну та зрозумілу реєстрацію та авторизацію користувачів;
- реалізувати логічну та інтуїтивно зрозумілу навігацію між основними функціональними модулями системи;
- надати можливість орендодавцям легко створювати, редагувати та видаляти оголошення про автомобілі через форми з полями для всіх необхідних параметрів;
- забезпечити орендарям можливість швидкого пошуку автомобілів за допомогою фільтрів та сортування за ціною, датою доступності, класом авто та іншими характеристиками;

Виконання цих вимог забезпечує високу зручність користування системою, швидкий доступ до необхідної інформації та ефективну взаємодію між усіма учасниками процесу прокату автомобілів. UI проєктується таким чином, щоб користувачі могли виконувати основні дії без додаткових пояснень або складних інструкцій, що значно підвищує ефективність роботи системи та загальне задоволення користувачів.

1.6 Висновки до першого розділу

У першому розділі було здійснено детальний аналіз предметної області розроблюваної системи бронювання автомобілів. Було визначено, що сучасний ринок прокату стрімко трансформується під впливом цифрових технологій, що зумовлює поступову заміну традиційних офлайн-моделей автоматизованими онлайн-платформами.

Проведений огляд існуючих ринкових рішень, таких як Rentalcars.com та Avis, дозволив визначити їхні основні переваги та функціональні обмеження. Зокрема, виявлено високу вартість оренди та закритість відомих корпоративних систем для приватних орендодавців, що обґрунтувало актуальність створення власного веб-застосунку з інтегрованим чатом для спрощення взаємодії на локальному ринку.

На основі проведеного дослідження було сформульовано постановку завдання й визначено ключові операційні параметри для моніторингу доступності транспортних засобів та обробки запитів. Сформовано детальний перелік функціональних та нефункціональних вимог до системи, включно з механізмами реєстрації користувачів, управління оголошеннями, пошуку за критеріями та захисту даних. Також визначено вимоги до інтерфейсу користувача, який має забезпечувати інтуїтивно зрозумілу навігацію, спрощену взаємодію та адаптивність дизайну. Сформульовані вимоги закладають теоретичний та практичний фундамент для об’єктного й функціонального моделювання системи у наступному розділі роботи.

2 МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Загальні відомості

Уніфікована мова моделювання (UML) – це стандартизована мова моделювання, яка широко використовується в програмній інженерії для візуалізації, специфікації, побудови та документування артефактів програмної системи. У контексті розробки системи бронювання автомобілів для служби прокату, UML забезпечує структурований підхід до представлення архітектури, взаємодій та процесів системи, дозволяючи розробникам, зацікавленим сторонам та користувачам краще зрозуміти дизайн та функціональність системи.

Діаграми UML можна загалом розділити на структурні та поведінкові типи. Структурні діаграми описують статичні аспекти системи, такі як зв'язки між сутностями, класами або компонентами, тоді як поведінкові діаграми фіксують динамічні взаємодії, робочі процеси та реакції системи на події.

Діаграми варіантів використання особливо корисні для ілюстрації функціональності системи з точки зору користувача. Вони показують учасників, таких як постачальники послуг прокату, орендарі та адміністратори, а також взаємодію між цими учасниками та системою, включаючи такі дії, як створення списків транспортних засобів, пошук автомобілів, подання запитів на оренду та спілкування через систему чату. Діаграми варіантів використання допомагають визначити межі системи та уточнити функціональні вимоги.

Діаграми класів, з іншого боку, моделюють статичну структуру системи, представляючи сутності, їх атрибути, методи та зв'язки. У системі бронювання автомобілів ключові класи включають Користувача, Транспортний засіб, Бронювання та Повідомлення. Такі зв'язки, як асоціації, агрегації та залежності, використовуються для зображення того, як ці сутності взаємодіють та пов'язані в системі.

Діаграми послідовностей фіксують динамічні взаємодії між об'єктами з часом. Наприклад, вони можуть ілюструвати процес подання орендарем запиту

на бронювання, його підтвердження системою, повідомлення постачальника та оновлення статусу бронювання. Діаграми активності забезпечують візуальне представлення робочих процесів та процесів, показуючи послідовності дій, точки прийняття рішень та паралельні операції. У контексті системи вони можуть зображати робочий процес бронювання транспортного засобу, затвердження доставки сповіщень. Діаграми станів описують життєвий цикл певних сутностей, таких як бронювання, яке може переходити між станами, такими як Очікування, Підтвердження, Відхилення та Завершення, залежно від подій, що ініціюють кожну зміну.

Використання UML під час розробки системи бронювання автомобілів забезпечує чітку комунікацію між командою розробників та зацікавленими сторонами, а також сприяє ранньому виявленню недоліків або невідповідностей у проєкті. Діаграми UML також підвищують якість документації, надаючи візуальні та текстові описи як структури, так і поведінки системи. Це полегшує посилання на проєкт під час впровадження та майбутнього обслуговування. Загалом, UML служить важливим інструментом у моделюванні предметної області, дозволяючи представляти складні реальні взаємодії в процесі оренди автомобілів у чіткій, структурованій та зрозумілій формі.

2.2 Об’єктне та функціональне моделювання

2.2.1 Діаграма прецедентів. Діаграма варіантів використання – це тип UML-діаграми, яка ілюструє функціональні вимоги до системи з точки зору її користувачів або «акторів». У контексті системи бронювання автомобілів для служби прокату, діаграми варіантів використання слугують візуальним представленням того, як різні користувачі взаємодіють із системою для досягнення певних цілей. Такий підхід дозволяє розробникам та зацікавленим

сторонам чітко розуміти очікувану поведінку системи та взаємодію між користувачами та функціональністю системи.

У такій системі основними учасниками зазвичай є постачальник послуг прокату та орендар. Постачальник послуг прокату взаємодіє із системою, створюючи та керуючи списками транспортних засобів, вказуючи такі деталі, як модель, ціна, наявність та додаткові функції, а також переглядаючи та реагуючи на запити на бронювання. Орендарі використовують систему для пошуку доступних транспортних засобів за різними критеріями, подання запитів на оренду, відстеження статусу бронювання та спілкування з постачальниками через інтегровану систему чату.

Діаграма варіантів використання підкреслює межі системи, показуючи, які дії виконуються користувачами, а які обробляються системою внутрішньо. Вона надає загальний огляд функціональних вимог без заглиблення в деталі реалізації. Це допомагає уточнити ролі, зменшити неоднозначності та забезпечити відповідність системи потребам своїх цільових користувачів.

Окрім ілюстрації взаємодій, діаграми варіантів використання можуть висвітлювати зв'язки між різними варіантами використання. Наприклад, деякі дії можуть включати або розширювати інші дії, такі як запит на бронювання, що включає повідомлення постачальнику послуг оренди, або розширювати процес підтвердження платежу. Ці зв'язки дозволяють розробникам моделювати залежності та додаткові потоки в системі, забезпечуючи більш повне розуміння поведінки користувачів та реакцій системи.

Розроблена діаграма прецедентів використання представлена на рис. 2.1.

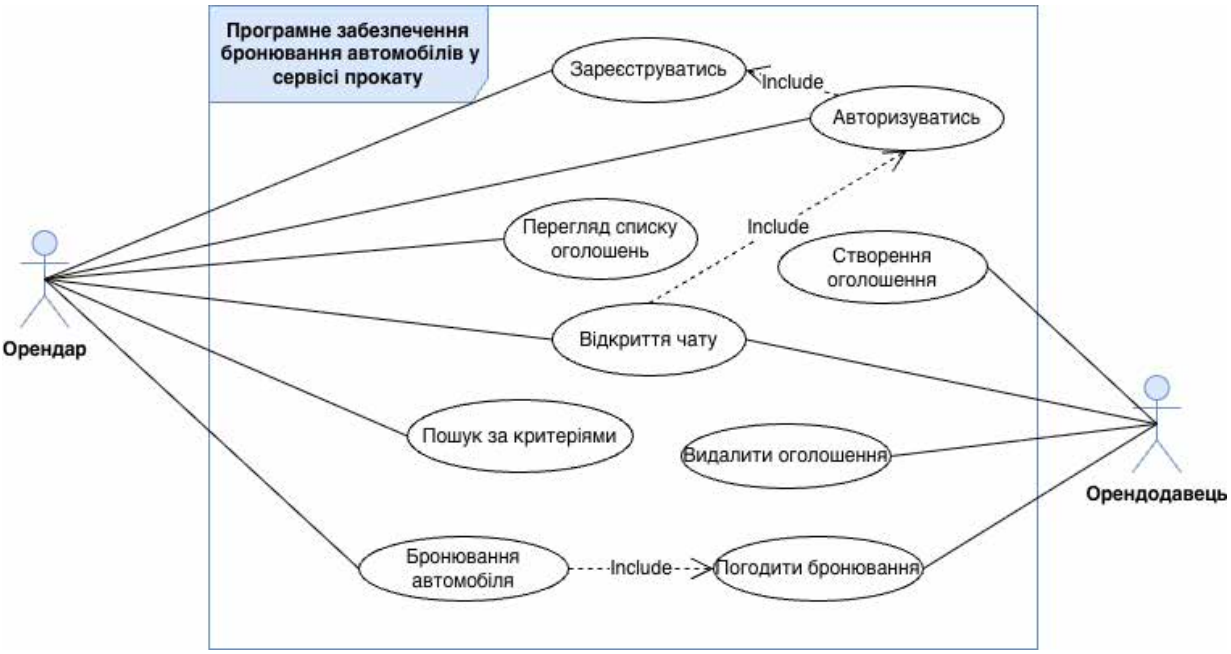

Рис. 2.1 Діаграма прецедентів

Створена діаграма прецедентів містить акторів:

- “Орендар”;
- “Орендодавець”.

Актор «Орендар» включає такі прецеденти:

- зареєструватись;
- авторизуватись;
- перегляд списку оголошень;
- відкриття чату;
- пошук за критеріями;
- бронювання автомобіля.

Актор «Орендодавець» включає такі прецеденти:

- створення оголошення;
- видалити оголошення;
- погодити бронювання.

Прецеденти певним чином залежать одне від одного.

Розглянемо детальніше вищеописані прецеденти.

Варіант використання: Перегляд списку оголошень

Актор: Орендар

Опис: Приклад використання «Перегляд списку оголошень» описує процес, за допомогою якого орендар отримує доступ та переглядає список доступних оголошень автомобілів сервісу прокату. Орендар може використовувати різні фільтри, щоб звузити результати пошуку на основі певних уподобань, таких як ціна, тип авто, двигун та інші зручності. Цей процес дозволяє орендарям переглядати та досліджувати варіанти авто, щоб знайти відповідний автомобіль.

Основний потік:

1. орендар отримує доступ до головної сторінки платформи оренди авто;
2. орендар натискає кнопку «Переглянути оголошення»;
3. система відображає список доступних оголошень про автомобіль;
4. орендар може застосовувати фільтри, такі як тип авто, ціновий діапазон, тип двигуна, витрата палива тощо;
5. система оновлює список на основі вибраних орендарем фільтрів;
6. орендар прокручує відфільтрований список оголошень;
7. орендар вибирає певне оголошення, щоб переглянути більше деталей, включаючи фотографії, опис, ціну за день та контактну інформацію орендодавця.

Альтернативні потоки:

1. якщо орендар не застосовує жодних фільтрів, система відображає всі доступні оголошення в базі даних;
2. якщо жодне оголошення не відповідає вибраним фільтрам, система відображає повідомлення типу «Немає оголошень, що відповідають вашим критеріям», і пропонує альтернативні варіанти пошуку або просить орендаря змінити свої фільтри;
3. якщо результатів забагато для відображення на одній сторінці, система може розбити результати на сторінки або реалізувати нескінченне прокручування, коли наступний набір оголошень завантажується, коли орендар прокручує вниз;

4. орендар може вибрати сортування оголошень на основі ціни, рейтингу або найновіших оголошень. Система оновлює порядок відображення відповідно до вибраних критеріїв.

Випадок використання «Перегляд списку оголошень» надає орендарям зручний інтерфейс для перегляду доступних об'єктів оренди автомобілів. Завдяки використанню фільтрів та параметрів сортування орендарі можуть швидко звузити свої варіанти та знайти об'єкти, які відповідають їхнім потребам. Цей варіант використання є важливим у процесі оренди авто, дозволяючи орендарям збирати інформацію, перш ніж приймати рішення зв'язатися з орендодавцем або подати запит на бронювання.

Розглянемо ще один сценарій використання.

Варіант використання: Підтвердження оренди

Актор: Орендодавець

Опис: Випадок використання «Підтвердження бронювання» описує процес, за допомогою якого орендодавець переглядає та підтверджує запит орендаря на оренду авто. Після того, як орендар подасть запит на бронювання, орендодавець отримує сповіщення та має можливість підтвердити або відхилити запит. Це гарантує, що обидві сторони узгоджують деталі бронювання, перш ніж продовжити транзакцію.

Основний потік:

1. орендодавець отримує сповіщення про новий запит на аренду від орендаря;
2. орендодавець входить у систему та переходить до розділу «Запити на аренду»;
3. система відображає список усіх запитів на бронювання, що очікують розгляду, включаючи такі деталі, як ім'я орендаря, дати та запитуваний автомобіль;
4. орендодавець вибирає запит на оренду для перегляду;

5. система показує детальну інформацію про оренду, включаючи контактну інформацію орендаря, запитувані дати, ціну за день та будь-які додаткові умови чи примітки, надані орендарем;
6. орендодавець переглядає деталі та вирішує, чи підтверджувати чи відхиляти оренду;
7. якщо орендодавець підтверджує оренду;
8. система оновлює статус бронювання на «Підтверджено»;
9. орендар отримує підтвердження успішного бронювання;
10. система надсилає підтвердження як орендодавцю, так і орендарю, включаючи деталі оренди, такі як дати, загальна вартість та контактну інформацію;
11. якщо орендодавець відхиляє оренду;
12. система оновлює статус бронювання на «Відхилено»;
13. орендар отримує повідомлення про відхилення з причиною (якщо вона вказана) та йому рекомендується ознайомитися з іншими доступними оголошеннями.

Альтернативні потоки:

1. якщо запит на оренду вже був підтверджений або відхилений іншим орендодавцем або після певного періоду, система повідомить орендодавця та вимкне можливість підтвердження або відхилення запиту;
2. якщо орендодавець помітить будь-яку відсутню або неповну інформацію в запиті на бронювання (наприклад, відсутні контактні дані), він може попросити орендаря надати відсутню інформацію, перш ніж продовжити підтвердження;
3. якщо запитувані дати перетинаються з існуючим бронюванням, система попередить орендодавця, і орендодавець може або відхилити бронювання, або запропонувати орендарю альтернативні дати.

Варіант використання «Підтвердження бронювання» є важливою частиною процесу оренди авто, надаючи орендодавцям контроль над своїми бронюваннями, забезпечуючи водночас ефективну комунікацію з орендарями. Дозволяючи орендодавцям підтверджувати або відхиляти запити на бронювання, система допомагає підтримувати точність інформації про наявність авто та гарантує, що обидві сторони чітко розуміють умови договору оренди. Потік сповіщень та оновлень системи сприяє безперебійній взаємодії та зміцнює довіру між орендодавцями та орендарями.

2.2.2 Діаграма послідовності. Діаграма послідовностей – це тип UML-діаграми, яка моделює динамічні взаємодії між об'єктами в системі з часом. Вона зосереджена на порядку виконання операцій та обміні повідомленнями між компонентами системи для досягнення певної функціональності. У контексті системи бронювання автомобілів для служби прокату, діаграми послідовностей є важливими для розуміння покрокових процесів, що відбуваються під час взаємодії користувачів із системою.

Діаграми послідовностей зображують об'єкти або компоненти як вертикальні лінії зв'язку, а взаємодії між ними – як горизонтальні стрілки, що представляють повідомлення або виклики методів. Ці діаграми показують, як керування проходить через систему під час певного сценарію, такого як подання запиту на оренду, підтвердження бронювання або обмін повідомленнями в системі чату. Візуалізуючи порядок взаємодій, діаграми послідовностей допомагають забезпечити правильне врахування та виконання всіх необхідних кроків.

У системі бронювання автомобілів діаграми послідовностей можуть ілюструвати такі процеси, як пошук орендарем доступного транспортного засобу, надсилання запиту на бронювання та отримання підтвердження від постачальника послуг прокату. Вони також можуть відображати, як система оновлює статус бронювання в базі даних, надсилає сповіщення залученим сторонам та записує історію транзакцій. Кожна діаграма послідовності висвітлює ролі різних об'єктів або сутностей, включаючи інтерфейс

користувача, компоненти обробки серверної частини та модулі бази даних, демонструючи, як вони співпрацюють для виконання сценарію.

Використання діаграм послідовності дозволяє розробникам виявляти потенційні проблеми в часі або логіці взаємодій, такі як затримки, конфлікти в оновленнях бронювання або відсутність сповіщень. Вони також сприяють комунікації між членами команди, забезпечуючи чітке візуальне представлення процесів, які інакше описані лише в текстових специфікаціях. Крім того, діаграми послідовності служать мостом між функціональними вимогами та реалізацією, підтримуючи розробку надійного, добре структурованого та зручного в обслуговуванні коду.

Діаграма послідовності, зображена на рис. 2.2, включає наступні об'єкти:

- “Орендар”;
- “Оголошення”;
- “Оренда авто”;
- “Чат”.

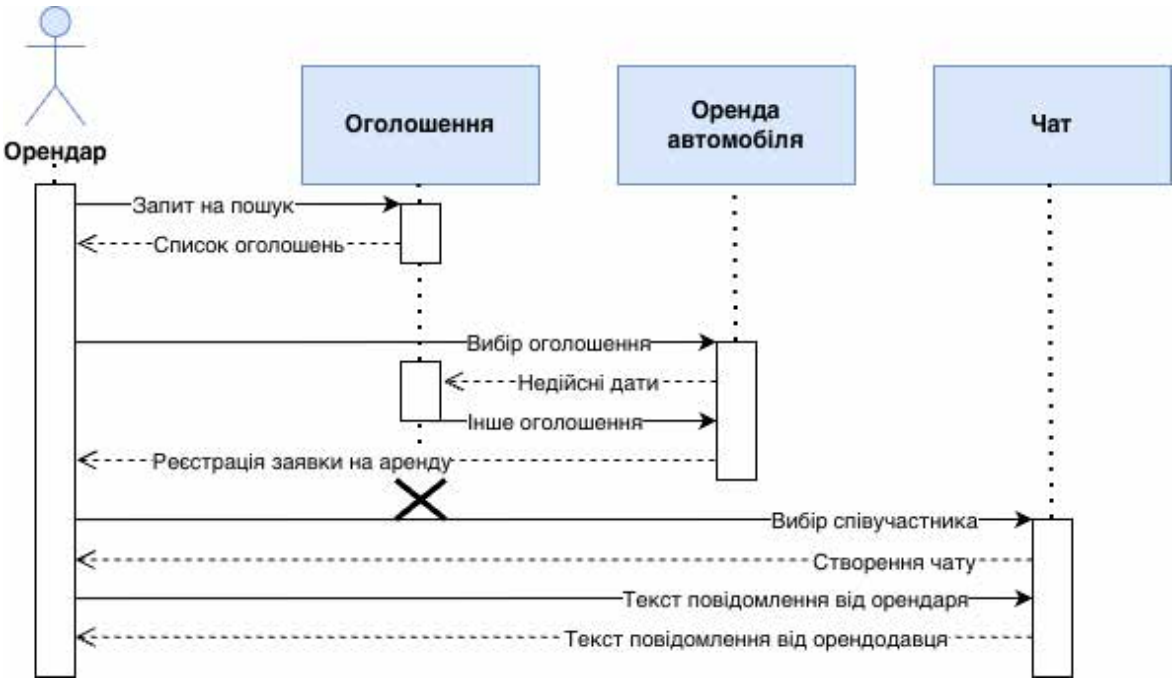

Рис. 2.2 Діаграма послідовності

Орендар розпочинає процес пошуку, звертаючись до системи оголошень, яка надає перелік доступних варіантів. Ознайомившись із

пропозиціями, він вибирає відповідне оголошення авто, після чого система перевіряє його деталі. Далі надходить заявка на бронювання, і система бронювання визначає відповідного орендодавця. Як тільки учасники процесу встановлені, між ними створюється чат, що дозволяє обговорити умови оренди та уточнити всі нюанси. Орендар надсилає повідомлення орендодавцю, а той, у свою чергу, відповідає, забезпечуючи ефективний та зрозумілий процес комунікації. Така структура дозволяє чітко уявити логіку взаємодії всіх елементів у процесі оренди.

2.2.3 Діаграма активності. Діаграма активності – це тип діаграми UML, яка представляє динамічні аспекти системи шляхом моделювання робочих процесів, процесів та послідовностей дій. На відміну від діаграм послідовностей, які зосереджені на взаємодії між об'єктами з часом, діаграми активності підкреслюють потік керування та перехід завдань від однієї дії до іншої. У контексті системи бронювання автомобілів для служби прокату, діаграми активності забезпечують чітку візуалізацію операційних процесів, дозволяючи як розробникам, так і зацікавленим сторонам зрозуміти, як виконуються та координуються різні функції.

Діаграми активності особливо корисні для зображення бізнес-процесів, що включають кілька точок прийняття рішень, паралельні дії або умовні потоки. Наприклад, процес бронювання транспортного засобу можна змоделювати як діаграму активності, яка показує такі кроки, як пошук доступних автомобілів, вибір транспортного засобу, подання запиту на оренду, перевірка запиту постачальником послуг прокату, підтвердження або відхилення бронювання та повідомлення обох сторін. Кожен з цих кроків представлений як вузол активності, тоді як рішення та розгалуження представлені за допомогою вузлів рішень, що дозволяє діаграмі фіксувати альтернативні потоки та результати.

Крім того, діаграми діяльності можуть представляти паралельні процеси, такі як оновлення наявності транспортних засобів з одночасним надсиланням сповіщень користувачам або паралельна обробка кількох запитів

на бронювання. Ця можливість допомагає розробникам передбачати потенційні проблеми із синхронізацією робочих процесів та забезпечує ефективне виконання системних процесів. Діаграми діяльності також часто включають початкові та кінцеві точки, переходи та доріжки для ілюстрації ролей або компонентів, відповідальних за кожну частину процесу, що полегшує розподіл обов'язків та визначення меж системи.

Використання діаграм діяльності під час розробки системи бронювання автомобілів надає кілька переваг. Вони підвищують ясність складних процесів, допомагають перевірити, чи враховано всі функціональні вимоги, та підтримують комунікацію між технічними та нетехнічними зацікавленими сторонами, пропонуючи інтуїтивно зрозуміле візуальне представлення робочих процесів. Діаграми діяльності також служать основою для проєктування детальної системної логіки та керівництва впровадженням, тестуванням та оптимізацією.

Розроблена діаграма активності представлена на рис. 2.3

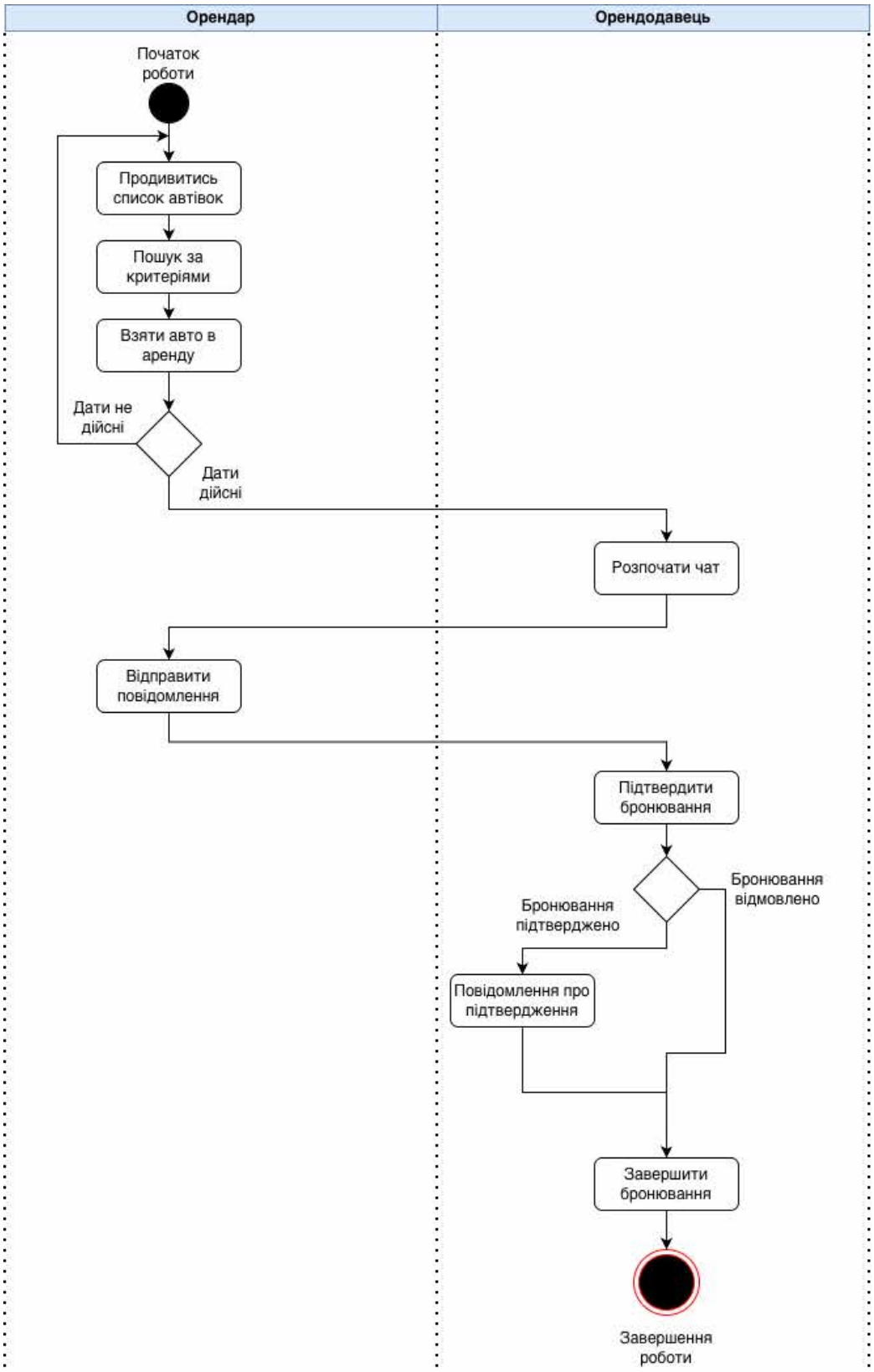

Рис. 2.3 Діаграма активності

Ця діаграма активності відображає послідовність дій орендаря при використанні системи оголошень, аренди та чату. Вона ілюструє, як орендар здійснює запит на пошук, отримує список доступних пропозицій автівок, обирає

відповідне оголошення, проходить перевірку дат, реєструє заявку на аренду, знаходить співучасника та ініціює чат для обговорення деталей. Процес завершується обміном повідомленнями між орендарем та орендодавцем, що дозволяє узгодити всі нюанси угоди. Візуалізація допомагає краще зрозуміти етапи взаємодії та забезпечує чіткий алгоритм дій.

2.3 Абстракції предметної області

Абстракції предметної області представляють собою узагальнене моделювання основних сутностей, процесів та зв’язків у системі бронювання автомобілів для обслуговування прокату. Методом такого моделювання є виділення ключових об’єктів та їх взаємодія, що дозволяє створити чітку концептуальну основу для подальшого проєктування та розробки програмного забезпечення.

Основними абстракціями предметної області є користувачі, автомобілі, заявки на оренду, оголошення про автомобілі та повідомлення. Користувачі поділені на орендодавця та орендаря. Орендодавець створює та керує оголошеннями про доступні автомобілі, надає інформацію про модель, технічні характеристики, ціну, період доступності та додаткові послуги. Орендар шукає автомобілів за певними критеріями, подає заявки на оренду та взаємодіє з орендодавцем через систему повідомлення.

Також важливо виділити абстракції для управління даними про автомобілі та користувачів, які зберігають історичні записи бронювання та оцінки, які дають змогу аналізувати активність системи та покращувати якість сервісу.

Використання таких абстракцій дозволяє розділити предметну область на логічні компоненти та встановити взаємозв’язки між ними. Це передбачає подальше проєктування баз даних, розробку функціональних модулів системи та створює основу для формування UML-діаграми, що відображають структуру такої поведінки системи. Абстракції предметної області забезпечують

концептуальну ясність і дозволяють розробникам зосередитися на ключових процесах та сутностях, без прив’язки до конкретної реалізації або технологій.

Рис. 2.4 Абстракції предметної області

Ці абстракції предметної області представляють основні сутності процесу оренди та їхні характеристики.

Абстракція оголошення охоплює ключові властивості, такі як ім'я, номер телефону, дата реєстрації. Вона відповідає за пошук і створення оголошень, а також за подання заявки на оренди.

Абстракція оренди авто, у свою чергу, містить інформацію про орендаря та орендодавця, оголошення, період оренди, суму оплати, дату та чат для комунікації. Основні функції цього компонента включають підтвердження або відхилення бронювання, а також можливість обміну повідомленнями.

Ці абстракції забезпечують структуровану модель процесу оренди, дозволяючи користувачам ефективно взаємодіяти та обмінюватися інформацією.

2.4 Діаграма класів

Діаграма класів – це тип UML-діаграми, яка представляє статичну структуру системи шляхом моделювання її ключових сутностей, їхніх атрибутів, методів та зв'язків між ними. У контексті системи бронювання автомобілів для

служби прокату, діаграма класів надає детальне уявлення про концептуальну організацію системи, показуючи, як різні об'єкти взаємодіють та пов'язані для підтримки бізнес-процесів оренди транспортних засобів.

Основна мета діаграми класів – визначити план моделі даних системи та проілюструвати зв'язки між сутностями перед впровадженням. Ключові класи в системі бронювання автомобілів зазвичай включають Користувача, Транспортний засіб, Бронювання, Оголошення та Повідомлення. Кожен клас інкапсулює атрибути, такі як ім'я користувача, характеристики транспортного засобу, дати бронювання або вміст повідомлення, а також методи, що визначають операції, які можна виконувати над цими об'єктами, такі як створення бронювання, оновлення оголошення або надсилання сповіщення.

Діаграми класів також зображують типи зв'язків між сутностями. Асоціації описують загальні зв'язки між об'єктами.

Візуалізуючи ці класи та їхні зв'язки, діаграма допомагає розробникам та дизайнерам визначати необхідні структури даних, застосовувати обмеження та розуміти поведінку системи. Вона також полегшує комунікацію між членами команди та зацікавленими сторонами, забезпечуючи спільне розуміння архітектури системи та обов'язків кожного компонента.

У системі бронювання автомобілів діаграма класів служить основою для впровадження, керуючи проєктуванням схеми бази даних, розробкою логіки серверної частини та інтеграцією системних модулів. Вона доповнює інші діаграми UML, такі як діаграми варіантів використання та послідовності, забезпечуючи статичну перспективу, яка лежить в основі динамічних процесів та взаємодії користувачів.

Для цього проєкту було створено спеціальну діаграму класів з використанням об’єктно-орієнтованого підходу (рис. 2.5).

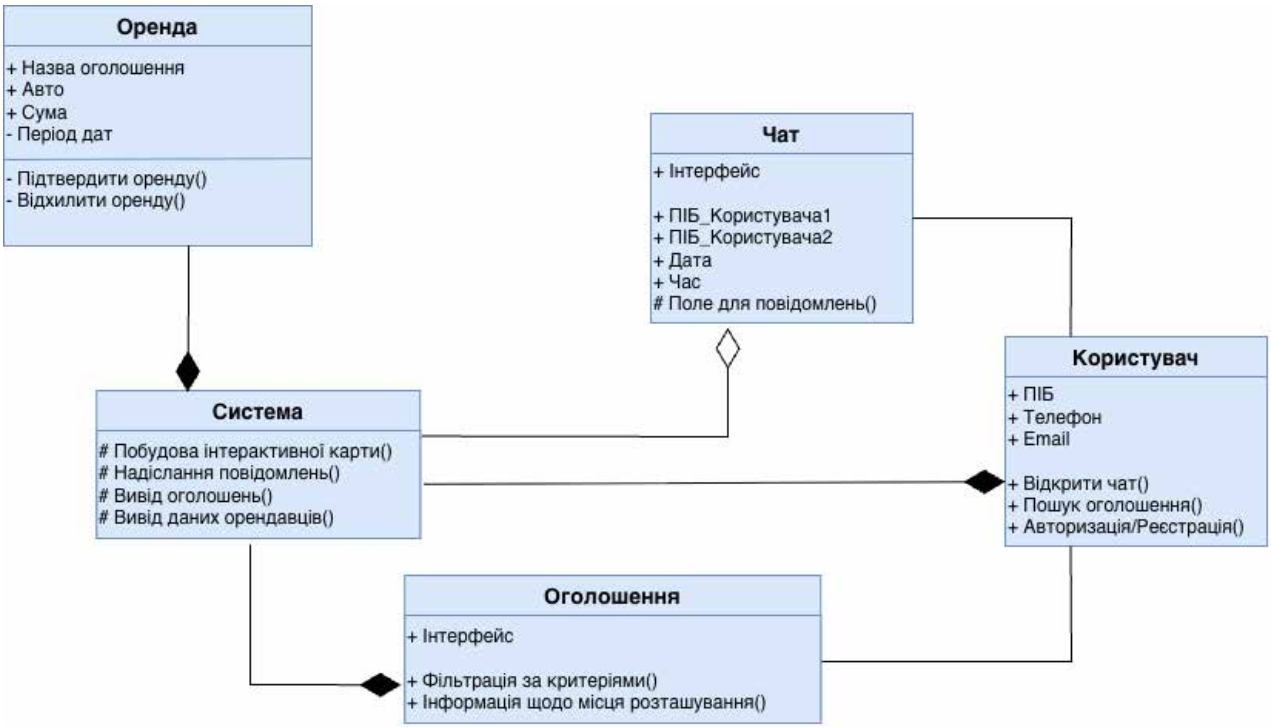

Рис. 2.5 Діаграма класів

Діаграма класів відображає структуру основних сутностей у процесі оренди, їхні характеристики та функції. Оголошення містить інформацію про орендаря, контактні дані, дату реєстрації забезпечуючи можливість створення та пошуку пропозицій, а також подання заявок на оренду. Оренда включає деталі про орендаря та орендодавця, вибране оголошення, період оренди, суму оплати, дату угоди та чат для комунікації. Воно дає змогу підтверджувати чи відхиляти заявки та організовувати обмін повідомленнями між сторонами. Така структурована модель дозволяє ефективно керувати орендним процесом, спрощуючи взаємодію між користувачами та підвищуючи зрозумілість операцій.

3 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Логічна модель даних

Діаграма «сутність-зв'язок» (ER) – це інструмент концептуального моделювання, який використовується для представлення структури бази даних, що ілюструє сутності, їх атрибути та зв'язки між ними. У контексті системи

бронювання автомобілів для служби прокату, ER-діаграма забезпечує чітке та організоване уявлення про те, як структуровані дані, як пов'язані різні об'єкти та як інформація передається системою. Це важливо для проєктування ефективної та узгодженої бази даних, яка підтримує всі функціональні вимоги.

Основні сутності в системі зазвичай включають Користувача, Транспортний засіб, Бронювання, Оголошення, Повідомлення та Відгук. Кожна сутність містить атрибути, що описують її властивості. Наприклад, сутність Користувача включає такі атрибути, як ім'я, електронна пошта, роль та дата реєстрації. Сутність Транспортний засіб фіксує інформацію про марку, модель, рік випуску, ціну та доступність. Сутність Бронювання зберігає деталі запитів на оренду, включаючи період оренди, статус, пов'язаного користувача та транспортний засіб, а також позначки часу створення та оновлень.

Зв'язки на ER-діаграмі визначають, як сутності взаємодіють одна з одною. Наприклад, Користувач може мати кілька записів Бронювання, що вказують на подані або отримані ним орендні послуги. Кожне Бронювання пов'язане лише з одним Транспортним засобом, тоді як Транспортний засіб може бути пов'язаний з кількома екземплярами Бронювання з часом. Аналогічно, Оголошення пов'язане з Транспортним засобом та Користувачем (постачальником оренди), що дозволяє системі відстежувати, які транспортні засоби пропонуються яким постачальником. Суб'єкти обміну повідомленнями пов'язані з користувачами та бронюваннями для ведення обліку комунікації та системних сповіщень.

Кардинальність та обмеження також є важливою частиною ER-діаграми. Вони визначають кількість екземплярів, які можуть брати участь у зв'язку, наприклад, зв'язки "один до багатьох" або "багато до багатьох". Наприклад, один користувач може надсилати кілька повідомлень, а один транспортний засіб може бути частиною кількох бронювань, але кожне бронювання має бути пов'язане лише з одним транспортним засобом та одним орендарем. Ці обмеження забезпечують цілісність даних та запобігають невідповідностям у базі даних.

ER-діаграма служить планом для реалізації реляційної бази даних. Вона керує створенням таблиць, первинних ключів, зовнішніх ключів та індексів,

гарантуючи, що всі необхідні зв'язки виконуються на рівні бази даних. Крім того, діаграма допомагає розробникам та адміністраторам баз даних зрозуміти структуру системи, підтримуючи ефективне проєктування запитів, звітність та майбутню масштабованість.

Підсумовуючи, ER-діаграма забезпечує візуальне та концептуальне представлення архітектури даних для системи бронювання автомобілів. Чітко визначаючи сутності, атрибути та зв'язки, вона гарантує, що проєкт бази даних є узгодженим, ефективним та здатним підтримувати функціональні та операційні вимоги системи, формуючи основу для надійної та масштабованої розробки програмного забезпечення.

Логічна модель системи представлена на рисунку 3.1.

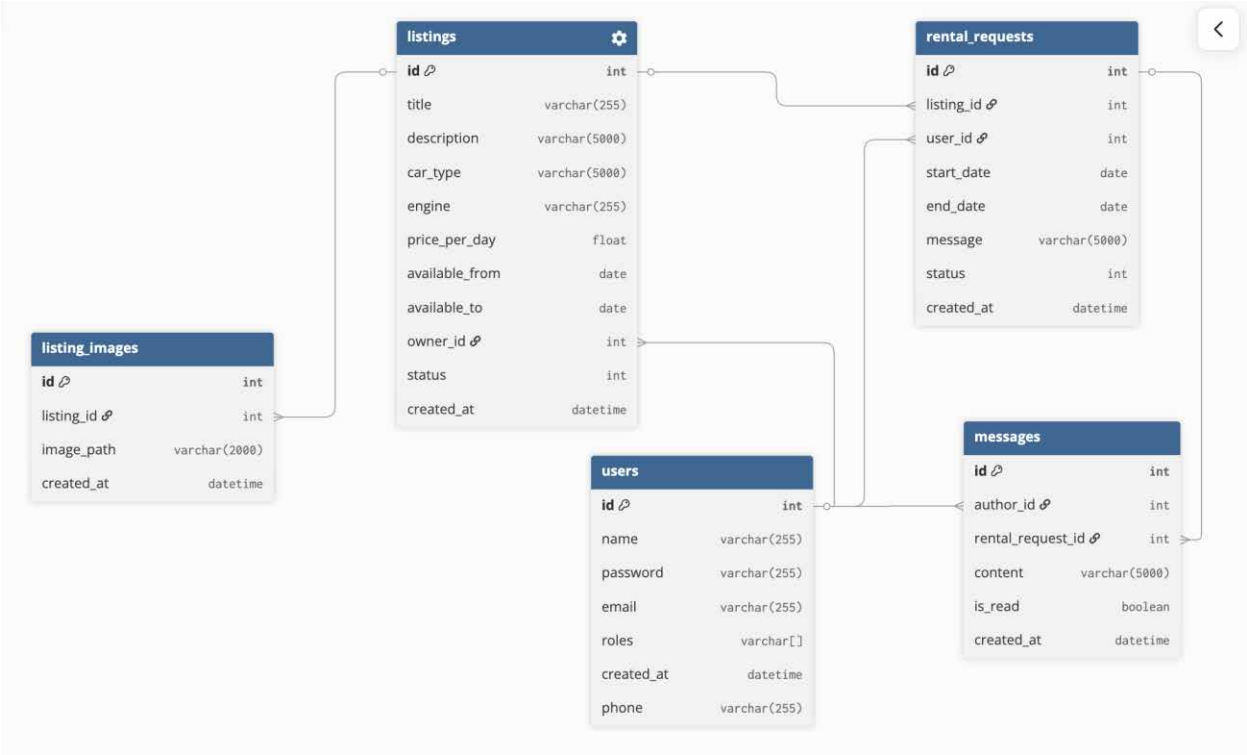

Рис. 3.1 ER-діаграма “carmate_db”

Ця ER-діаграма описує структуру бази даних для системи, яка управляє орендою, обробляючи користувачів, оголошення, запити на оренду та повідомлення між користувачами. Опис таблиць та зв'язків між ними:

Таблиця **User:**

1. id: Унікальний ідентифікатор користувача (первинний ключ);

2. name: Ім'я користувача;
3. email: Електронна пошта;
4. password: Пароль користувача;
5. phone: Телефон користувача;
6. avatar: Шлях до аватарки користувача;
7. createdAt: Дата та час створення запису;
8. updatedAt: Дата та час останнього оновлення запису.

Ця таблиця зберігає дані користувачів системи

Таблиця **Listing:**

1. id: Унікальний ідентифікатор оголошення (первинний ключ);
2. title: Заголовок оголошення;
3. description: Опис оголошення (необов'язкове);
4. car_type: тип авто (купе, седан, позашляховик);
5. engine: потужність двигуна авто;
6. pricePerDay: Ціна оренди за день;
7. availableFrom: Дата, з якої авто доступне для оренди (необов'язкове);
8. availableTo: Дата, до якої авто доступне для оренди (необов'язкове);
9. ownerId: Ідентифікатор користувача (власника), який створив оголошення;
10. status: Статус оголошення (наприклад, активне чи неактивне);
11. createdAt: Дата та час створення оголошення.

Ця таблиця зберігає дані оголошень для оренди, пов'язані з конкретними користувачами-власниками.

Таблиця **ListingImages:**

1. id: Унікальний ідентифікатор зображення (первинний ключ);
2. listingId: Ідентифікатор оголошення, до якого належить зображення;
3. imagePath: Шлях до зображення (необов'язкове);
4. createdAt: Дата та час створення запису зображення.

Ця таблиця зберігає зображення для оголошень про оренду. Кожне зображення прив'язане до конкретного оголошення.

Таблиця **RentalRequest:**

1. id: Унікальний ідентифікатор запиту на оренду (первинний ключ);
2. listingId: Ідентифікатор оголошення, на яке зроблений запит;
3. userId: Ідентифікатор користувача (орендаря), який зробив запит;
4. startDate: Дата початку оренди;
5. endDate: Дата кінця оренди;
6. message: Повідомлення користувача для орендодавця (необов'язкове);
7. status: Статус запиту (наприклад, очікує, схвалено або відхилено);
8. createdAt: Дата та час створення запиту.

Ця таблиця зберігає запити користувачів на оренду конкретних автівок. Кожен запит пов'язаний з користувачем та оголошенням.

Таблиця **Message:**

1. id: Унікальний ідентифікатор повідомлення (первинний ключ);
2. authorId: Ідентифікатор користувача, який написав повідомлення;
3. rentalRequestId: Ідентифікатор запиту на оренду, до якого відноситься повідомлення;
4. content: Текст повідомлення;
5. isRead: Статус прочитання повідомлення (boolean);
6. createdAt: Дата та час створення повідомлення.

Ця таблиця зберігає повідомлення, які користувачі надсилають стосовно запитів на оренду. Повідомлення можуть бути прочитані чи непрочитані.

У цій базі даних є кілька важливих зв'язків між таблицями, що визначають, як дані взаємодіють між собою.

Кожен користувач може створювати кілька оголошень, що пов'язано через поле ownerId у таблиці Listing, яке є зовнішнім ключем, що посилається на ідентифікатор користувача з таблиці User. Таким чином, один користувач може бути власником кількох автівок, що здаються в оренду.

Оголошення можуть містити багато зображень. Це забезпечується зв'язком через поле listingId у таблиці ListingImages, яке також є зовнішнім ключем і посилається на id в таблиці Listing. Таке відношення дозволяє додавати до кожного оголошення декілька зображень.

Кожне оголошення може бути пов'язане з кількома запитами на оренду. Зв'язок між таблицями Listing та RentalRequest реалізується через поле listingId у таблиці RentalRequest, яке є зовнішнім ключем і посилається на id в таблиці Listing. Це дозволяє користувачам робити запити на оренду для конкретного авто.

Користувачі також можуть подавати кілька запитів на оренду, і цей зв'язок між таблицями User і RentalRequest забезпечується через поле userId у таблиці RentalRequest, що є зовнішнім ключем, який посилається на ідентифікатор користувача з таблиці User.

Кожен запит на оренду може мати кілька повідомлень, що обробляються в таблиці Message. Зв'язок між цими двома таблицями встановлюється через поле rentalRequestId у таблиці Message, яке є зовнішнім ключем і посилається на id в таблиці RentalRequest. Це дозволяє відстежувати всі повідомлення, пов'язані з конкретними запитами.

Кожен користувач може надсилати кілька повідомлень. Зв'язок між таблицями User і Message визначається через поле authorId у таблиці Message, яке є зовнішнім ключем і посилається на ідентифікатор користувача з таблиці User. Цей зв'язок дозволяє зберігати всі повідомлення, які надіслав конкретний користувач.

Таким чином, система організовує зв'язки між користувачами, оголошеннями, запитами на оренду та повідомленнями, що забезпечує зручне та ефективне управління даними в межах платформи.

3.2 Вибір системи управління базою даних та її реалізація

Система управління базами даних (СУБД) є програмним забезпеченням, призначеним для збереження, організації, управління та ефективного отримання даних. СУБД забезпечує структуроване середовище, яке дозволяє додаткам взаємодіяти з базою даних через запити, транзакції та інші механізми керування даними, гарантуючи їхню цілісність, безпеку та надійність. Вибір відповідної СУБД є ключовим етапом розробки веб-застосунку, зокрема системи бронювання автомобілів, оскільки він безпосередньо впливає на продуктивність, масштабованість та підтримуваність системи.

Існують різні типи СУБД, кожен із яких підходить для певних завдань. Реляційні СУБД організовують дані у таблиці з рядками та стовпцями, а зв’язки між таблицями визначаються через зовнішні ключі. Прикладами є MySQL, PostgreSQL, Oracle Database та Microsoft SQL Server. Вони підходять для застосунків, що потребують суворої узгодженості даних, складних SQL-запитів і складних взаємозв’язків між даними, таких як керування користувачами, оголошеннями та заявками на оренду у системі прокату автомобілів.

Інший тип — NoSQL СУБД, які призначені для гнучкості та масштабованості, часто працюють із неструктурованими або напівструктурованими даними. До них відносяться документно-орієнтовані бази, наприклад MongoDB, ключ-значення, такі як Redis, колонкові, наприклад Cassandra, і графові бази, такі як Neo4j. NoSQL є корисними, коли застосунок потребує швидкого масштабування, горизонтального розподілу або гнучких схем, хоча інколи це може позначатися на строгій транзакційній узгодженості.

Інший напрям — СУБД в пам’яті, такі як Redis або Memcached, які зберігають дані в оперативній пам’яті для максимально швидкого читання та запису. Вони найчастіше використовуються як кешуючий шар, а не як основна база даних. Новітні NewSQL СУБД намагаються поєднати масштабованість NoSQL із транзакційною узгодженістю традиційних реляційних систем, прикладами є CockroachDB і Google Spanner.

Для розробки системи бронювання автомобілів було обрано реляційну СУБД MySQL. Вона забезпечує надійну роботу зі структурованими даними, підтримує транзакції відповідно до принципів ACID, зовнішні ключі та складні SQL-запити, що необхідно для управління взаємозв’язками між користувачами, оголошеннями, заявками на оренду, повідомленнями та зображеннями. Крім того, MySQL добре інтегрується з PHP і Symfony, що спрощує розробку та підвищує підтримуваність коду.

Реалізація бази даних передбачає створення таблиць для кожної сутності (User, Listing, ListingImages, RentalRequest, Message) з відповідними типами даних, первинними та зовнішніми ключами, а також обмеженнями для забезпечення цілісності даних. Взаємозв’язки між сутностями моделюються відповідно до бізнес-логіки: користувачі володіють оголошеннями, орендарі подають заявки, оголошення можуть мати декілька зображень, а повідомлення пов’язані із заявками на оренду. Застосовуються індекси та нормалізація для забезпечення ефективного виконання запитів, зокрема під час пошуку та генерації звітів.

Вибір MySQL для розробки системи бронювання автомобілів був обумовлений кількома важливими факторами, що забезпечують ефективну реалізацію проєкту та зручність подальшого обслуговування. По-перше, MySQL є реляційною СУБД, яка надійно підтримує структуровані дані та забезпечує строгі правила цілісності, що особливо важливо для системи, де зберігаються зв’язані сутності: користувачі, оголошення, заявки на оренду, повідомлення та зображення автомобілів. Використання зовнішніх ключів і транзакцій ACID гарантує, що дані не будуть пошкоджені навіть у випадку помилок або одночасного доступу декількох користувачів.

По-друге, MySQL широко підтримується у поєднанні з PHP та фреймворком Symfony, що значно спрощує розробку та інтеграцію бази даних у веб-застосунок. Наявність готових бібліотек і ORM Doctrine дозволяє швидко реалізувати складні взаємозв’язки між сутностями без необхідності ручного

написання великої кількості SQL-запитів, що прискорює розробку та зменшує ризик помилок.

Ще однією перевагою є висока продуктивність і оптимізація MySQL для операцій читання та запису, що забезпечує швидкий доступ до даних при великій кількості користувачів, пошуку автомобілів та обробці бронювань у реальному часі. Система індексів і можливості масштабування дозволяють забезпечити ефективну роботу навіть при збільшенні обсягів даних і кількості одночасних запитів.

Також важливою причиною вибору є стабільність і перевіреність MySQL у промислових проєктах. Вона має великий досвід використання у комерційних системах, що підвищує надійність проєкту та дозволяє швидко знаходити рішення у разі технічних проблем завдяки активній спільноті розробників і документації.

Вибір MySQL як СУБД забезпечує системі бронювання автомобілів надійну, масштабовану та підтримувану основу для зберігання та обробки всіх критично важливих даних, дозволяючи одночасно виконувати транзакційні операції, такі як підтвердження бронювання, та аналітичні, наприклад, відстеження історії заявок і формування звітів.

Таким чином було створено базу даних в середовищі MySQL Workbench (Рис. 3.2).

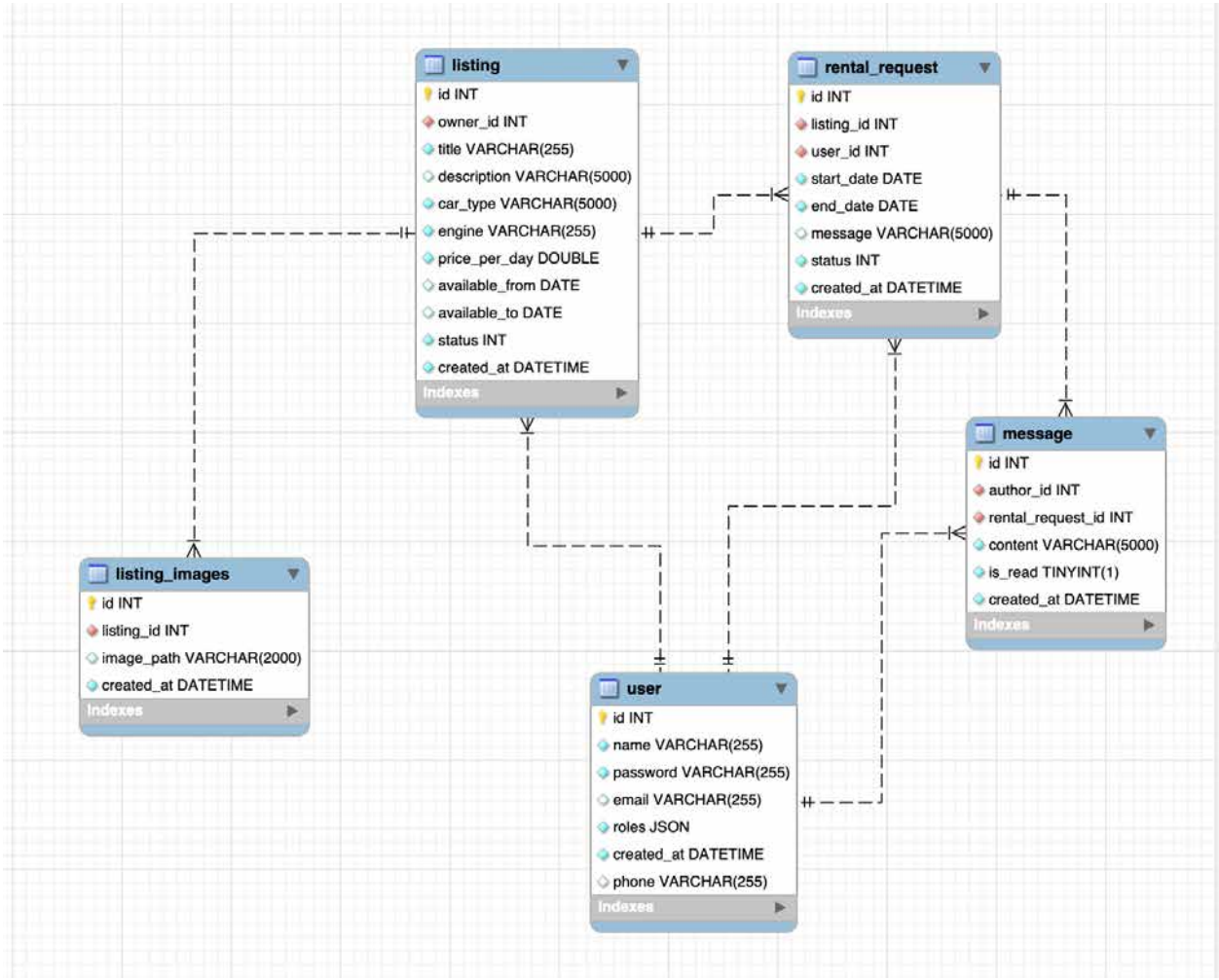

Рис. 3.2 База даних

Ця структура містить таблицю користувачів, яка зберігає інформацію про орендарів та орендодавців, включаючи ім'я, контактні дані та статус реєстрації. Таблиця оголошень містить деталі пропозицій оренди, дату розміщення та кількість оголошень. Взаємопов’язана з нею таблиця бронювання, яка фіксує періоди оренди, суму оплати та статус заявки.

Зв’язки між цими таблицями забезпечують узгоджений та зручний процес оренди, об’єднуючи дані про користувачів, їхні оголошення, бронювання та спілкування.

3.3 Архітектура програмного забезпечення

Архітектура програмного забезпечення визначає високорівневу структуру системи, окреслюючи, як взаємодіють її компоненти, як передаються

дані та як організована функціональність. У контексті системи бронювання автомобілів для служби прокату, добре спланована архітектура є критично важливою для забезпечення надійності, зручності обслуговування, масштабованості та безпеки. Архітектура служить планом, що керує процесом розробки, допомагаючи розробникам зрозуміти обов'язки кожного модуля та взаємозв'язки між ними.

Архітектура цієї системи бронювання автомобілів базується на багаторівневій моделі, яка розділяє програму на окремі шари, кожен з яких відповідає за конкретні завдання. Рівень представлення забезпечує інтерфейс користувача та обробляє взаємодію з орендарями та постачальниками послуг прокату. Цей рівень гарантує, що користувачі можуть переглядати оголошення, подавати запити на оренду, спілкуватися через чат та керувати своїми профілями у адаптивний та інтуїтивно зрозумілий спосіб. Рівень бізнес-логіки інкапсулює основні робочі процеси системи, включаючи управління оголошеннями про транспортні засоби, обробку запитів на оренду, розрахунок наявності та цін, а також забезпечення дотримання правил для ролей та дозволів користувачів. Таке розділення гарантує, що логіка програми залишається незалежною від інтерфейсу користувача, що забезпечує гнучкість у внесенні змін або розширенні функціональності.

Рівень доступу до даних взаємодіє з базою даних MySQL, обробляючи пошук, зберігання та оновлення даних. Він реалізує запити, об'єднання та транзакції, необхідні для підтримки узгодженості та цілісності між такими сутностями, як користувачі, оголошення, бронювання, повідомлення та зображення. Використання Doctrine ORM із Symfony спрощує зіставлення об'єктів PHP з таблицями бази даних, дозволяючи розробникам працювати з сутностями, а не з сирим SQL, зберігаючи при цьому ефективний доступ до даних та підтримуючи складні запити.

У цій архітектурі модулі організовані відповідно до функціональних областей системи. Модуль керування користувачами обробляє реєстрацію, автентифікацію та управління профілями. Модуль керування оголошеннями

дозволяє постачальникам послуг з оренди створювати, оновлювати та видаляти оголошення про автомобілі, тоді як модуль запитів на оренду керує життєвим циклом бронювань від створення до затвердження або відхилення. Модуль зв'язку керує обміном повідомленнями між користувачами в контексті запитів на оренду, а модуль сповіщень сповіщає користувачів про зміни статусу бронювання або нові повідомлення. Кожен модуль має чітко визначені інтерфейси, що забезпечує модульність та легкість обслуговування.

Безпека вбудована в усю архітектуру. Механізми автентифікації та авторизації на основі ролей обмежують доступ до конфіденційних операцій, таких як зміна оголошення або затвердження бронювання. Перевірка введених даних, шифрування паролів та протоколи безпечного зв'язку захищають конфіденційні дані. Крім того, архітектура підтримує ведення журналу для відстеження діяльності системи та забезпечення підзвітності.

Мірки масштабованості враховуються завдяки модульній конструкції, ефективному індексуванню бази даних, а також підтримці кешування та асинхронної обробки для таких завдань, як надсилання сповіщень або створення звітів. Це гарантує, що система може обробляти збільшений трафік, зростаючі обсяги даних та кількох одночасних користувачів без зниження продуктивності.

Програмна архітектура системи бронювання автомобілів підкреслює багаторівневу, модульну та безпечну конструкцію. Вона розділяє питання презентації, бізнес-логіки та доступу до даних, чітко визначаючи обов'язки кожного модуля. Такий підхід забезпечує ремонтопридатність, масштабованість та надійність, забезпечуючи міцну основу для реалізації функціональних та нефункціональних вимог системи.

Загалом, вибір 4-рівневої архітектури для цього програмного забезпечення для пошуку та оренди авто забезпечує чисте, модульне та масштабоване рішення. Це забезпечує більшу гнучкість у розробці, легше обслуговування та більш керований шлях зростання програми, оскільки потрібні нові функції або оптимізації.

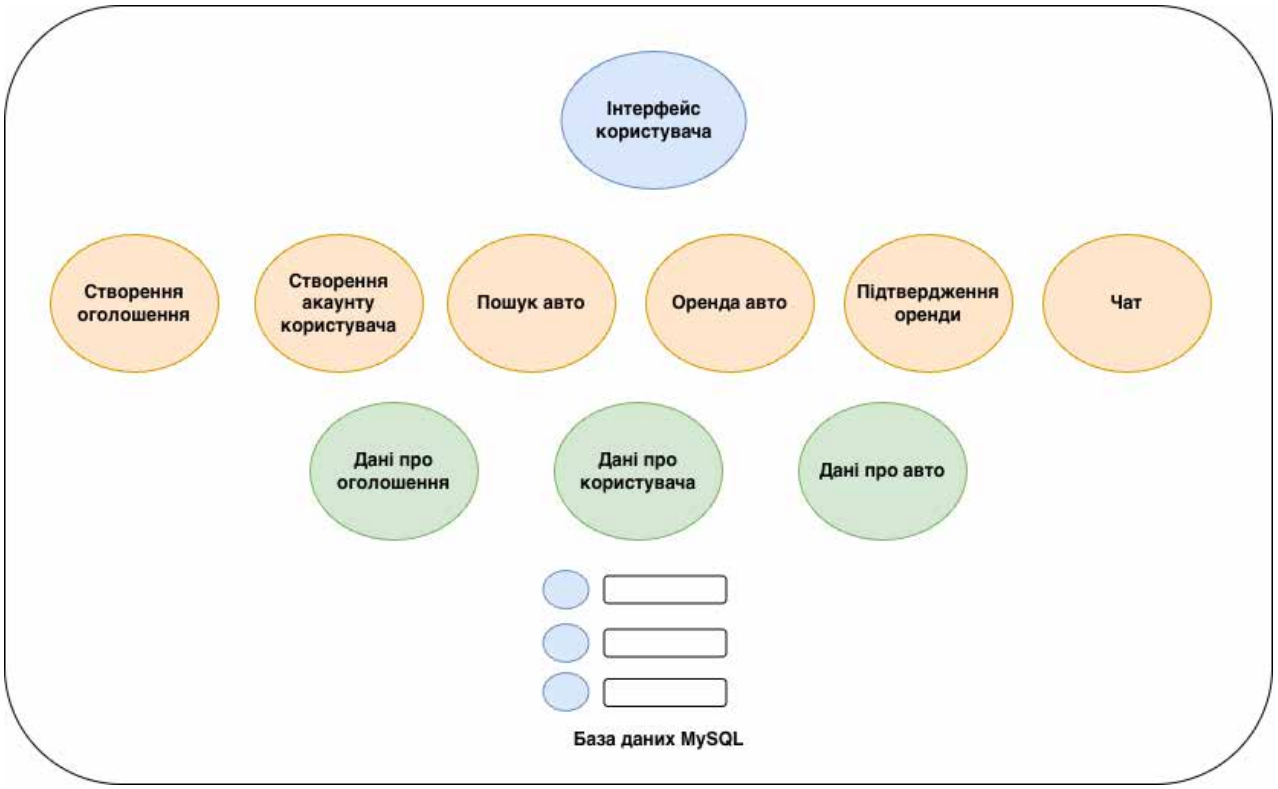

Рис. 3.3 Багатошарова архітектура

Ця архітектура відображає ключові компоненти системи оренди, їхні взаємозв'язки та загальні механізми функціонування. У центрі моделі знаходяться основні структурні блоки, які забезпечують безперебійний процес взаємодії між користувачами.

Система включає модуль управління оголошеннями, що дозволяє створювати, редагувати та шукати доступні варіанти оренди. Він пов'язаний з модулем бронювання, який забезпечує перевірку заявок, підтвердження угоди та реєстрацію необхідних даних. Крім того, архітектура містить систему комунікації, де користувачі можуть обмінюватися повідомленнями, уточнювати деталі та узгоджувати умови співпраці.

Взаємозв'язки між компонентами гарантують логічний і структурований процес. Наприклад, після вибору оголошення інформація передається в модуль бронювання. Додатково, інтеграція із системою чату забезпечує прозорий обмін повідомленнями між орендарем та орендодавцем.

Ця архітектура спрямована на підвищення зручності користування, оптимізацію процесів та забезпечення надійного управління орендними операціями.

3.4 Організаційна структура програмного забезпечення

3.4.1 Діаграма пакетів. Діаграма пакетів — це тип UML-діаграми, яка використовується для організації та групування пов'язаних елементів системи в одиниці вищого рівня, що називаються пакетами. У розробці програмного забезпечення діаграми пакетів допомагають візуалізувати модульну структуру програми, показуючи, як групуються класи, модулі та підсистеми, а також як ці групи взаємодіють одна з одною. Для складних систем, таких як система бронювання автомобілів для служби прокату, діаграми пакетів є важливими для розуміння залежностей та підтримки чистої, зручної в обслуговуванні архітектури.

У системі бронювання автомобілів програма поділена на кілька пакетів на основі функціональності та відповідальності. Пакет «Керування користувачами» містить усі класи та компоненти, відповідальні за реєстрацію, автентифікацію та управління профілями. Пакет «Керування оголошеннями» включає класи, які обробляють оголошення про автомобілі, деталі транспортного засобу, дати доступності та пов'язані зображення. Пакет «Запит на оренду» інкапсулює логіку створення, затвердження, відхилення та відстеження бронювань, забезпечуючи ефективне керування запитами на оренду користувачами. Пакет «Обмін повідомленнями» керує зв'язком між користувачами в контексті запитів на оренду, тоді як пакет «Повідомлення» обробляє сповіщення та системні сповіщення, щоб інформувати користувачів про зміни в бронюваннях або повідомленнях.

Діаграма пакета також ілюструє залежності між цими пакетами. Наприклад, пакет Rental Request залежить як від пакетів Listing Management, так і від пакетів User Management для отримання інформації про транспортні засоби

та орендарів. Аналогічно, пакети Messaging та Notification взаємодіють з пакетом Rental Request, щоб надавати відповідні повідомлення та сповіщення, пов'язані з певними бронюваннями. Така ієрархічна організація допомагає зменшити зв'язки, що полегшує зміну або розширення окремих пакетів без впливу на непов'язані частини системи.

Використання схеми пакетів дозволяє розробникам та зацікавленим сторонам швидко зрозуміти загальну структуру програми, визначити області відповідальності та забезпечити відповідність модуляризації системи найкращим практикам у програмній інженерії. Чітко визначаючи межі та взаємодії кожного пакета, схема сприяє кращій підтримці, масштабованості та зрозумілості системи.

Схема пакетів для системи бронювання автомобілів забезпечує загальне уявлення про модульну структуру програми. Вона групує пов'язані класи та компоненти в цілісні пакети, виділяє залежності між пакетами та служить керівництвом для розробки, обслуговування та майбутнього розширення системи.

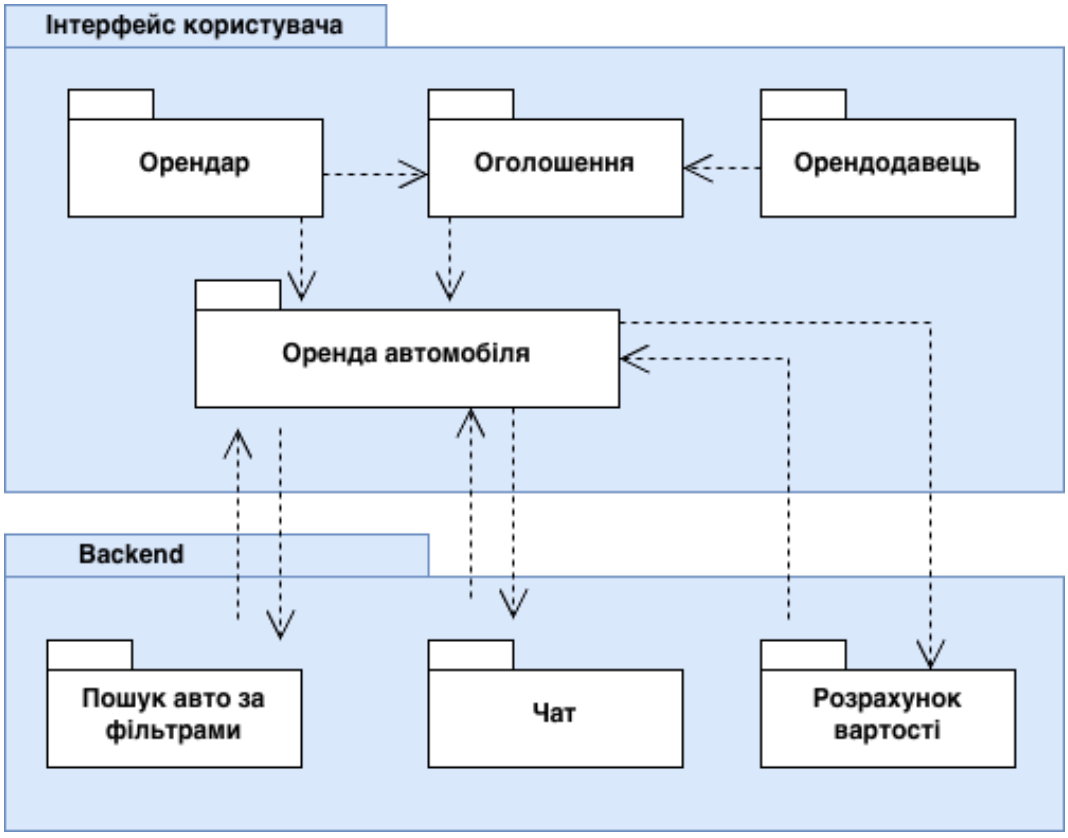

Рис. 3.4 Діаграма пакетів

Ця діаграма пакетів відображає структуру системи та спосіб організації її компонентів у логічні групи. Вона допомагає зрозуміти, як різні частини взаємодіють між собою, забезпечуючи цілісну роботу програми.

Діаграма включає кілька пакетів, кожен з яких виконує певну роль. Один із ключових пакетів – це управління оголошеннями, який містить класи для створення, пошуку та редагування оголошень. Він взаємодіє з модулем бронювання, що забезпечує процес реєстрації заявок, перевірку доступності та підтвердження угоди.

Крім того, представлено пакет комунікації містить механізми обміну повідомленнями, гарантуючи зручний зв’язок між орендарем та орендодавцем. Завершує архітектуру пакет користувачів, де зберігаються дані про зареєстрованих осіб, їхні профілі та права доступу.

Ця структура дозволяє ефективно розподілити функціональність, спростити розробку та забезпечити чіткі межі між логічними частинами системи.

3.5 Вибір інструментарію для створення програмного забезпечення

Вибір правильного набору інструментів розробки є важливим для створення надійної, зручної в обслуговуванні та ефективної програмної системи. Для системи бронювання автомобілів вибір технологій визначався вимогами розробки веб-додатків, необхідністю масштабованості та сумісністю компонентів в екосистемі розробки.

Мову програмування PHP було обрано як основну серверну мову завдяки її широкому поширенню, легкості інтеграції з веб-технологіями та сильній підтримці об'єктно-орієнтованого програмування. PHP забезпечує гнучкість, необхідну для реалізації складної бізнес-логіки, керування сеансами користувачів та ефективної взаємодії з базою даних. Для структурування проєкту та сприяння швидкій розробці було обрано фреймворк Symfony. Symfony пропонує модульну архітектуру, вбудовані інструменти для маршрутизації,

шаблонизації, безпеки та управління базами даних, а також інтеграцію з Doctrine ORM для об'єктно-реляційного відображення, що спрощує взаємодію з базою даних MySQL.

Для постійного зберігання даних MySQL було використано як систему управління реляційними базами даних. MySQL забезпечує цілісність даних, підтримує складні запити та дозволяє реалізацію зв'язків зовнішніх ключів, які відображають структуру системи, таку як зв'язки між користувачами, оголошеннями, запитами на оренду, повідомленнями та зображеннями.

На стороні фронтенду поєднання HTML, CSS та JavaScript дозволяє створювати адаптивні та інтерактивні інтерфейси. Bootstrap 5 було використано для прискорення розробки зручних макетів та компонентів, забезпечуючи узгоджений дизайн на різних пристроях та розмірах екранів. JavaScript покращує інтерактивність, обробляючи такі завдання, як оновлення в режимі реального часу в системі чату, динамічні форми та перевірка на стороні клієнта.

Для полегшення розробки, тестування та розгортання для контейнеризації було використано Docker. Docker забезпечує відтворюване середовище, яке забезпечує узгодженість між розробкою та виробництвом, дозволяючи системі надійно працювати незалежно від базової операційної системи. Це спрощує управління залежностями та зменшує кількість помилок конфігурації.

Крім того, Photoshop CS6 було використано для завдань графічного дизайну, таких як створення піктограм, банерів та макетів інтерфейсу. Це гарантує, що візуальні аспекти програми, включаючи зображення для оголошень про продаж автомобілів, є високоякісними та сприяють професійному та бездоганному користувацькому досвіду.

Загалом, обрані інструменти поєднують потужний бекенд із гнучким та адаптивним фронтендом, що підтримується ефективним управлінням даними та робочими процесами розробки. Цей технологічний стек гарантує, що система бронювання автомобілів є надійною, зручною в обслуговуванні та

масштабованою, забезпечуючи надійну платформу як для постачальників послуг з прокату, так і для користувачів, які шукають автомобілі в оренду.

3.6 Аналіз нормалізації бази даних

Для забезпечення цілісності даних, мінімізації надлишковості (дублювання інформації) та усунення потенційних аномалій модифікації (вставлення, оновлення та видалення) під час проєктування реляційної бази даних «carmate_db» було проведено послідовну нормалізацію структури таблиць до третьої нормальної форми (3НФ).

Перша нормальна форма (1НФ)

Схема бази даних перебуває в 1НФ, оскільки всі атрибути в усіх проєктованих таблицях (users, listings, listing_images, rental_requests, messages) є атомарними (неподільними), а самі таблиці не містять повторюваних груп або багатозначних атрибутів. Наприклад:

- Контактні дані користувача (phone, email у таблиці users) та параметри автомобіля (car_type, engine у таблиці listings) розділені на окремі скалярні поля.
- Замість збереження масиву посилань на зображення автомобіля у вигляді текстового рядка через кому всередині таблиці listings (що було б прямим порушенням вимоги атомарності), реалізовано декомпозицію сутностей.

Друга нормальна форма (2НФ)

Схема бази даних відповідає вимогам 2НФ, оскільки вона задовольняє умови 1НФ, і кожен неключовий атрибут має повну функціональну залежність від первинного ключа (Primary Key) відповідної таблиці.

Оскільки в усіх розроблених сутностях як первинні ключі використовуються штучні (сурогатні) одноатрибутні ідентифікатори id, часткова залежність неключових атрибутів від елементів складеного ключа математично

неможлива. Усі характеристики оголошення (заголовок, ціна, опис) залежать виключно від конкретного id в таблиці listings, а дані запиту на оренду — від id в таблиці rental_requests.

Третя нормальна форма (3НФ)

Схема бази даних зведена до 3НФ, оскільки вона перебуває в 2НФ, і в ній повністю відсутні транзитивні залежності неключових атрибутів від первинного ключа (тобто ситуації, коли неключовий атрибут функціонально залежить від іншого неключового атрибута). Усі поля таблиць є взаємно незалежними. Наприклад, статус заявки (status) чи супроводжувальне повідомлення (message) у таблиці rental_requests залежать безпосередньо від ідентифікатора заявки id, а не через проміжні сутності користувача чи оголошення.

Архітектурна декомпозиція сутностей як результат нормалізації:

Процес нормалізації логічної моделі безпосередньо визначив необхідність винесення логічно пов'язаних даних в окремі реляційні таблиці з організацією зв'язків типу ManyToOne та OneToMany:

1. **Винесення медіаресурсів у ListingImages**: збереження шляхів до графічних файлів безпосередньо в таблиці `listings` призвело б або до жорсткого обмеження кількості фотографій (структурна дефіцитність), або до порушення 1НФ. Створення окремої таблиці `listing_images` із зовнішнім ключем `listing_id` дозволило динамічно масштабувати кількість фотографій для одного автомобіля без надлишковості даних.
2. **Винесення транзакцій у RentalRequest**: спроба інтегрувати дані про бронювання в таблицю `listings` (наприклад, у вигляді полів орендаря та дат) унеможливила б збереження історії прокату та обробку кількох одночасних заявок (порушення 2НФ та 3НФ через дублювання даних автомобіля для кожного нового клієнта). Виділення сутності rental_requests забезпечило незалежне керування життєвим циклом кожної окремої угоди.

3. **Винесення комунікацій у Message**: повідомлення внутрішнього чату логічно залежать від контексту конкретного бронювання, а не просто від користувачів. Зберігання текстових повідомлень поза окремою сутністю спричинило б транзитивну залежність (User RentalRequest текст повідомлення). Декомпозиція комунікацій у таблицю messages із прив'язкою до rental_request_id та author_id дозволила зберегти структуру в 3НФ, забезпечивши швидку фільтрацію та індексацію повідомлень у межах конкретного діалогу.

Таким чином, розроблена реляційна структура бази даних є повністю нормалізованою, що гарантує високу швидкодію виконання SQL-запитів, консистентність даних на рівні СУБД MySQL та відсутність логічних колізій під час тривалої експлуатації веб-системи.

4 ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЯ СИСТЕМИ

4.1 Вимоги до апаратного та програмного забезпечення

Визначення вимог до апаратного та програмного забезпечення є важливим кроком у розробці веб-застосунку, оскільки це забезпечує ефективну та надійну роботу як середовища розробки, так і розгорнутої системи. Для системи бронювання автомобілів ці вимоги відображають потреби бекенду, фронтенду, бази даних та допоміжних інструментів, забезпечуючи безперебійну роботу за типових умов використання.

Щодо апаратного забезпечення, систему можна розгорнути на стандартних конфігураціях сервера. Для розробки та тестування рекомендується персональний комп'ютер з принаймні чотириядерним процесором, 8 ГБ оперативної пам'яті та 256 ГБ сховища для одночасного запуску застосунку, бази даних та контейнерних сервісів через Docker. Для виробничої роботи бажано використовувати сервер з чотириядерним або потужнішим процесором, 16 ГБ оперативної пам'яті та достатнім сховищем для забезпечення зростання бази даних та медіа-активів. Система також повинна підтримувати мережеве підключення з надійною пропускною здатністю для обробки кількох одночасних користувачів, які отримують доступ до оголошень, надсилають запити на оренду та обмінюються повідомленнями.

Вимоги до програмного забезпечення включають операційну систему, середовище програмування, базу даних, вебсервер та допоміжні інструменти. Бекенд побудовано за допомогою PHP (версії 8.1 або вище) та фреймворку Symfony, який забезпечує структуру для програми, маршрутизацію, безпеку та інтеграцію з базою даних. MySQL служить реляційною системою керування базами даних, зберігаючи всі сутності, такі як користувачі, списки, запити на оренду, повідомлення та зображення. Фронтенд спирається на HTML, CSS та JavaScript, а Bootstrap 5 забезпечує адаптивний фреймворк дизайну для забезпечення зручності використання на різних пристроях.

Для розробки та розгортання Docker використовується для створення ізольованих, відтворюваних середовищ, що забезпечує узгодженість між етапами розробки, тестування та виробництва. Також можуть знадобитися такі інструменти, як Composer для керування залежностями PHP та Node.js для компіляції ресурсів фронтенду. Для графіки та дизайну інтерфейсу використовується Photoshop CS6 для створення зображень, значків та макетів інтерфейсу.

Додаткові вимоги до програмного забезпечення включають сучасний веб-браузер, такий як Google Chrome або Mozilla Firefox, для тестування інтерфейсу користувача та інструменти командного рядка для керування операціями сервера та міграцією бази даних. Для відстеження стану програми та активності користувачів може бути впроваджено програмне забезпечення для моніторингу безпеки та продуктивності.

Визначаючи ці вимоги до апаратного та програмного забезпечення, проєкт гарантує, що система бронювання автомобілів буде надійною, масштабованою та зручною для користувача, забезпечуючи достатні ресурси для розробки, тестування та розгортання у виробничому середовищі, зберігаючи при цьому продуктивність та безпеку системи.

Також було зроблено діаграму розгортання для візуального огляду деплою системи (Рис. 4.1).

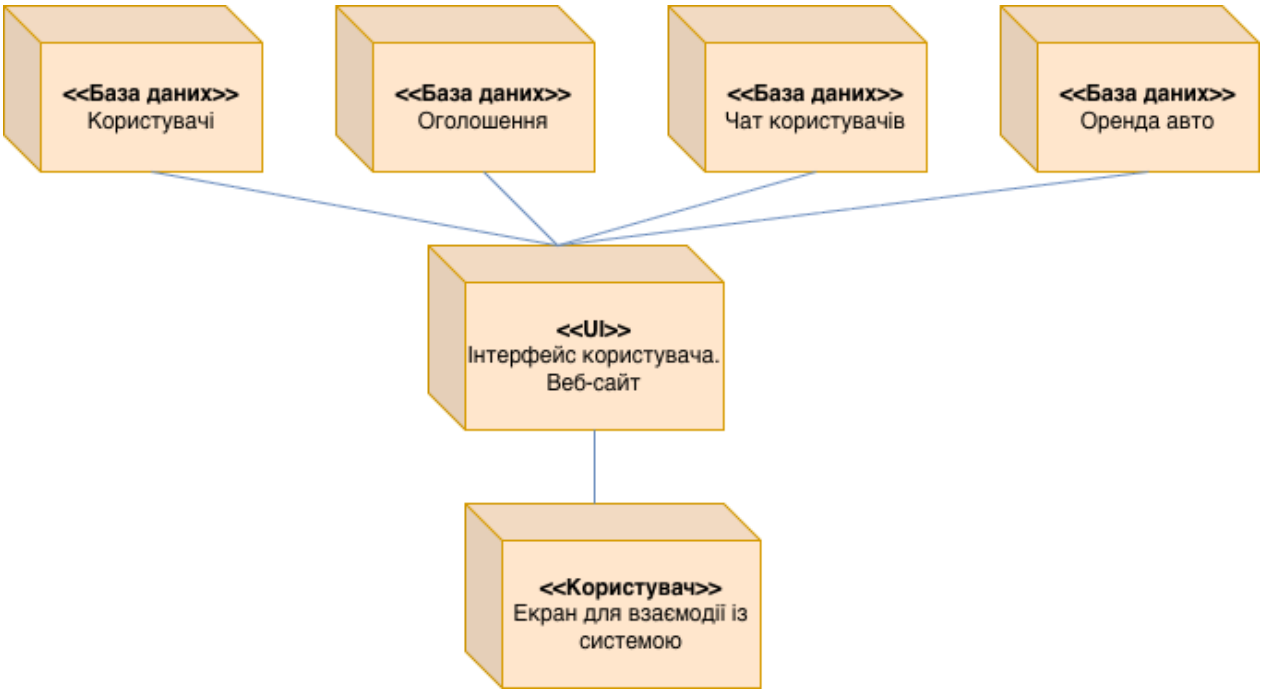

Рис. 4.1 Діаграма розгортання

4.2 Тестування системи

Запустивши систему, потрапляємо на форму авторизації, тут нам треба ввести логін та пароль користувача (рис. 4.2).

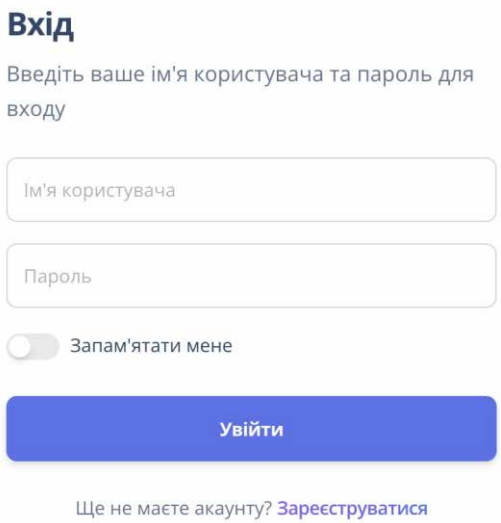

Рис. 4.2 Форма авторизації

Після авторизації ми переходимо на головну сторінку програми. Тут у нас виводиться різна статистична інформація про оренду та заробіток, а також особисті заявки на оренду (Рис. 4.3).

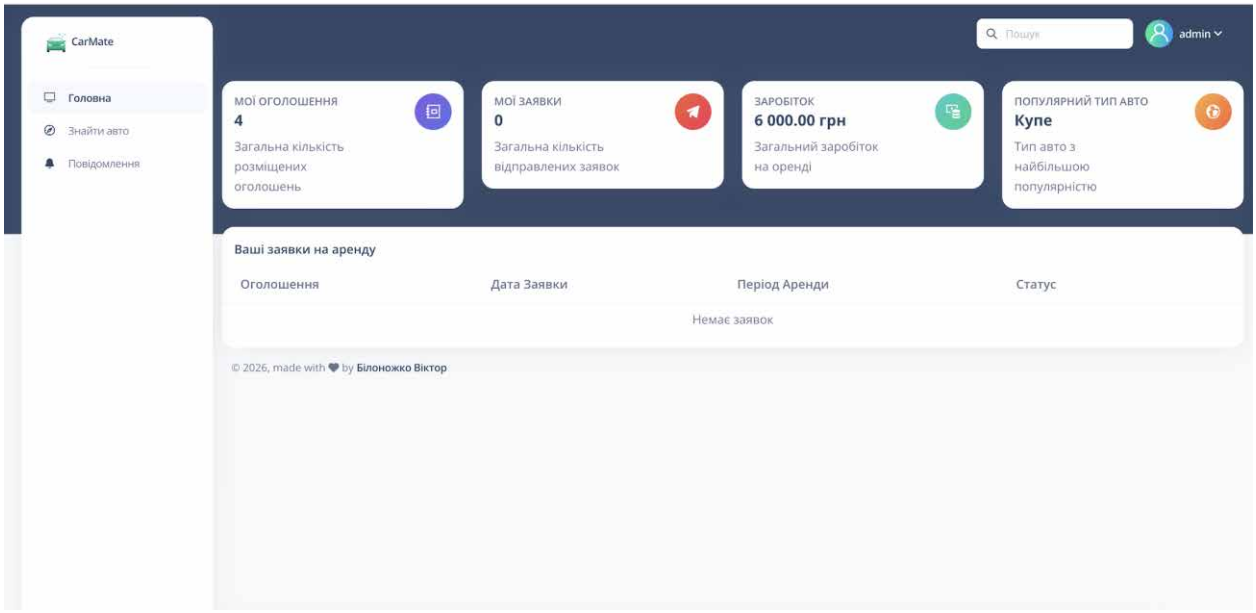

Рис. 4.3 Головна сторінка

Переходимо на сторінку "Знайти авто" і у нас виводиться список всіх створених оголошень (Рис. 4.4), ми можемо шукати цікаві для нас автомобілі за ключовими словами (Рис. 4.5), а також відкривати інтерактивну карту (Рис. 4.6).

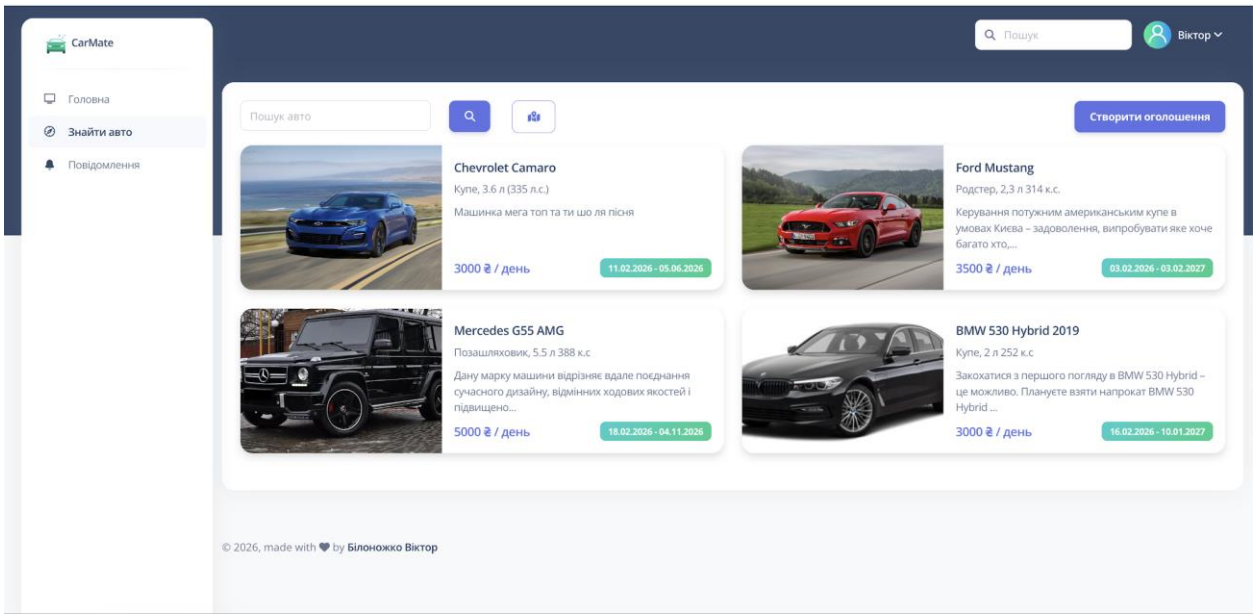

Рис. 4.4 Сторінка “Знайти авто”

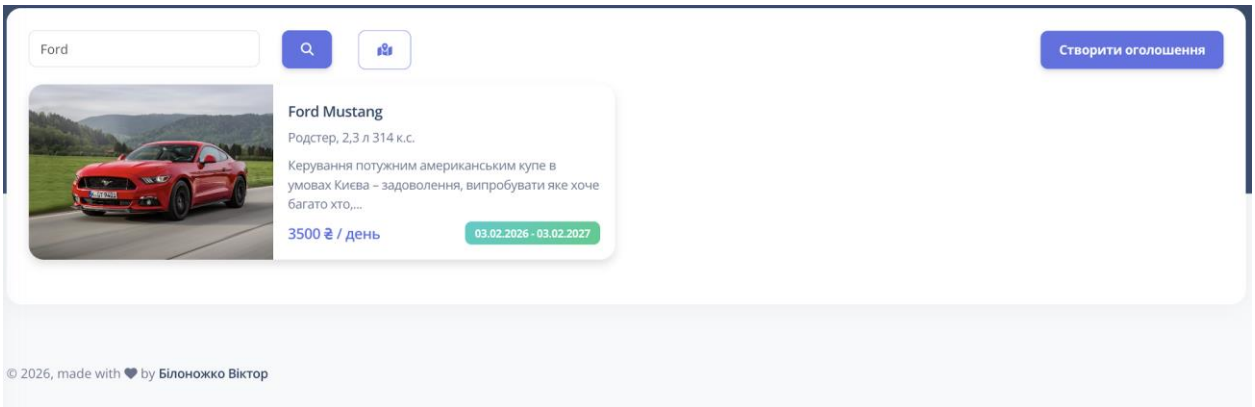

Рис. 4.5 Пошук за критеріями

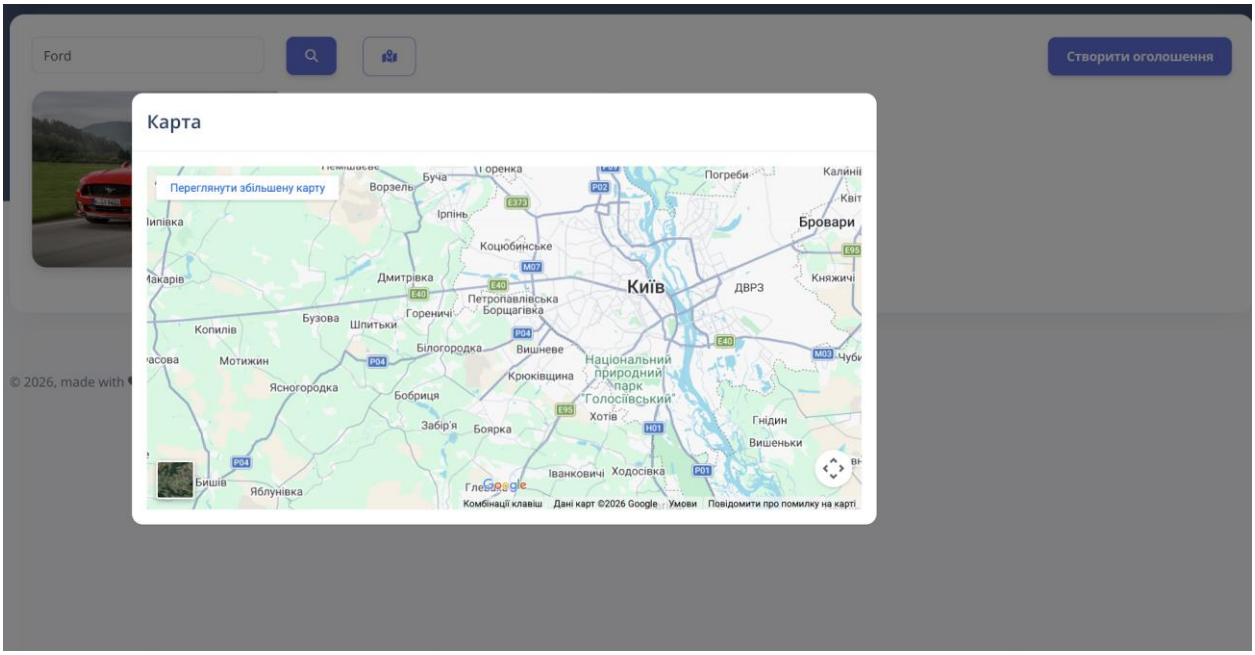

Рис. 4.6 Інтерактивна карта

Спробуємо створити оголошення, для цього нам потрібно натиснути кнопку створення оголошення на сторінці з пошуком авто, яка нам відкриє форму заповнення де нам необхідно ввести всі дані про автомобіль: назву, опис, фотографії, ціну за добу і т.д. Після підтвердження, наше оголошення буде успішно додано до бази даних (Рис. 4.7).

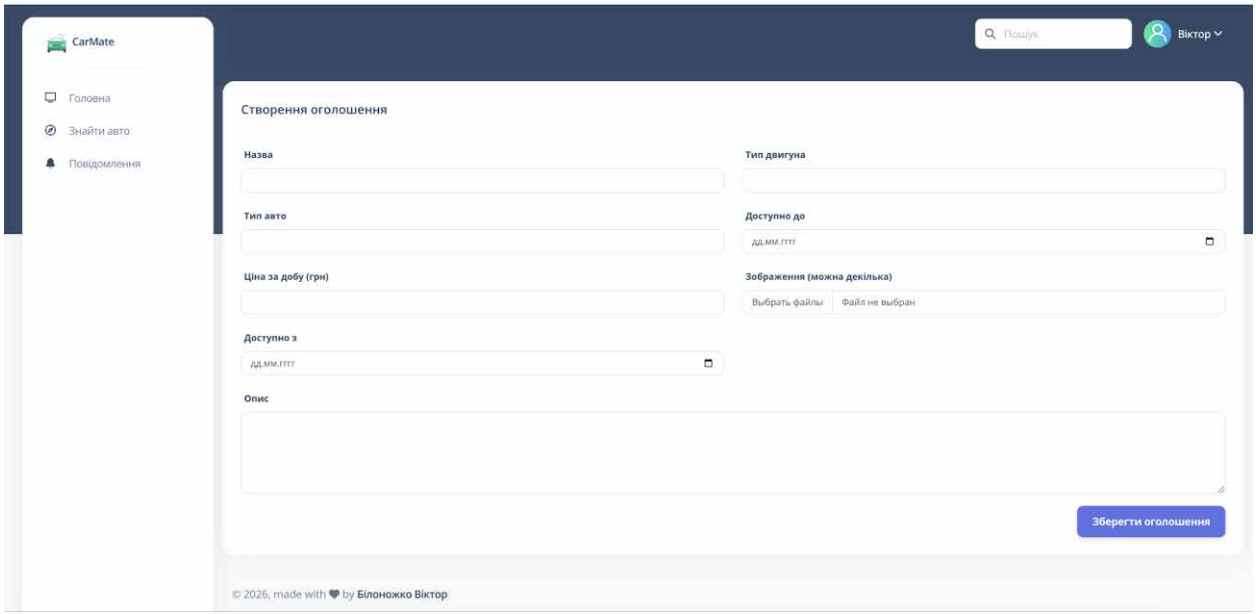

Рис. 4.7 Сторінка створення оголошення

Ми тепер можемо зайти на сторінку будь-якого створеного оголошення та переглянути детальну інформацію. Тут у нас представлені всі дані, всі фотографії авто і кнопки для оренди (Рис. 4.8).

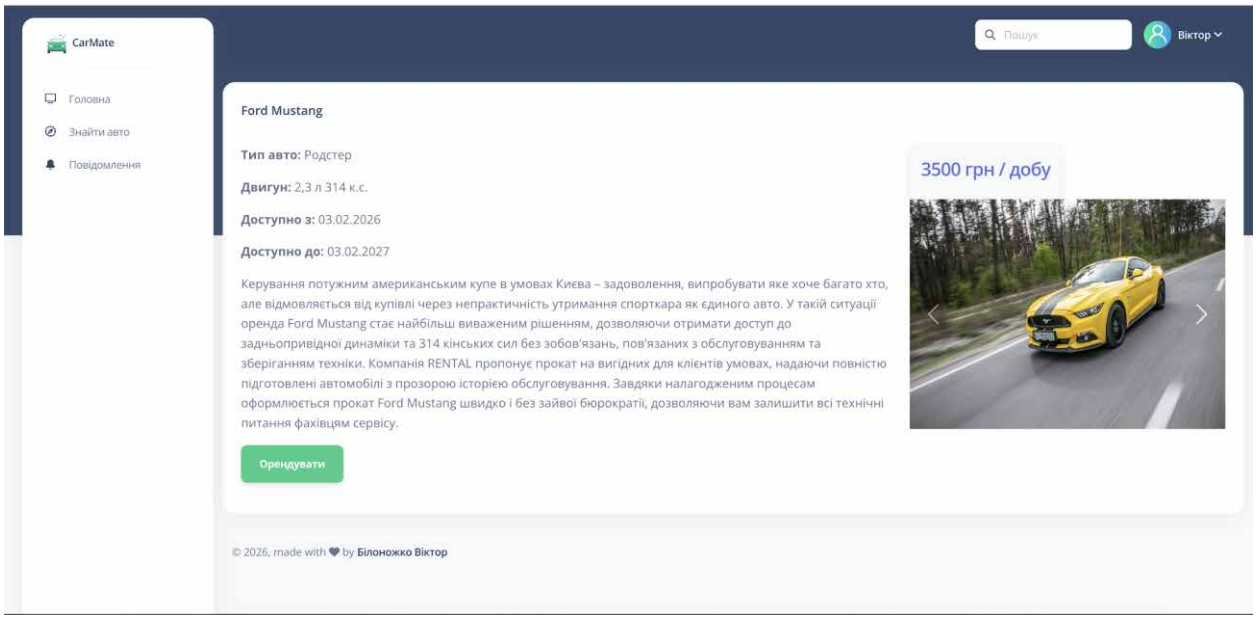

Рис. 4.8 Сторінка новоствореного оголошення

Якщо ми захочемо орендувати авто, переходимо на сторінку оформлення заявки, то нам треба вказати період дат для оренди, повідомлення при потребі та підтвердити заявку (Рис. 4.9).

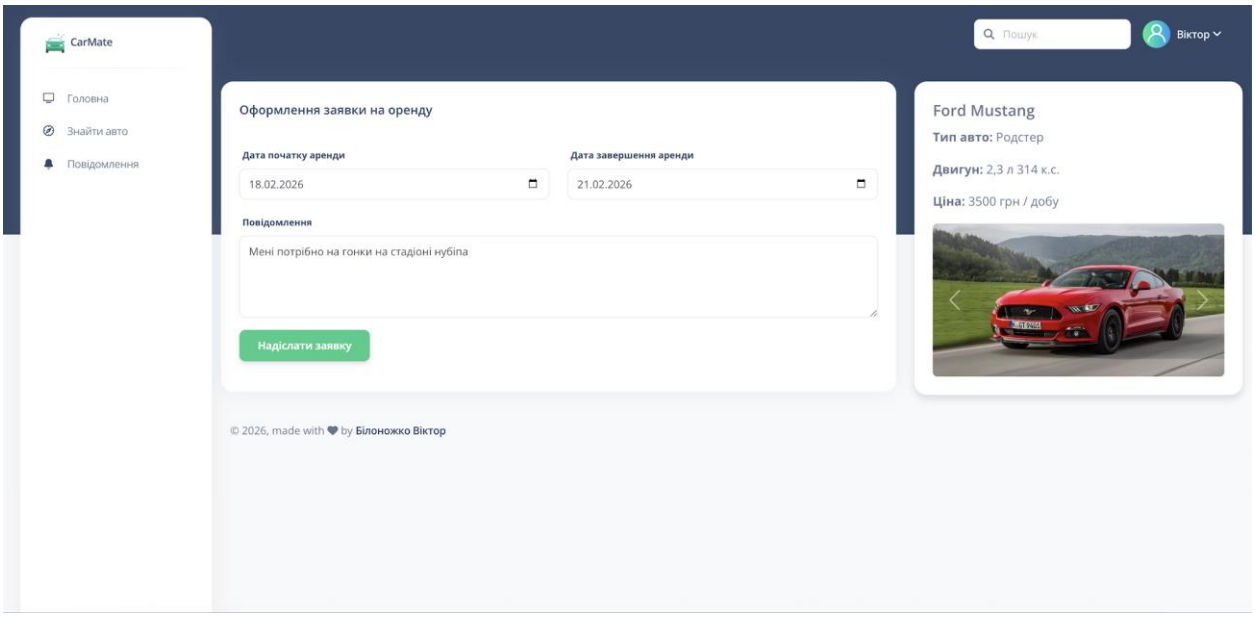

Рис. 4.9 Оформлення заявки на оренду

Переходимо на сторінку повідомлень, щоб побачити список усіх заявок на оренду наших оголошень від орендарів (Рис. 4.10).

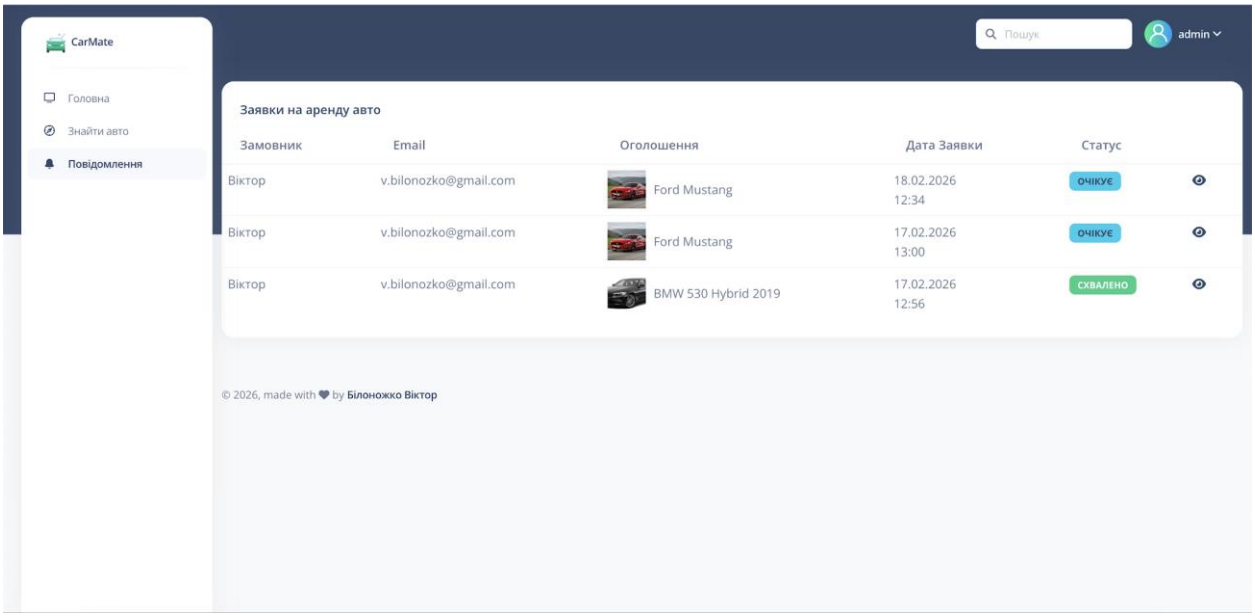

Рис. 4.10 Сторінка “Повідомлення”

Вибираємо якусь заявку та переходимо на сторінку обговорення. Тут у нас представлена вся інформація про оренду, а також чат для спілкування та переговорів між орендодавцем та орендарем. Власник оголошення може підтвердити або відмовити у бронюванні (Рис. 4.11).

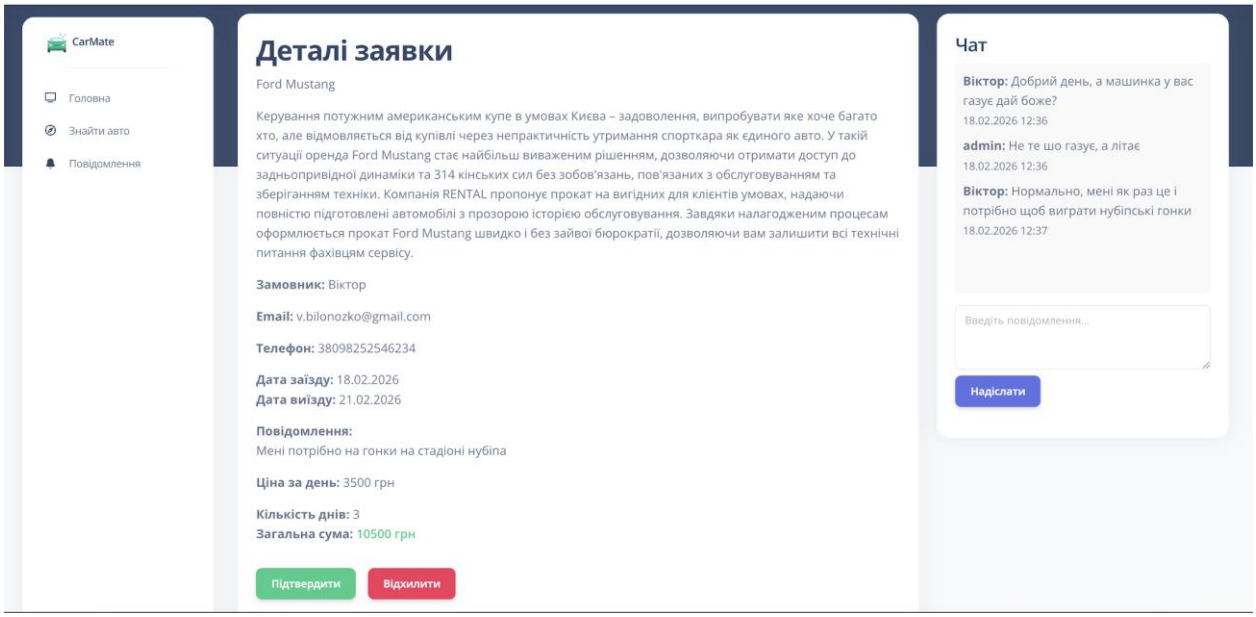

Рис. 4.11 Деталі заявки на бронювання

4.3 Склад інсталяційного пакету

Інсталяційний пакет розробленої веб-системи бронювання автомобілів містить повний набір програмних компонентів, конфігураційних файлів та службових ресурсів, необхідних для розгортання, налаштування та експлуатації системи у локальному або серверному середовищі. Загальний обсяг інсталяційного пакету становить близько 420 МБ без урахування кешу та тимчасових файлів.

Програмна система реалізована з використанням мови програмування PHP версії 8.2, фреймворку Symfony 6.4 та системи управління базами даних MySQL 8.0. Для контейнеризації та автоматизації процесу розгортання використовується Docker версії 26.0 та Docker Compose версії 2.24.

До складу інсталяційного пакету входять такі компоненти:

Вихідний код програмної системи

Вихідний код розміщується у каталозі /src та містить:

- контролери обробки HTTP-запитів;
- модулі бізнес-логіки;
- сутності Doctrine ORM;

- сервіси авторизації та обробки бронювань;
- модулі обробки повідомлень чату;
- конфігураційні файли Symfony.

Загальна кількість програмних файлів становить понад 250 одиниць, а сумарний обсяг вихідного коду перевищує 35 000 рядків.

Docker-конфігурація

Для автоматизованого розгортання системи використовуються файли:

- Dockerfile;
- docker-compose.yml.

Docker-конфігурація забезпечує запуск таких контейнерів:

- PHP-FPM 8.2;
- Nginx 1.27;
- MySQL 8.0;
- Redis 7.2 для кешування та обробки сесій.

Використання контейнеризації дозволяє стандартизувати середовище виконання та мінімізувати проблеми сумісності програмного забезпечення.

SQL-міграції бази даних

У каталозі /migrations розміщено SQL-міграції та Doctrine migration-файли для автоматичного створення структури бази даних. До складу входять:

- таблиці користувачів;
- таблиці оголошень про оренду;
- таблиці заявок на бронювання;
- таблиці повідомлень чату;
- таблиці зображень автомобілів.

Загальна кількість міграцій становить 18 файлів.

Конфігураційний файл середовища

Файл .env.example містить шаблон параметрів середовища:

- параметри підключення до бази даних;
- ключі безпеки;
- налаштування SMTP-сервера;
- конфігурацію Redis;
- URL веб-застосунку.

Використання окремого конфігураційного файлу забезпечує безпечне зберігання службових параметрів та спрощує перенесення системи між середовищами.

Інструкція запуску системи

До інсталяційного пакету входить файл README.md, який містить:

- вимоги до програмного середовища;
- порядок встановлення Docker;
- команди запуску контейнерів;
- процедуру створення бази даних;
- порядок виконання міграцій;
- інструкцію запуску frontend-збірки.

Інструкція містить приклади команд командного рядка та опис типових помилок під час розгортання.

Залежності Composer

У файлах composer.json та composer.lock описано серверні залежності проєкту. Основними бібліотеками є:

- Symfony Framework;
- Doctrine ORM;
- Twig;
- Symfony Security Bundle;
- PHPUnit.

Загальна кількість встановлених Composer-пакетів перевищує 70 бібліотек.

Frontend-залежності NPM

Для збірки клієнтської частини використовуються:

- Node.js 22;
- Vite;
- Vue.js 3;
- Bootstrap 5;
- Axios.

Залежності описані у файлах package.json та package-lock.json. Загальна кількість frontend-пакетів становить понад 120 модулів.

ВИСНОВКИ

Розробка системи бронювання автомобілів для служби прокату успішно вирішила основні цілі проєкту, забезпечивши функціональний, безпечний та масштабований веб-застосунок, який з'єднує постачальників послуг прокату з потенційними клієнтами. Система дозволяє користувачам реєструвати та керувати обліковими записами, публікувати та переглядати оголошення про автомобілі, подавати запити на прокат та спілкуватися через функцію обміну повідомленнями в режимі реального часу. Інтегруючи ці основні функції, застосунок спрощує процес оренди транспортних засобів, підвищуючи зручність як для постачальників, так і для орендарів.

Проєкт продемонстрував структурований підхід до розробки програмного забезпечення, починаючи з аналізу проблемної області, огляду існуючих рішень та специфікації функціональних та нефункціональних вимог. Етап проєктування включав створення UML-діаграм, таких як діаграми варіантів використання, послідовностей, дій, класів та пакетів, які керували впровадженням та забезпечували чітке розуміння структури системи та робочих процесів. Вибір MySQL як системи управління базами даних забезпечив надійне зберігання даних та підтримував цілісність посилань між користувачами, оголошеннями, запитами на прокат, повідомленнями та зображеннями.

Використання PHP та Symfony як бекенд-фреймворку в поєднанні з HTML, CSS, JavaScript та Bootstrap 5 для фронтенду призвело до створення адаптивного та зручного інтерфейсу. Docker було використано для забезпечення узгодженого середовища розробки та розгортання, тоді як Photoshop CS6 підтримував створення високоякісних візуальних елементів для інтерфейсу. Цей технологічний стек забезпечив зручність обслуговування, масштабованість та продуктивність у всій програмі.

У процесі виконання роботи було вирішено такі основні завдання:

- проведено аналіз предметної області та існуючих онлайн-сервісів прокату автомобілів;

- визначено функціональні та нефункціональні вимоги до програмної системи;
- спроєктовано архітектуру веб-застосунку та структуру бази даних;
- розроблено програмну систему з використанням PHP, Symfony та MySQL;
- реалізовано механізми реєстрації, авторизації та управління ролями користувачів;
- реалізовано функціонал створення та перегляду оголошень про оренду автомобілів;
- реалізовано систему подання заявок на бронювання;
- впроваджено чат для комунікації між користувачами;
- проведено функціональне та інтеграційне тестування системи.

На завершення, цей бакалаврський проєкт успішно завершив розробку комплексної системи бронювання автомобілів, яка демонструє ефективне застосування сучасних технологій веб-розробки, обґрунтованих архітектурних принципів та продуманого дизайну системи. Робота робить внесок у сферу онлайн-сервісів прокату, пропонуючи практичне рішення, яке можна надалі розширювати та адаптувати для задоволення потреб користувачів, що постійно змінюються.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Туссядія І., Песонен Дж. Рушійні сили прийняття послуг проживання між користувачами та каршерінгу. *Tourism Management*. 2016.
2. Бі Ю., Чен Х. Послуги прокату автомобілів: тенденції, бізнес-моделі та уподобання споживачів. *Journal of Transport and Logistics*. 2020.
3. Коу Г., Пен Ю., Ван Г. Онлайн-платформи прокату автомобілів: бізнес-аналіз та оптимізація послуг. *International Journal of Information Management*. 2019.
4. CarRental. URL: https://car-rent.ua/en/ (дата звернення: 22.05.2026).
5. Avis. URL: https://www.avis.com.ua/ (дата звернення: 22.05.2026).
6. Соммервіль І. Програмна інженерія. Pearson, 2016. 816 с.
7. Прессман Р. С. Програмна інженерія: підхід практикуючого спеціаліста. McGraw-Hill, 2019. 976 с.
8. Гамма Е., Гельм Р., Джонсон Р., Вліссідес Дж. Шаблони проєктування: елементи багаторазового об'єктно-орієнтованого програмного забезпечення. Addison-Wesley, 1995. 395 с.
9. Хоп Г., Вулф Б. Шаблони інтеграції підприємства: проєктування, створення та розгортання рішень для обміну повідомленнями. Addison-Wesley, 2012. 736 с.
10. Амблер С. Гнучкі методи баз даних: ефективні стратегії для гнучкого розробника програмного забезпечення. Wiley, 2003. 416 с.
11. PHP Manual. URL: https://www.php.net/manual/ (дата звернення: 22.02.2026).
12. Symfony Documentation. URL: https://symfony.com/doc/current/index.html (дата звернення: 21.02.2026).
13. MySQL Reference Manual. URL: https://dev.mysql.com/doc/ (дата звернення: 15.02.2026).

14. W3Schools HTML Tutorial. URL: https://www.w3schools.com/html/ (дата звернення: 16.01.2026).
15. W3Schools CSS Tutorial. URL: https://www.w3schools.com/css/ (дата звернення: 22.01.2026).
16. W3Schools JavaScript Tutorial. URL: https://www.w3schools.com/js/ (дата звернення: 12.03.2026).
17. Bootstrap 5 Documentation. URL: https://getbootstrap.com/docs/5.0/getting-started/introduction/ (дата звернення: 14.02.2026).
18. Docker Documentation. URL: https://docs.docker.com/ (дата звернення: 26.02.2026).
19. Doctrine ORM Documentation. URL: https://www.doctrine-project.org/projects/orm.html (дата звернення: 22.05.2026).
20. PHP: The Right Way. URL: https://www.phptherightway.com/ (дата звернення: 28.04.2026).
21. Stack Overflow. URL: https://stackoverflow.com/ (дата звернення: 11.05.2026).
22. GeeksforGeeks PHP Tutorials. URL: https://www.geeksforgeeks.org/php/ (дата звернення: 15.04.2026).
23. SitePoint PHP Articles. URL: https://www.sitepoint.com/php/ (дата звернення: 05.02.2026).
24. MySQL Performance Blog. URL: https://www.percona.com/blog/ (дата звернення: 18.01.2026).
25. SymfonyCasts Tutorials. URL: https://symfonycasts.com/ (дата звернення: 23.04.2026).
26. Mozilla Developer Network (MDN) JavaScript Guide. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 12.03.2026).
27. Mozilla Developer Network (MDN) CSS Guide. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 13.03.2026).

28. PHP Symfony Best Practices. URL: https://symfony.com/doc/current/best_practices.html (дата звернення: 05.03.2026).
29. Laravel News. URL: https://laravel-news.com/ (дата звернення: 06.04.2026).
30. DigitalOcean Tutorials (MySQL, PHP). URL: https://www.digitalocean.com/community/tutorials (дата звернення: 22.03.2026).
31. Tutorialspoint PHP Tutorial. URL: https://www.tutorialspoint.com/php/index.htm (дата звернення: 15.02.2026).
32. Tutorialspoint MySQL Tutorial. URL: https://www.tutorialspoint.com/mysql/index.htm (дата звернення: 22.03.2026).
33. FreeCodeCamp PHP Tutorial. URL: https://www.freecodecamp.org/news/tag/php/ (дата звернення: 25.03.2026).
34. FreeCodeCamp MySQL Tutorial. URL: https://www.freecodecamp.org/news/tag/mysql/ (дата звернення: 09.02.2026).
35. Envato Tuts+ Bootstrap Tutorials. URL: https://webdesign.tutsplus.com/categories/bootstrap (дата звернення: 08.01.2026).

ДОДАТОК А

Фрагменти програмного коду. Функція створення оголошення

```
class RentController extends AbstractController
{
    #[Route('/rents', name: 'app_rent')]
    public function index(Request $request, EntityManagerInterface $entityManager): Response
    {
        $query = $request->query->get('query');
        $listingsQuery = $entityManager->getRepository(Listing::class)->createQueryBuilder('l')
            ->orderBy('l.createdAt', 'DESC');

        if ($query) {
            $listingsQuery->andWhere('l.title LIKE :query OR l.description LIKE :query')
                ->setParameter('query', '%'.$query.'%');
        }

        $listings = $listingsQuery->getQuery()->getResult();

        return $this->render('rent/index.html.twig', [
            'listings' => $listings,
        ]);
    }

    #[Route('/rents/create', name: 'app_rent_create')]
    public function create(Request $request, EntityManagerInterface $entityManager,
SluggerInterface $slugger): Response
    {
        if ($request->isMethod('POST')) {
            $listing = new Listing();
            $listing->setTitle($request->request->get('title'));
            $listing->setDescription($request->request->get('description'));
            $listing->setCarType($request->request->get('car_type'));
            $listing->setEngine($request->request->get('engine'));
            $listing->setPricePerDay((float) $request->request->get('pricePerDay'));
            $listing->setAvailableFrom(new \DateTime($request->request->get('availableFrom')));
            $listing->setAvailableTo(new \DateTime($request->request->get('availableTo')));
            $listing->setCreatedAt(new \DateTimeImmutable());
            $listing->setStatus(Listing::LISTING_ACTIVE);
            $listing->setOwner($this->getUser());

            $entityManager->persist($listing);
            $entityManager->flush();
```

```
        $imageFiles = $request->files->get('images');
        if ($imageFiles) {
            foreach ($imageFiles as $imageFile) {
                if ($imageFile->isValid()) {
                    $originalFilename = pathinfo($imageFile->getClientOriginalName(),
PATHINFO_FILENAME);
                    $safeFilename = $slugger->slug($originalFilename);
                    $newFilename = $safeFilename.'-'.uniqid('', true).'.'.$imageFile-
>guessExtension();

                    try {
                        $imageFile->move(
                            $this->getParameter('listing_images_directory'),
                            $newFilename
                        );

                        $listingImage = new ListingImages();
                        $listingImage->setListing($listing);
                        $listingImage->setImagePath($newFilename);
                        $listingImage->setCreatedAt(new \DateTimeImmutable());
                        $entityManager->persist($listingImage);
                    } catch (FileException $e) {
                        flash()->error('Не вдалося завантажити зображення: '.$e->getMessage());
                    }
                }
            }
            $entityManager->flush();
        }

        flash()->success('Оголошення успішно створено');
        return $this->redirectToRoute('app_rent');
    }

    return $this->render('rent/create.html.twig');
}

#[Route('/rents/{id}', name: 'app_rent_show')]
public function show(int $id, EntityManagerInterface $entityManager): Response
{
    $listing = $entityManager->getRepository(Listing::class)->find($id);

    if (!$listing) {
        throw $this->createNotFoundException('Оголошення не знайдено');
    }
```

```
        $images = $listing->getImages();

        return $this->render('rent/show.html.twig', [
            'listing' => $listing,
            'images' => $images,
        ]);
    }
}
```

ДОДАТОК Б

Фрагменти програмного коду. Функціонал бронювання авто

```
class RequestController extends AbstractController
{
    #[Route('/notifications', name: 'app_notifications')]
    public function index(EntityManagerInterface $entityManager): Response
    {
        $user = $this->getUser();

        $requests = $entityManager->getRepository(RentalRequest::class)
            ->createQueryBuilder('r')
            ->join('r.listing', 'l')
            ->leftJoin('l.images', 'i')
            ->leftJoin('r.user', 'c')
            ->addSelect('l', 'i', 'c')
            ->where('l.owner = :user')
            ->setParameter('user', $user)
            ->orderBy('r.createdAt', 'DESC')
            ->getQuery()
            ->getResult();

        return $this->render('notifications/index.html.twig', [
            'requests' => $requests,
        ]);
    }

    #[Route('/notifications/{id}', name: 'app_notification_show')]
    public function show(RentalRequest $rentalRequest, Request $request, MessageRepository
$messageRepository): Response
    {
        $listing = $rentalRequest->getListing();
        $user = $rentalRequest->getUser();

        $start = $rentalRequest->getStartDate();
        $end = $rentalRequest->getEndDate();

        $days = $start->diff($end)->days;
        $totalPrice = $listing->getPricePerDay() * $days;

        $messages = $messageRepository->findBy(['rentalRequest' => $rentalRequest], ['createdAt'
=> 'ASC']);

        $isOwner = $this->getUser() === $listing->getOwner();
```

```
        return $this->render('notifications/show.html.twig', [
            'request' => $rentalRequest,
            'listing' => $listing,
            'client' => $user,
            'days' => $days,
            'totalPrice' => $totalPrice,
            'messages' => $messages,
            'isOwner' => $isOwner,
        ]);
    }

    #[Route('/notifications/{id}/approve', name: 'app_notification_approve', methods: ['POST'])]
    public function approve(RentalRequest $rentalRequest, Listing $listing,
EntityManagerInterface $entityManager): Response
    {
        $rentalRequest->setStatus(RentalRequest::RENTAL_APPROVED);
        $listing->setStatus(Listing::LISTING_RENTED);
        $entityManager->flush();

        flash()->success('Заявку підтверджено');
        return $this->redirectToRoute('app_notification_show', ['id' => $rentalRequest->getId()]);
    }

    #[Route('/notifications/{id}/reject', name: 'app_notification_reject', methods: ['POST'])]
    public function reject(RentalRequest $rentalRequest, EntityManagerInterface
$entityManager): Response
    {
        $rentalRequest->setStatus(RentalRequest::RENTAL_REJECTED);
        $entityManager->flush();

        flash()->error('Заявку відхилено');
        return $this->redirectToRoute('app_notification_show', ['id' => $rentalRequest->getId()]);
    }
}
```

Додаток В

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій Декларація про академічну доброчесність та використання ШІ

Я, Білоножко Віктор Станіславович, здобувач НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота на тему «Розробка системи бронювання автомобілів у сервісі прокату» є результатом моєї особистої праці.

2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.

3. Щодо використання інструментів ШІ зазначаю, що:

інструменти ШІ Gemini були використані для:

- + перекладу іншомовних джерел;
- + стилістичного редагування та перевірки граматики;
- + допомоги у написанні/відлагодженні програмного коду;
- + інше: пошуку додаткових інструментів у розробці програмного продукту.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«22» Травня 2026 р. _____ **Віктор БІЛОНОЖКО**