НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

Комп'ютерних наук

_____ Голуб Б.Л.

"___" _____ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему

«Програмне забезпечення мобільного додатку для орендарів і власників житла з підтримкою рейтингової системи»

Спеціальність 121 «Інженерія програмного забезпечення»

Гарант освітньої програми

к.т.н, доцен _____ Вайганг Г.О.
(Науковий ступень та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

_____ Ніколаєнко Д.В.
(Науковий ступень та вчене звання) (підпис) (ПІБ)

Виконав _____ Сілютін В.В.
(підпис) (ПІБ)

КИЇВ-2025

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
Комп'ютерних наук
_____Голуб Б.Л.
_“ ”_____20 р._

З А В Д А Н Н Я

на виконання бакалаврської кваліфікаційної роботи студенту

Сілютіну Владиславу Віталійовичу

Спеціальність 121 – «Інженерія програмного забезпечення»

Тема бакалаврської кваліфікаційної роботи: «програмне забезпечення мобільного додатку для орендарів і власників житла з підтримкою рейтингової системи»

Затверджена наказом ректора НУБіП України від 16.12.2024 № 2248 «С».

Термін подання завершеної роботи на кафедру _____

Вихідні дані до бакалаврської кваліфікаційної роботи: документація бібліотек, нормативні та технічні документи

Перелік питань , які потрібно розробити: проаналізувати предметну область, аналоги, розробити вимоги до системи, спроектувати та розробити систему

Дата видачі завдання “_____”_____20___ р.

Керівник бакалаврської кваліфікаційної роботи

_____ _____ ___ Ніколаєнко Д.В.

(науковий ступінь та вчене звання) (підпис) (ПІБ)

Завдання прийняв до виконання _____ Сілютін В.В.

(підпис) (ПІБ студента)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....5

ВСТУП..... 6

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....9

1.1. Опис предметної області..... 9

1.2. Аналіз вимог до програмної системи..... 10

1.3. Моделювання предметної області..... 12

1.4. Огляд інформаційних джерел та існуючих рішень.....17

1.4.1. Аналоги та існуючі рішення..... 17

1.4.2. Відмінності запропонованого рішення..... 21

1.5. Постановка завдання.....23

1.6. Висновки до розділу 1.....24

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....25

2.1. Логічна модель даних у вигляді ER-діаграми..... 25

2.2. Діаграма класів та кооперацій.....28

2.3. Діаграма пакетів..... 32

2.4. Діаграма компонентів..... 35

2.5. Висновок до розділу 2.....37

3 РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....38

3.1. Система управління інформаційною базою.....38

3.2. Розробка інформаційної бази..... 40

3.3. Вибір інструментарію для створення прикладного програмного забезпечення..... 44

3.4. Алгоритмізація та програмування програмних модулів..... 47

3.5. Висновки до розділу 3.....57

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУТАЦІЇ СИСТЕМИ..... 59

4.1. Тестування системи.....59

4.2. Вимоги до апаратного та програмного забезпечення.....66

4.3. Склад інсталяційного пакету.....69

4.4. Висновки до розділу 4.....72

ВИСНОВКИ.....73

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 75

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API — Інтерфейс програмування застосунків (Application Programming Interface)

Auth — Механізм автентифікації та авторизації користувача

БД — База даних

Blade — Шаблонізатор у фреймворку Laravel для створення веб-інтерфейсів

Bootstrap — Фреймворк CSS для адаптивної верстки та побудови інтерфейсів

CRUD — Create, Read, Update, Delete — основні операції над даними

CSS — Cascading Style Sheets, каскадні таблиці стилів для оформлення сторінок

HTML — HyperText Markup Language, мова розмітки гіпертексту

HTTP — HyperText Transfer Protocol, протокол передавання гіпертексту

JS — JavaScript, мова програмування для інтерактивної поведінки сторінок

JSON — JavaScript Object Notation, формат обміну структурованими даними

Laravel — PHP-фреймворк для розробки веб-застосунків

MVC — Model-View-Controller, архітектурна модель програмного забезпечення

MySQL — Система управління реляційними базами даних

PWA — Progressive Web Application, прогресивний веб-додаток з можливістю встановлення

PHP — Мова серверного програмування, що використовується в Laravel

PZ — Програмне забезпечення

SQL — Structured Query Language, мова запитів до бази даних

Service Worker — Сценарій у браузері для офлайн-роботи та кешування PWA

UI — User Interface, інтерфейс користувача

UX — User Experience, досвід користувача

Token — Токен, цифровий маркер доступу користувача

URL — Uniform Resource Locator, уніфікований ідентифікатор ресурсу

UX/UI — Спільне позначення дизайну інтерфейсу та зручності використання

ВСТУП

У сучасному світі мобільні технології стали невід’ємною частиною нашого повсякденного життя. Це особливо відчутно у сфері оренди житла, де люди дедалі частіше шукають прості, швидкі та зручні способи знайти чи здати помешкання. Суттєву роль у цьому відіграють мобільні додатки — особливо ті, що працюють без потреби щось встановлювати з магазину додатків.

Часто сервіси оренди житла мають обмежений функціонал, незручні для використання на смартфонах або потребують додаткового ПЗ. Тому з’являється потреба у створенні сучасного прогресивного вебдодатку (PWA), який працюватиме швидко, легко відкриватиметься з будь-якого пристрою та не вимагатиме встановлення.

(PWA) – це поєднання вебсторінки та мобіль-ного додатка; подібно до власних програм PWAможуть працювати в автономному режимі танадсилати push-сповіщення; але, на відмінувід власних програм, PWA можна зв’язувати, їхможна використовувати на різних платформах(для Android та iOS не потрібні окремі кодовібази), і вони не вимагають участі в магазинахдодатків; гібридне розроблення додатків такожохоплює елементи власних та вебпрограм[1].

PWA-додаток, який розробляється в рамках цього проєкту, передбачає:

- зручний пошук житла з можливістю фільтрації;
- систему оцінок і рейтингів користувачів;
- перегляд історії взаємодій;
- запуск прямо з головного екрану смартфона без встановлення з маркету;
- сортування оголошень за категоріями (тип, локація, ціна тощо);
- розширений функціонал пошуку;
- підтримку ролей з багаторівневою автентифікацією.

Метою дослідження кваліфікаційної роботи є розробка мобільного додатку у вигляді прогресивного вебзастосунку (PWA), який забезпечує повний цикл взаємодії між орендарями та орендодавцями, включаючи систему рейтингів, пошуку, а також функції адміністративного керування.

Об’єктом дослідження дипломної роботи є є процес взаємодії користувачів в цифровому середовищі у сфері оренди житла.

Предметом дослідження є розробка адаптивного мобільного додатку з підтримкою PWA та розширеним функціоналом взаємодії, пошуку і управління.

Завдання дослідження:

1. Виконати всебічний аналіз сфери оренди житла, визначивши її особливості та ключові аспекти.
2. Оцінити існуючі рішення на ринку та окреслити основні вимоги до створення майбутнього програмного продукту.
3. Розробити концепцію архітектури інформаційної системи, яка враховуватиме підтримку різних типів користувацьких ролей. Реалізувати мобільний додаток із використанням технологій Laravel, HTML, CSS, JavaScript, MySQL і Bootstrap.
4. Інтегрувати підтримку прогресивного веб-додатку (PWA), забезпечивши функції web-manifest, service worker і режиму роботи офлайн.
5. Додати інструменти для надсилання сповіщень, налаштування фільтрів, відстеження рейтингів і зберігання історії взаємодій.
6. Організувати тестування програмного продукту та розробити рекомендації щодо його впровадження.

Під час розробки використовувався сучасний стек вебтехнологій, орієнтований на адаптивність, безпеку та зручність. Завдяки впровадженню PWA-додатку, користувачі отримують миттєвий доступ до сервісу з будь-

якого пристрою, можливість запуску з головного екрану смартфона та навіть роботу без інтернету.

Розроблене рішення має практичну користь як для орендарів, так і для власників житла. Воно полегшує взаємодію, підвищує рівень довіри завдяки вбудованій системі рейтингів і створене з урахуванням потреб мобільного користувача.

У цій бакалаврській роботі описано весь шлях створення інформаційної системи — від аналізу проблем до реалізації готового функціонального продукту, який відповідає сучасним стандартам та може бути впроваджений на практиці.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Опис предметної області

У наш час оренда житла стає дедалі популярнішою — особливо серед молоді, студентів та людей, які часто змінюють місце проживання через роботу або особисті обставини. Водночас власники квартир також шукають надійних орендарів, з якими можна встановити довірливі стосунки. Проте користування сучасними онлайн-сервісами нерідко викликає труднощі: більшість платформ або погано працює на мобільних пристроях, або ж вимагає встановлення через App Store чи Google Play, що може бути незручним для багатьох користувачів.

Крім технічних обмежень, у сфері оренди житла існує низка інших проблем. Наприклад, часто ускладнена комунікація між орендодавцями та орендарями, бракує прозорої системи оцінювання співпраці, а також немає зручного способу переглядати історію взаємодій між сторонами. Ще одна поширена проблема — обмежений функціонал пошуку: більшість сервісів дозволяють фільтрувати житло лише за базовими параметрами, не даючи можливості гнучко налаштувати пошук, скажімо, за типом санвузла, кількістю кімнат чи ціновим діапазоном.

Саме тому виникає нагальна потреба у створенні сучасного мобільного застосунку у форматі прогресивного вебзастосунку (PWA). Такий додаток відкривається напряму з браузера, не потребує встановлення, працює швидко, має адаптивний дизайн, підтримує офлайн-режим і зберігає дані локально. До того ж користувачі можуть додати його на головний екран свого смартфона, як звичайний застосунок.

Функціональність додатку повинна охоплювати:

- зручне створення і публікацію оголошень про оренду житла;
- гнучкий пошук із можливістю фільтрування за різними критеріями;
- систему рейтингу, що дозволяє оцінювати орендарів і орендодавців після завершення співпраці;

- збереження історії всіх контактів і взаємодій;
- категоризацію об'єктів для легшої навігації;
- вбудований чат для швидкого спілкування між користувачами.

Окрему увагу слід приділити ролям у системі. Адміністратор, орендодавець і орендар повинні мати власні функції та інтерфейс, що відповідає їхнім потребам і рівню доступу. Такий підхід зробить використання платформи зручним, безпечним і логічно структурованим.

Отже, створення адаптивного та доступного мобільного застосунку у форматі PWA допоможе суттєво покращити досвід оренди житла, зробити його прозорішим, зручнішим і доступним для широкого кола користувачів.

1.2. Аналіз вимог до програмної системи

Після вивчення предметної області було визначено низку ключових вимог до майбутнього мобільного застосунку, які охоплюють як функціональність, так і технічні аспекти.

Функціональні вимоги:

1. Реєстрація та авторизація користувачів із підтримкою ролей: адміністратор, орендодавець, орендар.
2. Можливість створення оголошень з детальним описом житла (кімнати, площа, санвузол, ціна тощо).
3. Пошук житла з розширеними фільтрами — за типом, кількістю кімнат, вартістю, площею, районом тощо.
4. Вбудований чат для комунікації між користувачами.
5. Рейтингова система, що дозволяє залишати оцінки після завершення оренди.
6. Збереження історії контактів та взаємодій у профілі користувача.
7. Автоматична класифікація житла за категоріями.
8. Адаптивний інтерфейс, зручний для мобільних пристроїв.

9. Повна підтримка PWA: офлайн-доступ, швидкий запуск з головного екрана, кешування даних.
10. Адмін-панель для керування користувачами, статистикою і модерацією оголошень

Нефункціональні вимоги:

1. Висока швидкодія: забезпечення швидкого завантаження сторінок без значних затримок.
2. Надійність: захищене середовище для зберігання персональних даних користувачів.
3. Масштабованість: можливість розширення системи відповідно до потреб користувачів.
4. Кросплатформена підтримка для мобільних браузерів, планшетів і персональних комп’ютерів.
5. Забезпечення безпеки: впровадження CSRF-захисту, контроль доступу та хешування паролів.
6. Підтримка автономної роботи (offline-first), що передбачає доступ до раніше завантажених даних навіть без з’єднання з Інтернетом.
7. Інтуїтивний і зрозумілий інтерфейс з продуманою навігацією та оптимізацією під невеликі екрани пристроїв.

Технічні вимоги:

Мова програмування: PHP

Фреймворк: Laravel

Шаблони: Blade

База даних: MySQL

Фронтенд: HTML5, CSS3, Bootstrap, JavaScript

PWA-компоненти:

- manifest.json — налаштування іконки, назви та поведінки застосунку;
- service worker — кешування, робота офлайн;

Серверне середовище: локальний сервер (наприклад, XAMPP або Laravel Sail)

1.3. Моделювання предметної області

Моделювання предметної області — це дуже важливий етап у розробці програмного забезпечення. Воно допомагає краще зрозуміти, як повинна працювати майбутня система, і дає змогу розкласти все по поличках. Завдяки цьому можна заздалегідь виявити можливі проблеми, правильно визначити вимоги і створити основу для наступних кроків — проектування і програмування.

Для цього часто використовують UML — спеціальну мову, яку придумали ще в 90-х роках, коли об’єднували різні підходи до побудови систем. Сьогодні UML — це стандарт у світі програмістів і аналітиків. Вона дозволяє будувати різні типи діаграм, які показують, як саме працює система.

Основні типи діаграм:

- Use Case (варіанти використання) — показують, що саме може робити користувач у системі.
- Class (класи) — описують об’єкти в системі, їхні властивості, функції та зв’язки.
- Sequence (послідовності) — показують, у якій послідовності об’єкти взаємодіють між собою.
- Activity (діяльності) — зображають, як відбувається певний процес або потік даних.
- State (станів) — показують, як змінюється стан об’єкта протягом його "життя" в системі.

Щоб побудувати модель для Rent Trust, спершу потрібно визначити головні сутності (тобто основні поняття або об’єкти системи). На рисунку 1.1 зображені ці ключові елементи.

Рис. 1.1. Основні абстракції системи

Користувач — це базова сутність. Вона включає ID, ім’я, email, роль (орендар або орендодавець), дату реєстрації та рейтинг.

Від неї походять дві ролі:

- Орендар — може шукати житло, залишати відгуки тощо.
- Орендодавець — може створювати і керувати оголошеннями.

Оголошення — включає назву, ціну, опис, статус, дату створення, фото і дані про того, хто розмістив це оголошення. Один користувач може створити кілька оголошень.

Оцінка — містить ID, оцінку, коментар, дату, інформацію про того, хто залишив оцінку, і про того, кому її поставили. Вона потрібна для підрахунку рейтингу користувачів.

PWAФункції — описують технічні можливості мобільної версії: робота без інтернету, сповіщення, кешування тощо.

Заявка — пов’язана з оголошенням і користувачем. Включає дату, статус, коментар, ID орендаря і ID оголошення. Вона показує, хто і коли подав запит на оренду.

Діаграма варіантів використання (Use Case Diagram) – діаграма, на якій відображенавзаємодія певної сутності (діючої особи, актора) і системи, що моделюється. Перелік всіхваріантів використання фактично визначає функціональні вимоги до системи [2, с. 76].

У моделі передбачено три основні актори: орендар, орендодавець та адміністратор.

У системі є три головні ролі:

Орендар — може зареєструватися, увійти в додаток, шукати житло з фільтрами (район, ціна, площа), переглядати деталі, додавати в обране, залишати відгуки і оцінки. Також він може переглядати все навіть без інтернету — завдяки підтримці PWA.

Орендодавець — також входить у систему, створює, редагує та видаляє оголошення, переглядає заявки, ставить оцінки орендарям і бачить статистику переглядів.

Адміністратор — перевіряє нові оголошення, розглядає скарги, блокує порушників і має доступ до даних про користувачів і їхню активність.

Рис. 1.2. Діаграма прецедентів системи

На рисунку 1.3 представлено діаграму послідовності, яка демонструє типовий сценарій взаємодії користувача з мобільним додатком системи «Rent Trust», враховуючи роботу як в онлайн-, так і в офлайн-режимах.

Сценарій починається з того, що користувач, наприклад орендар, відкриває додаток на своєму мобільному пристрої. Інтерфейс ініціює запит до сервера, щоб перевірити наявність інтернет-з’єднання та завантажити базові дані. Якщо з’єднання доступне, виконується авторизація: сервер звіряє облікові дані користувача через базу даних і повертає токен для доступу.

Після успішної авторизації користувач здійснює пошук житла. Інтерфейс програми надсилає серверу запит із вказаними параметрами для фільтрації. Сервер обробляє цей запит і повертає список результатів, які відображаються на екрані користувача. Отримані дані кешуються у локальну

пам’ять за допомогою Service Worker, що дозволяє відкривати ті самі оголошення в офлайн-режимі у майбутньому.

У разі натискання користувачем кнопки «Додати до обраного», додаток надсилає серверу відповідний запит. Якщо інтернет-з’єднання відсутнє, виконана дія тимчасово записується в локальний кеш для подальшої синхронізації після відновлення мережі.

Під час встановлення PWA-клієнт запитує у користувача дозвіл на створення ярлика додатка на головному екрані, збереження файлу manifest.json і реєстрацію сервіс-воркера. Після виконання цих кроків додаток функціонує як нативний навіть за відсутності доступу до мережі.

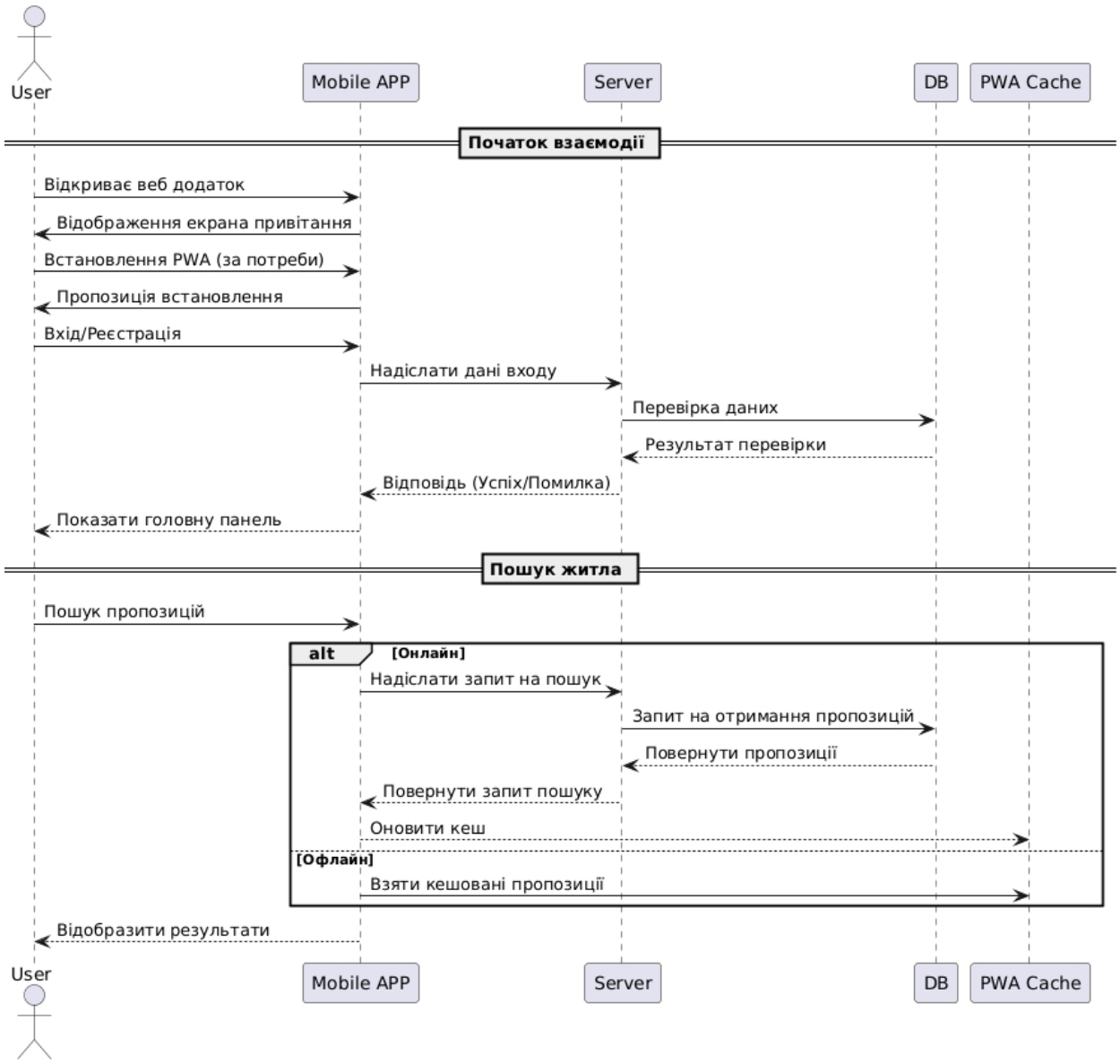


Рис. 1.3. Діаграма послідовності до проектованої системи

1.4. Огляд інформаційних джерел та існуючих рішень

Для успішного проєктування мобільного застосунку "Rent Trust" було виконано детальний аналіз наукових джерел, методичних матеріалів,

технічної документації та проведено огляд існуючих цифрових рішень у сфері платформ для оренди житла.

Ключовими джерелами інформації стали:

1. Наукова література: навчальні посібники з проєктування інформаційних систем, матеріали щодо створення UML-діаграм з описом методів структурного та об’єктно-орієнтованого аналізу прикладного програмного забезпечення.

2. Методичні вказівки університетів, які висвітлюють практичні аспекти побудови діаграм прецедентів, послідовностей, класів та інших елементів інформаційних систем.

3. Офіційна технічна документація інструментів і технологій, застосованих у розробці проєкту, зокрема:

- Laravel
- Bootstrap
- PWA
- MySQL
- Додаткова документація Blade та Service Worker API.

Окрім цього, було здійснено оцінку функціоналу й структури аналогічних сервісів для оренди житла, вже присутніх на ринку.

1.4.1. Аналоги та існуючі рішення

На сучасному ринку оренди житла функціонує кілька популярних платформ, які забезпечують часткову або повну реалізацію таких функцій, як пошук житла, публікація оголошень, зворотний зв’язок із користувачами, система рейтингів та зручний мобільний доступ.

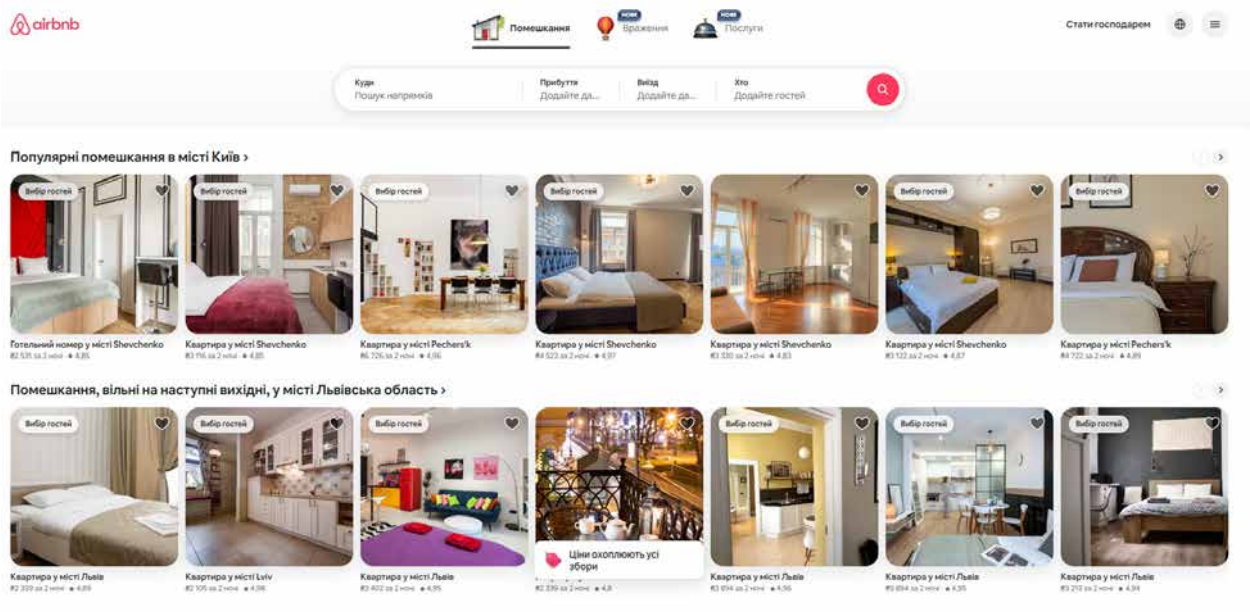


Рис. 1.4. Скріншот роботи сервісу «Airbnb»

Airbnb є однією з найпопулярніших платформ для короткострокової оренди житла. Вона надає користувачам зручний мобільний інтерфейс, систему відгуків, можливість бронювання та підтримки. Однак варто зазначити, що сервіс не реалізований у форматі прогресивного веб-застосунку (PWA), тому потребує стабільного інтернет-з'єднання.

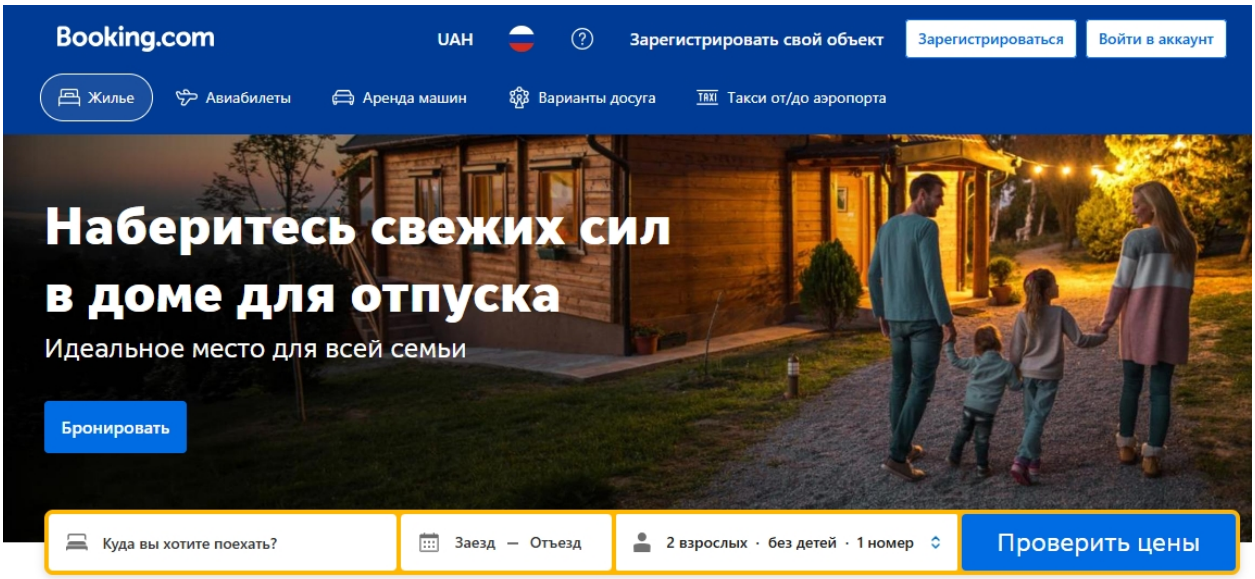


Рис. 1.5. Скріншот роботи сервісу «Booking.com»

Booking.com є впливовою платформою для онлайн-бронювання, що має інтегровану мобільну версію. Сервіс надає можливість резервування житла,

забезпечує комунікацію між орендодавцями та орендарями через систему повідомлень, а також реалізує процедури перевірки користувачів. Водночас платформа не передбачає відкриту систему рейтингу для оцінювання орендарів.

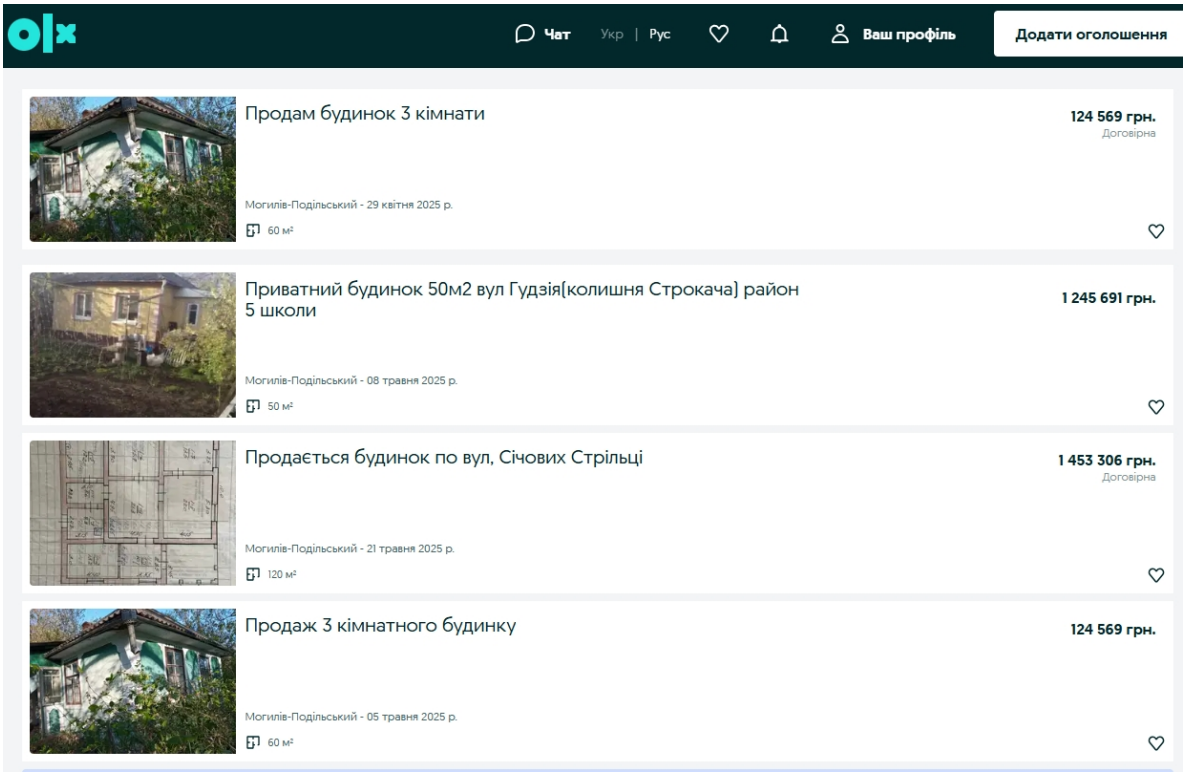


Рис. 1.6. Скріншот роботи сервісу «OLX»

OLX – онлайн-дошка оголошень, де користувачі можуть публікувати та знаходити пропозиції оренди. Інтерфейс менш структурований, відсутня вбудована система відгуків і рейтингу, немає прогресивної мобільної версії.

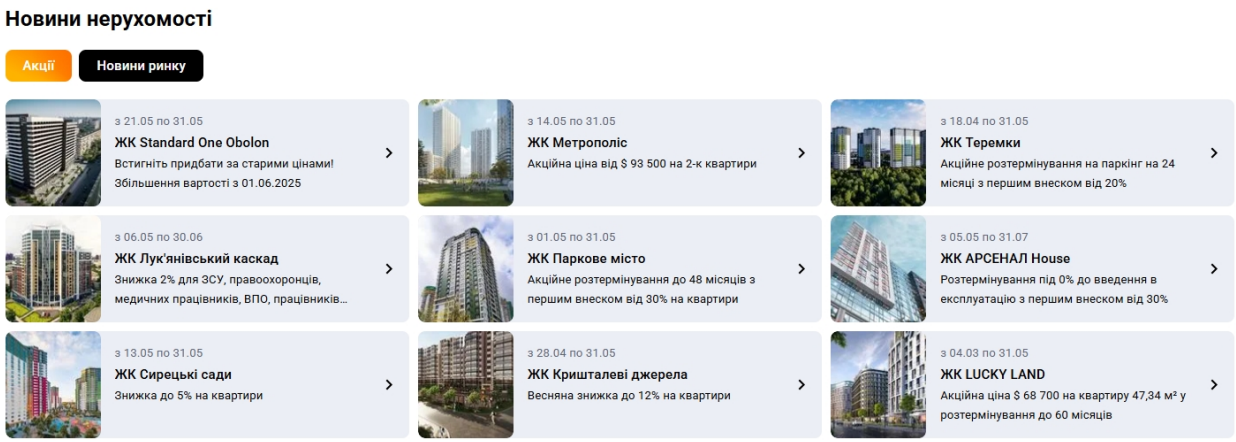


Рис. 1.7. Скріншот роботи сервісу «Lun.ua»

Lun.ua – спеціалізований сервіс з пошуку новобудов та житла в оренду. Має привабливий UX, фільтри, карту об’єктів. Проте не підтримує персоналізовані ролі, оцінки користувачів, а функції працюють лише онлайн.

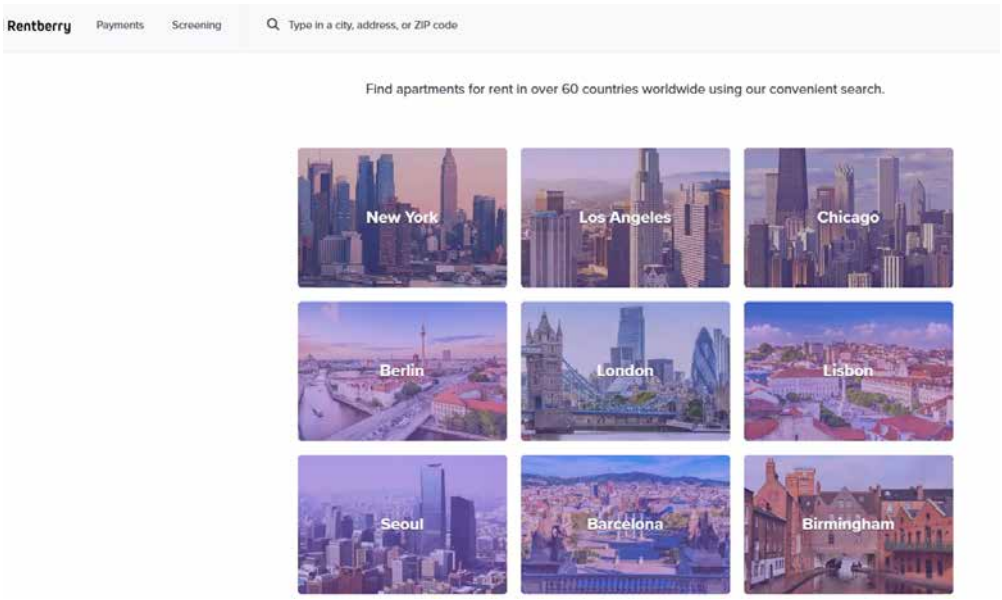


Рис. 1.8. Скріншот роботи сервісу «Rentberry»

Rentberry – менш відомий, але орієнтований саме на ринок оренди житла. Має систему ставок, рейтингів, підтримку мобільного інтерфейсу. Частково реалізує PWA-функціональність, але потребує реєстрації з підтвердженням через банківські дані.

Таблиця 1.1 – Порівняння аналогів інформаційної системи Rent Trust

Критерій / Система	**Rent Trust**	**Airbnb**	**Booking.com**	**OLX Нерухомість**	**Lun.ua**	**Rentberry**
Система рейтингів для обох сторін	Так	Так	Частково	Ні	Ні	Так
Офлайн-режим (PWA)	Так	Ні	Ні	Ні	Ні	Частково
Адаптивний мобільний інтерфейс	Так	Так	Так	Частково	Так	Так
Підтримка ролей (орендодавець/орендар)	Так	Так	Так	Ні	Ні	Так
Реєстрація та модерація оголошень	Так	Так	Так	Частково	Так	Так
Можливість залишати відгуки	Так	Так	Так	Ні	Ні	Так

Критерій / Система	Rent Trust	Airbnb	Booking.com	OLX Нерухомість	Lun.ua	Rentberry
Встановлення на головний екран (PWA)	Так	Ні	Ні	Ні	Ні	Частково

1.4.2. Відмінності запропонованого рішення

У рамках цієї дипломної роботи розробляється мобільний додаток Rent Trust, який відрізняється від існуючих аналогів рядом суттєвих особливостей:

- він функціонує як прогресивний веб-додаток (PWA), що дозволяє користуватися ним подібно до традиційних мобільних додатків без необхідності завантаження з офіційних магазинів;
- орієнтується виключно на ринок оренди житла, на відміну від універсальних платформ, що обслуговують широкий спектр нерухомості;
- запроваджує двосторонню рейтингову систему, яка дозволяє як орендарям, так і орендодавцям оцінювати один одного, підвищуючи рівень прозорості та довіри;
- пропонує адаптивний інтерфейс із темною темою, що оптимізує використання на мобільних пристроях;

- забезпечує чіткий поділ ролей користувачів (орендар, орендодавець, адміністратор), з унікальним функціоналом для кожної групи;
- використовує інтуїтивну систему фільтрації та пошуку оголошень, яка враховує регіональні особливості, тип житла, цінові параметри тощо;
- передбачає автоматизоване створення PDF-карток об'єктів оренди для зручного перегляду або друку;
- надає можливість залишати відгуки, переглядати історію взаємодій та рейтинги інших користувачів;

- гарантує відсутність платних обмежень у базовій функціональності — усі критично важливі інструменти доступні безкоштовно.

Таким чином, дане рішення заповнює нішу, яка існує між традиційними платформами оголошень і складними CRM-системами для агентств нерухомості, забезпечуючи простоту, надійність та довіру в орендних операціях для всіх учасників процесу. F

1.5. Постановка завдання

Метою цієї дипломної роботи є розробка мобільного вебзастосунку Rent Trust у форматі прогресивного вебдодатку (PWA), орієнтованого на безпечну взаємодію між орендарями та орендодавцями, із вбудованою рейтинговою системою і адаптацією для мобільних пристроїв.

Для реалізації цієї мети передбачено виконання наступних завдань:

розробити сучасний адаптивний інтерфейс користувача у темній темі з орієнтацією на мобільні пристрої;

реалізувати систему реєстрації та автентифікації користувачів із поділом на ролі: орендар, орендодавець, адміністратор;

створити особисті кабінети з можливістю додавання/редагування оголошень про оренду житла;

реалізувати систему перегляду доступних об'єктів із фільтрами за районами, ціною, типом житла;

забезпечити функцію залишення рейтингу та відгуків після завершення оренди;

розробити адміністративний модуль для керування користувачами, оголошеннями та перегляду статистики;

налаштувати offline-режим, кешування даних та можливість встановлення застосунку на домашній екран (через manifest.json та service worker);

реалізувати можливість експорту оголошень у PDF-файл;

забезпечити повну підтримку української мови;

протестувати та забезпечити безперебійну роботу на різних типах мобільних пристроїв.

Очікується створення інноваційного інструменту для взаємодії учасників ринку оренди житла, з особливим акцентом на довіру, зручність та мобільність.

1.6. Висновки до розділу 1

У цьому розділі здійснено глибокий аналіз сфери мобільної оренди житла, який підтвердив необхідність створення зручного та надійного інструменту для цифрової взаємодії між орендарями й власниками нерухомості.

На основі вивчення інформаційних джерел, аналітичної літератури, методичних матеріалів та огляду популярних платформ, таких як OLX, DOM.RIA, Flatfy і Booking, було виявлено значні недоліки існуючих рішень. До основних проблем належать відсутність повноцінної системи зворотного зв’язку, недостатня адаптація до мобільних пристроїв та нечітка ідентифікація ролей користувачів.

Розробка мобільного застосунку Rent Trust покликана усунути ці недоліки завдяки своїм унікальним перевагам. Його ключові особливості включають орієнтацію на довірчі відносини між користувачами, використання PWA-архітектури, впровадження двосторонньої рейтингової системи, зручний інтуїтивний мобільний інтерфейс та прозору взаємодію між сторонами процесу оренди.

Формулювання задачі дозволило детально визначити технічне бачення майбутньої системи. Воно охоплює реалізацію авторизації, фільтрації оголошень, інтеграцію рейтингової моделі,підтримку режиму роботи без доступу до мережі.

Завдання було зосереджене на виконанні повного комплексу робіт, що включав вибір технологічного стеку (Laravel, PHP, MySQL, Blade, Bootstrap, JavaScript, PWA), розробку адаптивного інтерфейсу в темному дизайні, створення системи автентифікації з чітким поділом ролей (орендодавець, орендар, адміністратор), розробку рейтингової системи, впровадження пошуку та фільтрації житла, інтеграцію підтримки мобільної платформи через

Progressive Web App, а також реалізацію механізму адміністрування контенту та користувачів.

Завдяки проведеному аналізу сформульовано чіткі вимоги, які стануть концептуальною основою для проєктування архітектури та реалізації функціоналу Rent Trust у наступних частинах дипломної роботи.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Логічна модель даних у вигляді ER-діаграми

ER-моделювання являє собою низхідний підхід до проєктування БД, що починається з визначення найбільш важливих даних, так званих сутностей, і зв'язків між даними, що мають бути подані в моделі [3].

ER-діаграма повинна відображати всі бізнес правила предметної області, які, в свою чергу, визначають сутності, атрибути, зв’язки та ін., то без детального аналізу предметної області проектант може не врахувати деякі її важливі аспекти[4].

Основні компоненти ER-діаграми:

Сутність (Entity) — об’єкт чи концепція з реального світу, яка є значущою для системи. Відображається прямокутником, наприклад: Користувач, Будинок, Бронювання.

Атрибут (Attribute) — характеристика сутності, яка зображується у вигляді овалу, під’єднаного до сутності. Наприклад: Ім’я, Адреса, Оцінка.

Ключовий атрибут (Key Attribute) — унікальна характеристика, що ідентифікує кожен екземпляр сутності. Позначається підкресленням.

Зв’язок (Relationship) — асоціація між сутностями. Відображається в ромбі, наприклад: Бронює, Оцінює, Володіє.

Кардинальність (Cardinality) — визначає кількість екземплярів однієї сутності, які можуть бути пов’язані з іншою. Наприклад, один користувач може створити кілька будинків.

На рисунку 2.1 представлена ER-модель бази даних мобільного застосунку для оренди житла з використанням PWA. Основні системні сутності:

Користувач (User) — може виконувати одну з ролей (адміністратор, орендодавець або орендар). Він володіє будинками, здійснює бронювання, залишає оцінки та відгуки.

Роль (Role) — визначає права доступу користувача. Має зв’язок 1:N, де одна роль може стосуватися багатьох користувачів.

Область (Area) — географічна зона, де розташовані будинки. Створюється користувачем.

Будинок (House) — об’єкт оренди з такими атрибутами, як адреса, кількість кімнат, орендна плата. Пов’язаний із користувачем та областю через зв’язки N:1.

Бронювання (Booking) — запис про факт оренди будинку між орендодавцем і орендарем. Має зв’язок M:N між користувачами та зв’язок 1:N між будинком і бронюваннями.

Оцінка (Rating) — показує оцінювання будинку користувачем. Включає зв’язки 1:N між користувачем та оцінками, а також між будинком та оцінками.

Відгук (Review) — текстовий коментар користувача щодо конкретного будинку.

PWA-компонент — логічна умовна сутність, що моделює функціонал прогресивного вебдодатку. Хоча окрема таблиця в базі даних для цього компонента не створюється, він впливає на логіку кешування, доступ до даних офлайн і структуру інтерфейсу користувача.

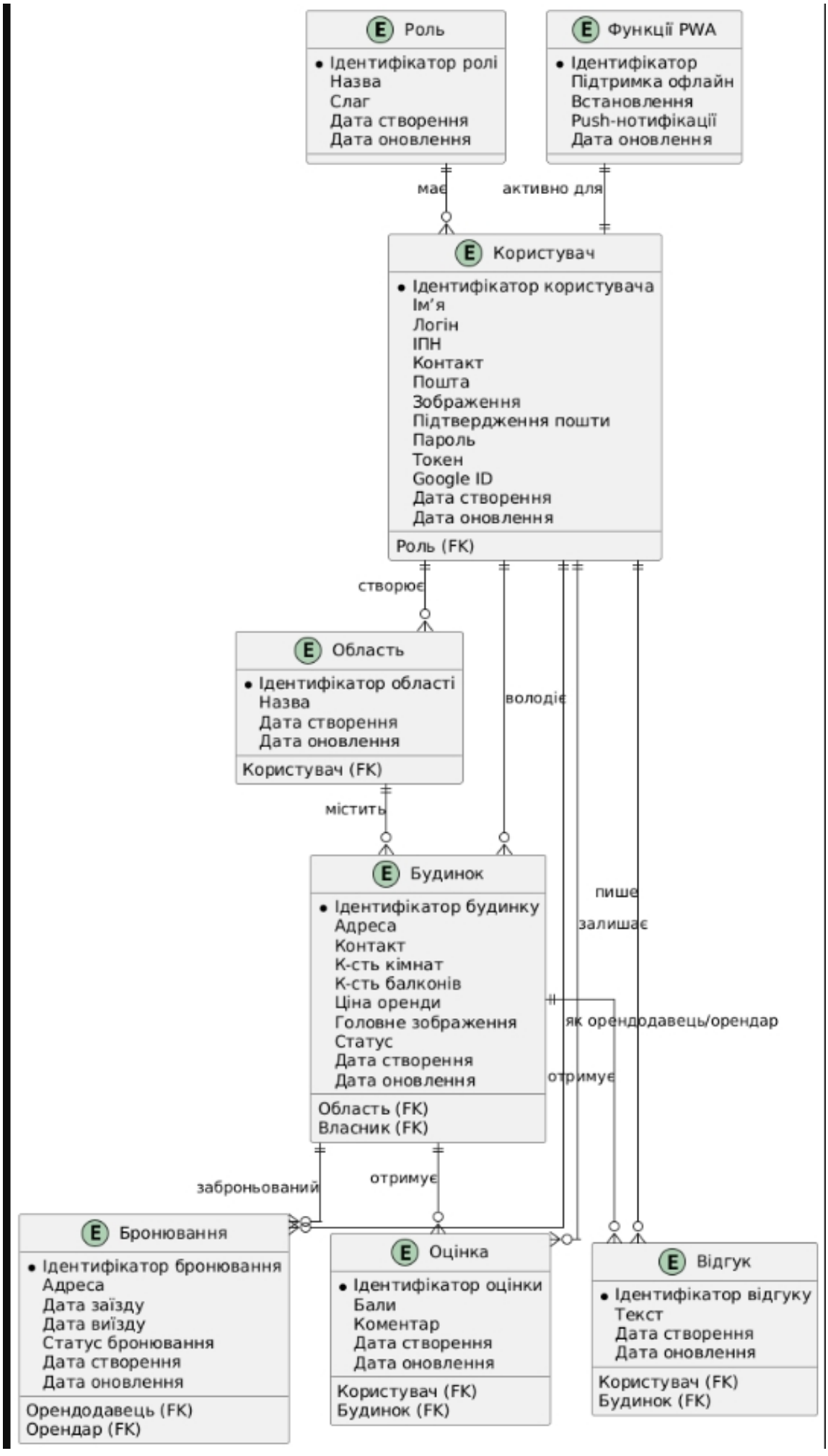

Роль
• Ідентифікатор ролі
Назва
Слаг
Дата створення
Дата оновлення
Функції PWA
• Ідентифікатор
Підтримка офлайн
Встановлення
Push-нотифікації
Дата оновлення
має
активно для
Користувач
• Ідентифікатор користувача
Ім'я
Логін
ІПН
Контакт
Пошта
Зображення
Підтвердження пошти
Пароль
Токен
Google ID
Дата створення
Дата оновлення
Роль (FK)
створює
Область
• Ідентифікатор області
Назва
Дата створення
Дата оновлення
Користувач (FK)
володіє
містить
Будинок
• Ідентифікатор будинку
Адреса
Контакт
К-сть кімнат
К-сть балконів
Ціна оренди
Головне зображення
Статус
Дата створення
Дата оновлення
Область (FK)
Власник (FK)
пише
залишає
як орендодавець/орендар
отримує
заброньований
отримує
Бронювання
• Ідентифікатор бронювання
Адреса
Дата заїзду
Дата виїзду
Статус бронювання
Дата створення
Дата оновлення
Орендодавець (FK)
Орендар (FK)
Оцінка
• Ідентифікатор оцінки
Бали
Коментар
Дата створення
Дата оновлення
Користувач (FK)
Будинок (FK)
Відгук
• Ідентифікатор відгуку
Текст
Дата створення
Дата оновлення
Користувач (FK)
Будинок (FK)

Рис. 2.1. Логічна модель даних у вигляді ER-діаграми

2.2. Діаграма класів та кооперацій

За допомогою діаграми кооперації можна описатиповний контекст взаємодій як своєрідний часовий зріз сукупності об’єктів, що взаємодіють між собою для виконання певного завдання або бізнес-мети програмної системи. На відміну від діаграми послідовності, на діаграмі кооперації зображені тільки відношення між об’єктами, що відіграють певніролі у взаємодії. [5].

Під час моделювання системи було виокремлено основні складові її структури: Користувач, Роль, Будинок, Область, Бронювання, Оцінка, Відгук та механізми підтримки PWA. Усі ці елементи відображено на діаграмі класів. Основна мета такої діаграми — показати взаємозв’язки між об’єктами, а також перелік атрибутів і методів кожного класу.

Зв’язок між роллю та користувачем є неідентифікуючим: одна роль може відповідати кільком користувачам, але кожен користувач має унікальний ідентифікатор і виконує лише одну роль у системі.

Аналогічний тип зв’язку існує між користувачем і областю. Один користувач може створювати декілька областей, до яких у подальшому додаються будинки.

У зв’язку між областю та будинком також спостерігається неідентифікуючий характер. Область може містити багато будинків, при цьому кожен будинок має власний ідентифікатор і не залежить від конкретної області.

Зв’язок між користувачем і будинком відображає ситуацію, коли один користувач, як власник, може виставляти кілька об’єктів нерухомості для оренди (зв’язок типу 1:N).

Користувач також може створювати багато бронювань. Кожне з них прив’язане до певного будинку та його власника. Цей зв’язок між користувачем і бронюванням також є типу 1:N.

У свою чергу, будинок може бути заброньований декілька разів різними орендарями, але кожне бронювання завжди стосується лише одного будинку. Це також зв’язок типу 1:N.

Орендарі мають можливість оцінювати житло, тому між користувачем і оцінками існує зв’язок типу 1:N. Кожна оцінка містить посилання на автора. Аналогічний зв’язок — між будинком та оцінками.

Користувач може також залишати відгуки про житло після завершення оренди, що формує ще один зв’язок типу 1:N. Відгуки прив’язуються до конкретних будинків, які можуть мати багато таких записів від різних осіб.

Особливе місце у системі займає реалізація підтримки PWA (Progressive Web App). Це реалізовано через окремий клас, який відповідає за офлайн-доступ, можливість встановлення застосунку на головний екран пристрою та push-сповіщення. Цей функціонал пов’язаний із середовищем використання системи кожним користувачем і суттєво покращує взаємодію з додатком.

З точки зору об’єктно-орієнтований підходу основною діаграмою UML є діаграма класів. Діаграма демонструє класифікацію моделей з точки зору їх представлення (реалізації). Базовими елемен-тами діаграми класів виступають класи та відноше-ння між ними. Клас може зображатися у деталізо-ваній формі (як клас моделі), скороченій (як класипредметні та інформаційні) та найпростішій (яккласи вербальні та формалізовані [6].

Під час процесу моделювання було ідентифіковано основні сутності, які формують архітектурну основу програмної системи. До них належать: Користувач, Роль, Будинок, Область, Бронювання, Оцінка, Відгук і PWA-механізми. Усі ці елементи відображено на діаграмі класів (рисунок 2.2), яка є ключовим інструментом для опису структури системи, визначення атрибутів та методів класів, а також відношень між ними.

Основні зв’язки між класами системи мають наступний вигляд:

Роль – Користувач (1:N): одна роль може бути призначена багатьом користувачам, тоді як кожен користувач має лише одну роль. Клас користувача містить унікальний ідентифікатор (ID), що забезпечує однозначну ідентифікацію у системі.

Користувач – Область (1:N): користувач має можливість створювати кілька географічних областей, до яких згодом можуть прив'язуватись будинки.

Область – Будинок (1:N): кожна область може включати декілька будинків. При цьому будинки мають власні унікальні ідентифікатори і можуть існувати незалежно від області, до якої були додані.

Користувач – Будинок (1:N): користувач, який виступає у ролі орендодавця, може створювати кілька оголошень про доступні для оренди будинки.

Користувач – Бронювання (1:N): користувач як орендар може створювати багато записів про бронювання. Кожне бронювання обов’язково прив’язується до конкретного будинку та його власника.

Будинок – Бронювання (1:N): один будинок може бути заброньований багаторазово різними користувачами. Проте кожне бронювання асоціюється лише з одним конкретним будинком.

Користувач – Оцінка (1:N): користувач може залишати кілька оцінок різним будинкам. Кожна оцінка містить посилання на автора, що дозволяє відстежувати її походження.

Будинок – Оцінка (1:N): кожен будинок може мати багато оцінок, які залишають інші користувачі, що забезпечує гнучку систему зворотного зв’язку.

Користувач – Відгук (1:N): користувачі мають можливість залишати текстові відгуки про досвід оренди, які зберігаються у системі.

Будинок – Відгук (1:N): один будинок може містити багато відгуків від різних орендарів.

Крім основних сутностей, система підтримує сучасні технології Progressive Web App (PWA). Вони реалізуються через окремий клас PWA, який відповідає за обробку функціоналу, пов’язаного з автономним режимом роботи, можливістю встановлення застосунку на головний екран пристрою користувача, а також реалізацію push-нотифікацій. Цей клас логічно асоціюється з контекстом взаємодії користувача з додатком, значно покращуючи загальний користувацький досвід.

Таким чином, діаграма класів відображає логічну структуру системи, дозволяючи наочно побачити взаємозв’язки між її складовими, що є необхідною основою для подальшого проектування та реалізації системи.

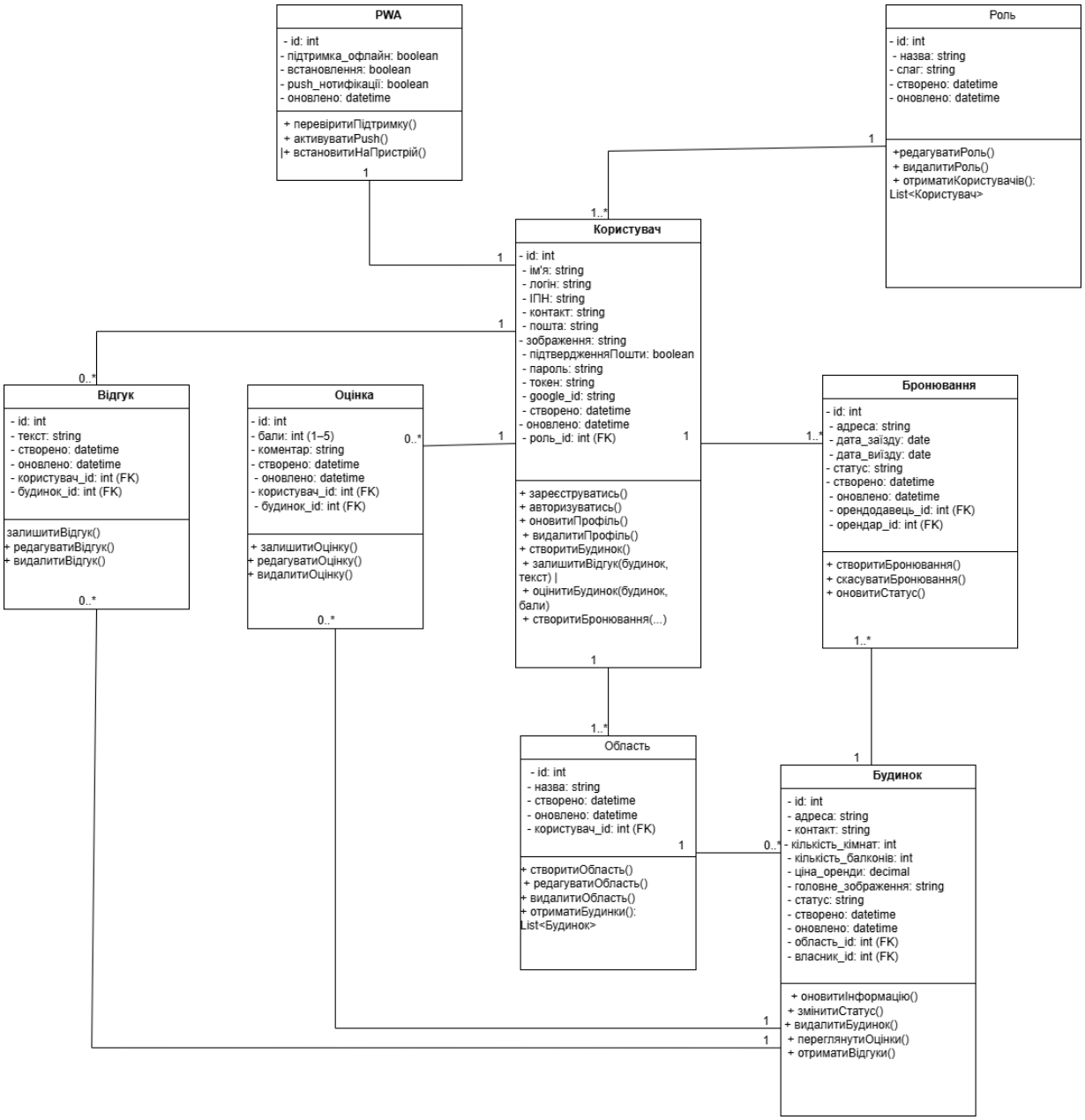


Рис. 2.2. Діагама класів

2.3. Діаграма пакетів

Діаграма пакетів (Package diagram) – структурна діаграма, основним змістом якої єпакети і відносини між ними [7, с.39].

Діаграма пакетів використовується для логічного впорядкування компонентів системи за функціональними або технічними критеріями. Її основне завдання — організувати систему так, щоб забезпечити краще розуміння архітектури, спростити процеси розробки, тестування, масштабування та підтримки. На рисунку 2.3 зображено діаграму пакетів, яка ілюструє архітектуру мобільного застосунку Rent Trust, структуровану за основними логічними рівнями.

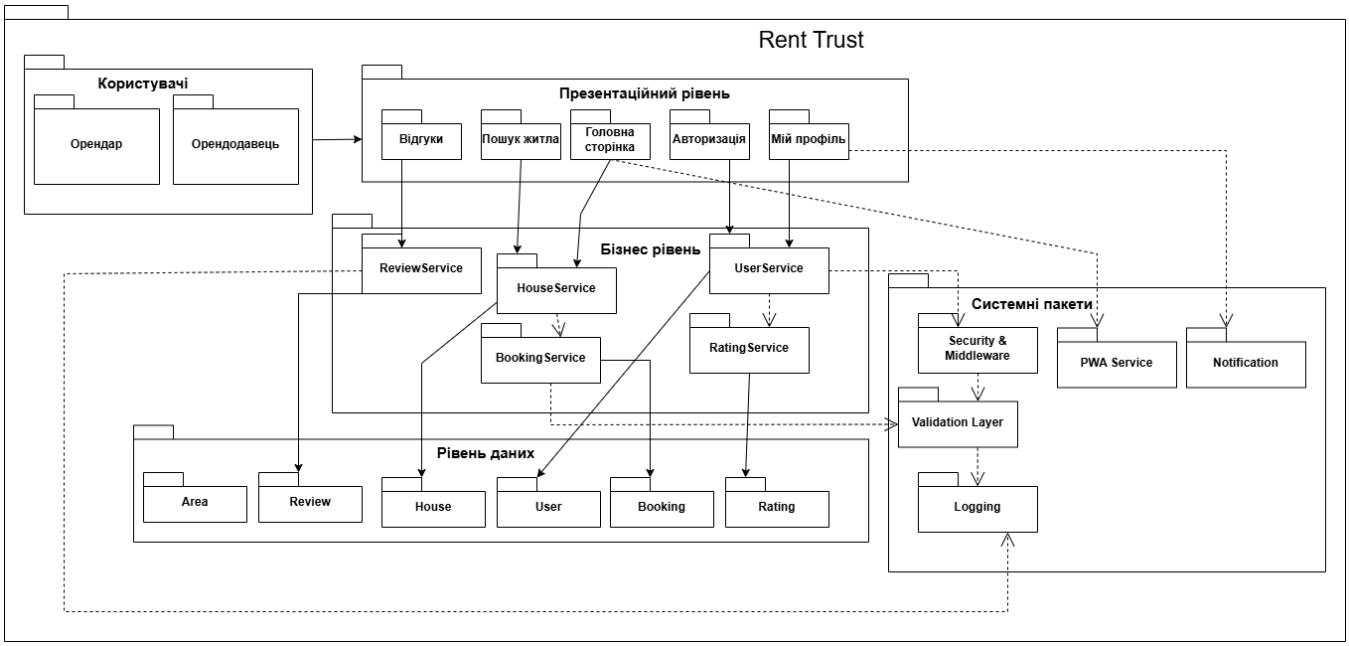


Рис. 2.3. Діаграма пакетів

Рівень презентації здійснює взаємодію користувача з додатком. У системі Rent Trust інтерфейс побудований як прогресивний веб-додаток (PWA) із використанням Blade-шаблонів, Bootstrap, адаптивного дизайну та підтримки темної теми. На цьому рівні користувачі можуть: реєструватися та авторизуватися, переглядати житло та фільтрувати його за різними параметрами, створювати або бронювати оголошення, залишати рейтинги та відгуки. Тут відбувається обробка всіх взаємодій, включаючи кліки, форми,

перемикачі, пуш-нотифікації тощо. Запити надсилаються до контролерів бізнес-рівня для подальшої обробки.

На цьому рівні обробляються всі події взаємодії, такі як кліки, заповнення форм, активація перемикачів, отримання push-нотифікацій тощо. Запити далі переспрямовуються до контролерів бізнес-рівня для подальшої обробки.

Бізнес-рівень є основним шаром логіки, на якому базується робота додатку. Його реалізація включає використання Laravel-контролерів, сервісів, middleware і політик доступу. Основні функціональні можливості організовані в такі пакети:

HouseService — відповідає за створення, редагування, публікацію і деактивацію оглошень;

BookingService — займається обробкою заявок на бронювання, перевіркою дат і ролей;

RatingService — реалізує логіку розрахунку рейтингів користувачів і будинків;

ReviewService — забезпечує управління текстовими відгуками;

AreaService — відповідає за керування географічними одиницями;

UserService — охоплює оновлення профілю, зображення та підтвердження електронної пошти;

AuthService — реалізує автентифікацію через Google, роботу з токенами і middleware.

Цей рівень забезпечує контроль дозволів, перевірку даних, обробку транзакцій і взаємодію з рівнем даних.

Рівень даних реалізовано за допомогою ORM Laravel (Eloquent), що забезпечує взаємодію з реляційною базою даних MySQL. У цьому шарі представлені всі основні сутності: Користувачі, Ролі, Будинки, Області, Бронювання, Оцінки, Відгуки.

Моделі Eloquent відповідають за збереження та оновлення даних, а також за встановлення зв’язків між таблицями (hasMany, belongsTo тощо) і здійснення фільтрації інформації. Уся логіка, пов’язана з читанням і записом даних, централізована через ці об’єкти, що гарантує їх цілісність і узгодженість.

Системні пакети

Цей рівень включає міжфункціональні компоненти, які забезпечують стабільність, безпеку та зручність користування системою:

PWA Service — підтримка роботи в офлайн-режимі, створення маніфесту та використання сервісворкерів;

Notifications — впровадження push-сповіщень для користувачів (наприклад, підтвердження бронювань);

Validation Layer — централізована обробка та перевірка введених даних (використання Form Requests);

Security & Middleware — функції авторизації, аутентифікації, захист від CSRF-атак і контроль доступу до ресурсів;

Logging & Monitoring — фіксування подій, дій користувачів і системних помилок.

Завдяки модульній структурі система Rent Trust відзначається логічною зрозумілістю, високою масштабованістю, сприяє повторному використанню коду, полегшує процес відлагодження та забезпечує зручність подальшого розвитку. Крім того, використання фреймворку Laravel дає змогу ефективно розділяти логіку на рівні та забезпечує суворе дотримання принципів MVC.

2.4. Діаграма компонентів

Діаграма компонентів (Component diagram) – статична структурна діаграма, яка демонструє декомпозицію програмної системи на структурні компоненти і зв’язки(залежності) між компонентами. В якості фізичних компонентів можуть виступати файли,бібліотеки, модулі, виконувані файли, пакети тощо [8, с.111].

Діаграма компонентів дозволяє визначити архітектуру розроблюваної системи, встановивши залежності між програмними компонентами, в ролі яких може виступати вихідний, бінарний і виконуваний код. [9].

Під час розробки системи TrustRent створено діаграму компонентів (рис. 2.4), що відображає ключові структурні елементи архітектури, їхню взаємодію та зв’язок із базою даних. Ця модель дає змогу чітко визначити функції компонентів, їхні завдання, а також сприяє підвищенню масштабованості, підтримуваності та безпеки системи.

Рівень презентації :

Компонент PWA App відповідає за мобільний прогресивний вебзастосунок, який забезпечує взаємодію користувача із системою. Тут реалізовано інтерфейс користувача, можливість роботи в офлайн-режимі, встановлення застосунку на головний екран пристрою, а також виклики до бізнес-логіки через PWA безпосередньо контактує з компонентом User, що є логічним уособленням користувача в системі, і використовує порт Account для управління обліковими даними.

Рівень бізнес-логіки :

На цьому рівні функціонують ключові компоненти:User management,який забезпечує реєстрацію, авторизацію, редагування профілю та управління користувачами.House management,який відповідає за додавання, редагування, видалення та перегляд інформації про житло.Booking management котрий займається процесами бронювання житла: створенням, підтвердженням і скасуванням замовлень.Rating management який охоплює

роботу з оцінками: виставлення рейтингів, перегляд середніх показників та обробку відгуків.

Інфраструктурний рівень, Backend взаємодіє з базою даних MySQL, у якій розміщені такі таблиці: User Table (містить облікові записи користувачів). Houses Table (зберігає інформацію про об’єкти житла). Bookings Table (фіксує дані про всі бронювання). Ratings Table (містить відгуки та рейтинги, залишені користувачами).PWA Table. Ці таблиці є фізичними репрезентаціями моделей, що використовуються у логіці застосунку. Всі бізнес-компоненти взаємодіють із даними виключно через Backend, що гарантує узгодженість, цілісність і безпеку системи.

Діаграма компонентів наочно демонструє систему TrustRent як сукупність ізольованих функціональних модулів з чітко визначеними ролями та взаємодією між ними. Такий підхід сприяє модульній розробці, спрощує тестування окремих елементів системи і забезпечує можливість інтеграції нових сервісів без шкоди для загальної структури. Використання PWA-додатку забезпечує зручність роботи із системою на мобільних пристроях, зберігаючи повний функціонал і адаптивність інтерфейсу.

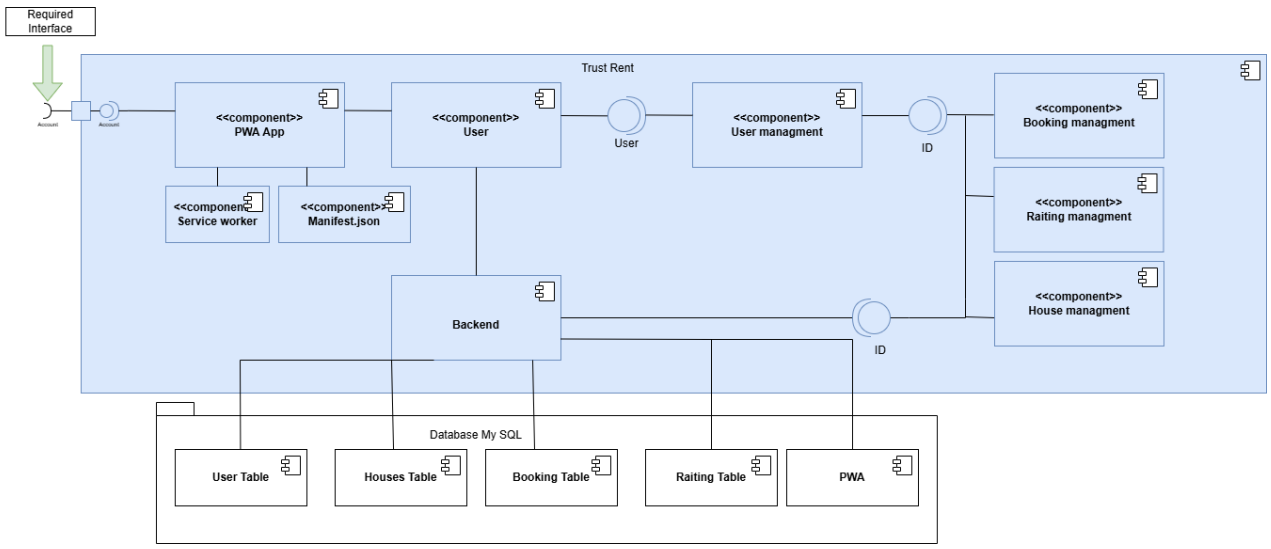


Рис. 2.4. Діаграма компонентів

2.5. Висновок до розділу 2

У цьому розділі описано поетапне проєктування програмного забезпечення для мобільного застосунку Rent Trust, що встановило міцну базу для подальшої реалізації системи. На основі аналізу вимог і сценаріїв використання створено комплексну модель, яка враховує як інформаційну, так і архітектурну структуру додатку.

Під час побудови ER-діаграми були визначені основні сутності:User, House, Booking, Rating, а також їхні атрибути і зв’язки. Це дозволило сформувати логічну модель бази даних з урахуванням взаємозв’язків між орендарями, орендодавцями, об'єктами житла та оцінками. Такий підхід забезпечив цілісність даних та мінімізацію зайвої інформації.

Діаграма класів деталізувала об'єктну модель програми, окресливши ключові класи, їхні властивості (id, name, email, address, ratingValue тощо) та методи (bookHouse(), rateUser(), createHouse()). Це дозволило забезпечити узгодженість у структурі коду й бази даних, що полегшує подальшу реалізацію програмного компонента.

Діаграма кооперацій ілюструвала взаємодію об'єктів у межах ключових сценаріїв, таких як створення бронювання, виставлення оцінки чи перегляд об’єктів оренди. Вона відображала послідовність викликів та обмін повідомленнями між класами User, Booking, Rating, House і компонентами RatingService та NotificationService. Це допомогло деталізувати поведінкові особливості системи та взаємодію її компонентів.

Діаграма компонентів відобразила архітектуру програмного забезпечення на логічному рівні й включала три основні рівні: мобільний інтерфейс користувача (PWA App), бізнес-рівень (User Management, House Management, Booking Management, Rating Management) та інфраструктурний рівень (Backend і база даних MySQL). Така модульна структура сприяє гнучкості, масштабованості й спрощенню тестування. Формат PWA було обрано для забезпечення користувачам доступу до застосунку через веб-

інтерфейс із функціями мобільного додатку без потреби завантаження з магазинів. Це включає підтримку офлайн-режиму, push-сповіщень і швидкого завантаження.

У підсумку створена модель інтегрує зрозумілу структуру даних, логіку взаємодії та надійну технологічну архітектуру, яка відповідає сучасним стандартам для мобільних застосунків. Вона надає міцну основу для реалізації функціонального, масштабованого й зручного рішення, орієнтованого на ефективну взаємодію між орендарями та орендодавцями.

3 РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Система управління інформаційною базою

У сучасних мобільних додатках, особливо тих, які реалізуються у форматі Progressive Web App (PWA), правильний вибір системи управління базами даних (СУБД) відіграє вирішальну роль у забезпеченні продуктивності, цілісності, безпеки та масштабованості даних. Для застосунку Rent Trust, спрямованого на взаємодію між орендарями та орендодавцями, використано реляційну модель бази даних, яка передбачає чітко визначені зв’язки між різними сутностями.

Основою вибору стала MySQL — одна з найбільш популярних СУБД реляційного типу. Вона підтримує стандарт SQL, забезпечує індексацію, транзакції (зокрема через рушій InnoDB) і демонструє високу продуктивність при операціях читання та запису. Рішення віддати перевагу цьому продукту обумовлено потребою в надійному зберіганні структурованих даних, таких як інформація про користувачів, об’єкти житла, бронювання, відгуки, повідомлення та інші взаємопов’язані елементи.

На етапі локальної розробки та тестування СУБД розгорнуто у середовищі Laravel Homestead або XAMPP, що дає можливість надалі переносити розроблену базу на хмарну платформу чи VPS. Робота з базою

даних здійснюється через ORM Laravel — Eloquent, який значно полегшує взаємодію з таблицями та гарантує безпечне формування SQL-запитів з використанням об’єктної моделі.

У таблиці 3.1 наведено короткий порівняльний аналіз найбільш популярних реляційних СУБД, які могли б бути розглянуті як альтернативні варіанти для цього застосунку.

Таблиця 3.1 – Порівняльний аналіз реляційних СУБД

СУБД	**Переваги**	**Недоліки**
MySQL	Висока швидкість роботи, простота встановлення, поширеність	Має обмежену гнучкість у складних аналітичних запитах і не повністю відповідає стандарту SQL.
PostgreSQL	Пропонує потужну функціональність, підтримує складні транзакції та роботу з JSON-полями	Відзначають вищу складність конфігурації та більшу вимогливість до ресурсів
SQLite	Є чудовим вибором для швидкого розгортання та прототипування завдяки своїй простоті	Він не підходить для масштабованих систем та має обмеження у багатокористувацькому доступі
Microsoft SQL Serve	Відомий інтеграцією з екосистемою Microsoft і наявністю інструментів бізнес-аналітики	Його недоліками є закритість платформи, висока вартість ліцензій і орієнтованість

		переважно на Window
MariaDB	Це відкрита альтернатива MySQL із додатковими можливостями	Сумісність із MySQL часткова, а підтримка деяких сторонніх сервісів менш активна.

Беручи до уваги архітектуру проєкту, специфіку даних та переваги використання Laravel як основного фреймворку, MySQL виявляється оптимальним вибором для створення інформаційної бази мобільного застосунку. Він забезпечує хороший баланс між продуктивністю, гнучкістю, простотою налаштування та підтримкою з боку спільноти розробників.

3.2. Розробка інформаційної бази

Розробка інформаційної бази для програмного забезпечення мобільного додатку оренди житла з рейтинговою системою розпочалася з проектування логічної моделі даних та створення моделей Eloquent на основі фреймворку Laravel. У цьому контексті кожна модель у Laravel відповідає окремій таблиці реляційної бази даних MySQL, а зв’язки між об’єктами реалізовуються через вбудовані методи ORM.

Основними сутностями системи є:

- users — користувачі системи з ролями орендодавця, орендаря або адміністратора;
- roles — ролі користувачів;
- houses — оголошення про оренду житла;
- areas — райони розміщення житла;
- bookings — заявки на бронювання житла;
- reviews — рейтинги та відгуки користувачів.

Структура даних складається з наступних таблиць:

1. users: містить інформацію про користувачів із такими полями — id, role_id, name, username, email, password, google_id, nid, contact, email_verified_at.
2. roles: зберігає дані про ролі з полями — id, name.
3. houses: включає інформацію про будинки — id, user_id, area_id, title, description, price, address, created_at.
4. areas: відповідає за дані про райони із наступними полями — id, name, user_id.
5. bookings: забезпечує інформацію про бронювання з такими полями — id, landlord_id, renter_id, house_id, start_date, end_date, status.
6. reviews: зберігає відгуки з полями — id, user_id, house_id, rating, comment, created_at,ratings

```
     * @return void
     */
    public function up()
    {
        Schema::create( table: 'users', function (Blueprint $table) {
            $table->bigIncrements( column: 'id');
            $table->integer( column: 'role_id')->default( value: 2);
            $table->string( column: 'name');
            $table->string( column: 'username')->unique();
            $table->string( column: 'nid')->unique();
            $table->string( column: 'contact')->unique();
            $table->string( column: 'email')->unique();
            $table->string( column: 'image')->nullable();
            $table->timestamp( column: 'email_verified_at')->nullable();
            $table->string( column: 'password');
            $table->rememberToken();
            $table->timestamps();
        });
    }
```

```
    public function up()
    {
        Schema::create( table: 'roles', function (Blueprint $table) {
            $table->bigIncrements( column: 'id');
            $table->string( column: 'name');
            $table->string( column: 'slug');
            $table->timestamps();
        });
    }
```

```php
 */
public function up()
{
    Schema::create( table: 'areas', function (Blueprint $table) {
        $table->bigIncrements( column: 'id');
        $table->string( column: 'name');
        $table->integer( column: 'user_id');
        $table->timestamps();
    });
}
```

```php
class CreateBookingsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create( table: 'bookings', function (Blueprint $table) {
            $table->bigIncrements( column: 'id');
            $table->string( column: 'address');
            $table->integer( column: 'rent');
            $table->string( column: 'leave')->nullable();
            $table->integer( column: 'landlord_id');
            $table->integer( column: 'renter_id');
            $table->string( column: 'booking_status')->default( value: 'requested');
            $table->timestamps();
        });
    }
```

```php
class CreateHousesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create( table: 'houses', function (Blueprint $table) {
            $table->bigIncrements( column: 'id');
            $table->string( column: 'address');
            $table->integer( column: 'area_id');
            $table->integer( column: 'user_id');
            $table->string( column: 'contact');
            $table->integer( column: 'number_of_room');
            $table->integer( column: 'number_of_toilet');
            $table->integer( column: 'number_of_belcony');
            $table->integer( column: 'rent');
            $table->string( column: 'featured_image');
            $table->text( column: 'images');
            $table->string( column: 'status')->default( value: 1);
            $table->timestamps();
        });
```

```php
class CreateReviewsAndRatingsTables extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create( table: 'reviews', function (Blueprint $table) {
            $table->bigIncrements( column: 'id');
            $table->unsignedBigInteger( column: 'user_id');
            $table->unsignedBigInteger( column: 'house_id');
            $table->text( column: 'opinion');
            $table->unsignedTinyInteger( column: 'rating');
            $table->timestamps();
        });
    }
```

Рис. 3.1. Основні таблиці інформаційної бази системи Rent Trust

Зв’язки між сутностями задаються через Eloquent-моделі. Так, модель User містить методи houses(), areas(), bookings() і reviews(), які відображають зв'язки один-до-багатьох. Водночас, модель Booking має методи landlord() та renter() з параметрами зовнішнього ключа, що забезпечують відношення багато-до-одного до User.

Для збереження цілісності даних у базі використовуються зовнішні ключі та каскадне видалення, що дозволяє автоматично видаляти пов'язані записи.

Також застосовуються методи оптимізації вибірки даних, такі як попереднє завантаження зв’язків (eager loading) і індексація колонок на кшталт user_id, house_id, area_id.

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class House extends Model
{
    public function user(){
        return $this->belongsTo( related: User::class);
    }

    public function area(){
        return $this->belongsTo( related: Area::class);
    }

    no usages
    public function reviews(){
        return $this->hasMany( related: Review::class);
    }
}
```

Рис. 3.2. Фрагмент прикладу структури моделей

Додаткові функціональні можливості, реалізовані в базі даних: Інтеграція механізмів аутентифікації та авторизації через Laravel із забезпеченням збереження сесійних токенів і підтвердження електронної пошти.

Опція реєстрації користувачів через Google із використанням поля google_id.

Впроваджена рейтингова система, у якій кожен відгук має поле rating (від 1 до 5), що впливає на репутацію орендодавця або житла.

Інформаційна база мобільного додатку для оренди житла була створена з урахуванням ключових принципів реляційного моделювання: чіткого розподілу ролей, забезпечення зв’язків між сутностями, можливості розширення, логічної цілісності та належного рівня безпеки. Така структура надає змогу оперативно знаходити житло, легко керувати користувачами й об'єктами, а також впроваджувати рейтингову систему для вдосконалення взаємодії між користувачами сервісу.

3.3. Вибір інструментарію для створення прикладного програмного забезпечення

Розробка прикладного програмного забезпечення для мобільного додатку, орієнтованого на орендарів і власників житла, із вбудованою рейтинговою системою, вимагала ретельного вибору технологічного стеку. Завдання полягало у тому, щоб цей стек відповідав високим стандартам продуктивності, безпеки, масштабованості та забезпечував мобільну доступність у форматі прогресивного веб-додатку (PWA).

Оскільки цільовою платформою є мобільні пристрої, а додаток повинен функціонувати як офлайн, так і з можливістю синхронізації з сервером, було прийнято рішення зосередитися на класичному веб-стеку з підтримкою PWA. Це забезпечує можливість використовувати застосунок як на настільних комп’ютерах, так і на смартфонах, інтегруючи його у вигляді встановленого додатка. Під час аналізу було розглянуто кілька альтернатив для реалізації бекенду:

Node.js разом із Express або Next.js — ці фреймворки забезпечують асинхронну обробку запитів і високу гнучкість. Однак для них потрібна окрема розробка як фронтенду, так і бекенду.

Django (Python) — пропонує потужну ORM-систему, ефективний механізм авторизації, але менш зручний для створення мобільного PWA-інтерфейсу.

Laravel (PHP) — володіє вбудованою підтримкою MVC-архітектури, потужною маршрутизацією, шаблонним рушієм Blade, а також активною спільнотою та великим набором готових пакетів.

Зважаючи на наявний досвід, швидкість розробки та зручність інтеграції з мобільними інтерфейсами, для реалізації проєкту був обраний технологічний стек на основі Laravel, Blade, Bootstrap 5 разом із функціоналом прогресивного веб-застосунку (PWA).

Складові обраного стеку:

Laravel (PHP фреймворк) — забезпечує побудову застосунку за архітектурним шаблоном MVC, дає можливість швидко створювати RESTful API, пропонує вбудовані інструменти для захисту від CSRF-атак, функціонал авторизації, middleware, CLI Artisan для генерації компонентів і багато іншого.

Шаблонізатор Blade — дозволяє створювати динамічні та адаптивні HTML-сторінки з урахуванням темної теми й інтеграції логіки безпосередньо в шаблонах.

Bootstrap 5 — виступає основним CSS-фреймворком для створення адаптивного мобільного інтерфейсу. Він забезпечує підтримку темної теми, сіток, компонентів, модальних вікон, панелей навігації тощо.

Laravel PWA (пакет) — відповідає за створення manifest.json, реєстрацію service worker, додавання мета-тегів для встановлення веб-застосунку на головний екран мобільного пристрою.

MySQL — реляційна база даних, яка використовується для зберігання інформації про користувачів, оголошення, рейтинги, повідомлення та взаємодії між орендарями та орендодавцями.

PhpMyAdmin / Adminer — інструменти для адміністрування бази даних під час роботи в локальному середовищі.

PhpStorm IDE — основне середовище розробки з підтримкою функціоналу Laravel і Blade-шаблонів, зручним підсвічуванням синтаксису та інтеграцією із системами контролю версій.

PWA-підтримка: Для перетворення звичайного веб-застосунку на PWA були впроваджені такі ключові компоненти:

1. manifest.json — містить інформацію про іконки, колірну схему, назву застосунку і спосіб запуску (standalone).
2. Service Worker — забезпечує кешування сторінок і ресурсів, підтримуючи доступ до них в офлайн-режимі. Адаптивна верстка — оптимізована для малих екранів із врахуванням потреб користувачів мобільних пристроїв.

Обраний стек технологій:

- Backend: API: Laravel 6.2 (PHP 7.2.x)
- Frontend: Blade (Laravel View Engine)
- Стилі: Bootstrap 5 + кастомні стилі
- База даних: MySQL
- Аутентифікація: Laravel Breeze / Laravel UI
- PWA: laravel-pwa пакет
- IDE: PhpStorm
- Веб-сервер:LaravelValet

Отже, обраний інструментарій дозволяє досягти ефективного поєднання: зручної розробки, мобільної адаптивності та реалізації PWA-функціоналу для орендарів і власників житла, при збереженні повноцінної серверної логіки й безпеки на базі Laravel.

3.4. Алгоритмізація та програмування програмних модулів

Під час створення мобільного застосунку для орендарів та орендодавців із вбудованою рейтинговою системою було використано такі інструменти, як PHP (Laravel), JavaScript, Blade, Bootstrap, а також концепції прогресивних веб-застосунків (PWA), що гарантують зручність, надійність та безпеку роботи.

1. Реєстрація нового користувача

Початок процесу: Алгоритм активується, коли новий користувач переходить на сторінку реєстрації в мобільному додатку або через PWA-версію сайту.

Заповнення реєстраційної форми: Користувач вводить такі дані: ім’я, електронну пошту, пароль і повторює пароль для підтвердження. Додатково доступний вибір ролі — орендар або орендодавець (роль можна змінити пізніше).

Підтвердження реєстрації: Після заповнення форми користувач натискає кнопку «Зареєструватися», надсилаючи свої дані для обробки.

Валідація введених даних: Система перевіряє правильність зазначеної електронної пошти, довжину пароля та збіг пароля з його підтвердженням. У випадку помилок з'являється відповідне повідомлення для виправлення.

Створення нового облікового запису: Система Laravel зберігає інформацію нового користувача в таблиці users, попередньо хешуючи пароль. Водночас призначається початкова роль і генерується токен для підтвердження електронної пошти.

Надсилання листа для підтвердження: На вказану електронну пошту користувача автоматично відправляється лист із посиланням для верифікації. До цього моменту обліковий запис залишається неактивним.

Сповіщення в інтерфейсі: Після успішного завершення реєстрації користувач побачить повідомлення: «Реєстрація успішна. Перевірте свою електронну пошту, щоб активувати акаунт».

Завершення процесу: Алгоритм завершується після створення облікового запису та відправлення листа для верифікації. Мета процесу: Забезпечити новим користувачам можливість зареєструвати обліковий запис із базовими правами доступу та надалі верифікувати свою електронну пошту для активації.

..

Рис. 3.4. Блок-схема алгоритму авторизації

2. Додавання оголошення про житло

Початок: Процес стартує, коли авторизований користувач із роллю орендодавця натискає кнопку "Додати оголошення" у своєму кабінеті.

Заповнення форми оголошення: Користувач вводить необхідну інформацію: назву об'єкта, його опис, адресу, район, тип житла, кількість кімнат, вартість, а також додає фотографії.

Натискання кнопки "Опублікувати": Заповнені дані передаються на сервер через форму.

Перевірка коректності даних: Форма здійснює валідацію обов’язкових полів, перевіряє формат чисел, розмір і тип файлів зображень.

Збереження в базі даних: Сервер створює новий запис у таблиці listings, а додані фотографії завантажуються у сховище та прив’язуються до об’єкта через таблицю listing_images.

Статус модерації: Новостворене оголошення отримує статус "pending" і переходить на етап перевірки модератором.

Повідомлення користувачу: Інтерфейс інформує користувача повідомленням "Оголошення надіслано на модерацію".

Кінець процесу: Процедура завершується після успішного створення запису та відображення повідомлення.

Мета: Забезпечити орендодавцям можливість створювати й розміщувати нові пропозиції оренди для подальшої модерації та публікації.

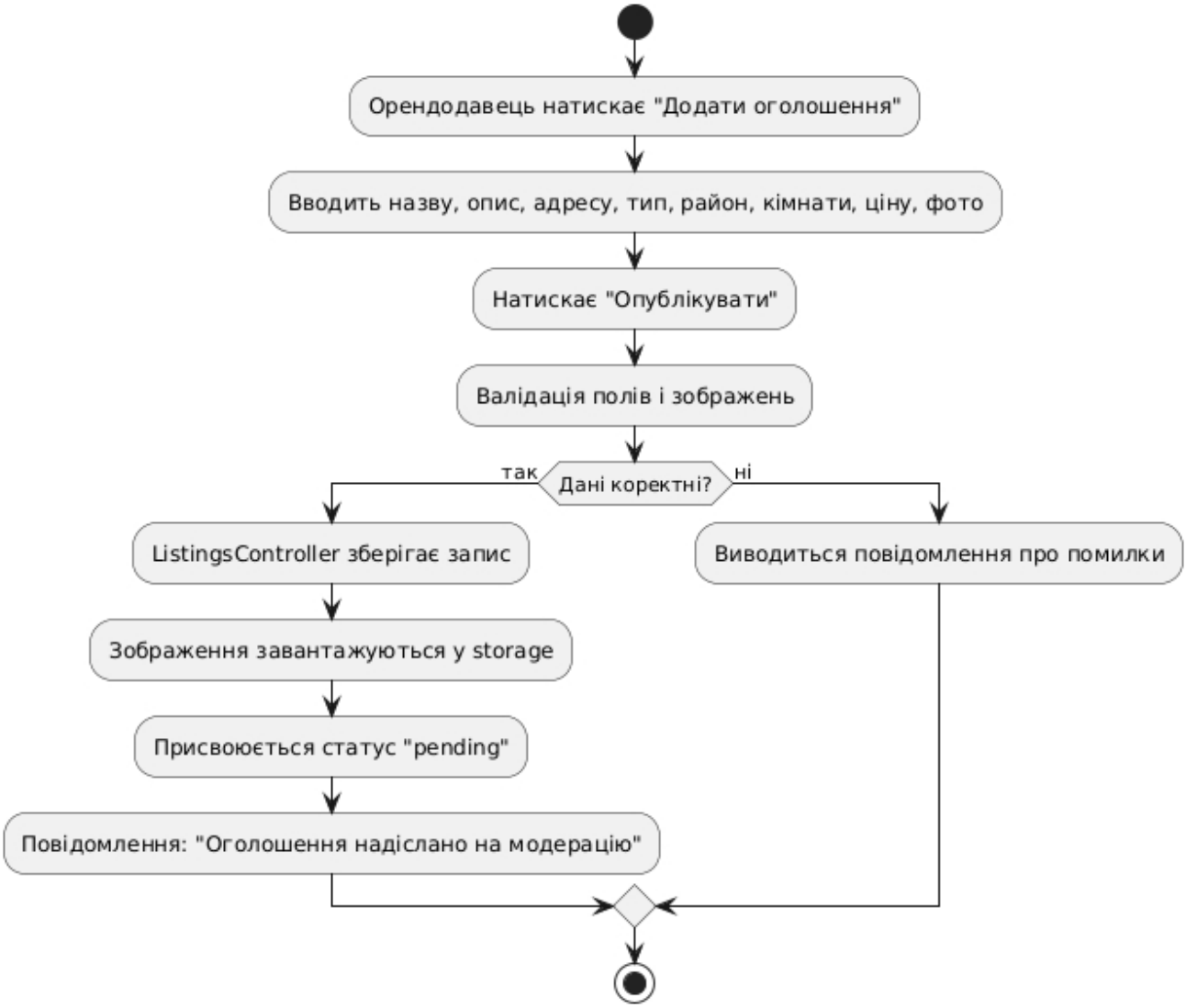


Рис. 3.5. Додавання оголошення про житло

3. Надання оцінки/відгуку після оренди

Початок: Алгоритм активується, коли користувач (орендар або орендодавець) заходить на сторінку завершеного бронювання у своєму особистому кабінеті. Доступ до цієї функції можливий лише після закінчення періоду оренди.

Перевірка можливості залишити відгук: Система перевіряє, чи було вже надано відгук за це бронювання. Якщо відгук існує, функція оцінювання недоступна, і користувачу виводиться повідомлення: "Відгук уже залишено". Якщо ж відгук ще не залишений, система відкриває форму для його створення.

Заповнення форми з оцінкою та коментарем: Користувач обирає рейтинг за п’ятизірковою шкалою (від 1 до 5 балів) та за бажанням додає текстовий коментар щодо свого досвіду оренди. Форму складають два текстові поля: "заголовок" і "детальний опис".

Відправлення відгуку: Після заповнення форми користувач натискає кнопку «Надіслати оцінку».

Валідація введених даних: Перед надсиланням система перевіряє, чи всі обов’язкові поля заповнені (рейтинг обраний, текстовий коментар не залишений порожнім). У разі виявлення помилок користувач отримує повідомлення: "Заповніть всі поля".

Передача даних на сервер: За успішного проходження перевірки формується POST-запит до серверного API на маршрут /api/reviews для обробки введених даних.

Збереження даних у базі: На сервері Laravel-контролер перевіряє, чи користувач має право залишати відгук (проводиться звірка за бронюванням зі статусом "completed"). Якщо все відповідає вимогам, створюється запис у таблиці reviews із такими полями: reviewer_id, reviewed_user_id, rating, comment, booking_id, created_at.

Оновлення середнього рейтингу: Після збереження нового відгуку викликається метод updateAverageRating(). Він обчислює актуальне середнє значення рейтингу для користувача на основі всіх отриманих ним оцінок.

Повідомлення про успішне оцінювання: Після завершення процесу користувач бачить повідомлення: "Дякуємо за ваш відгук!" і автоматично повертається на сторінку завершених бронювань.

Кінець алгоритму: Робота алгоритму завершується після успішного збереження всіх даних і оновлення інтерфейсу.

Мета: Система спрямована на забезпечення контрольованого та однократного процесу оцінювання взаємодії між орендодавцем та орендарем для кожного завершеного бронювання. Це сприяє прозорості системи та

зміцнює довіру між користувачами.Мета: Забезпечити швидке створення документа зі стандартними налаштуваннями та перехід до його редагування.

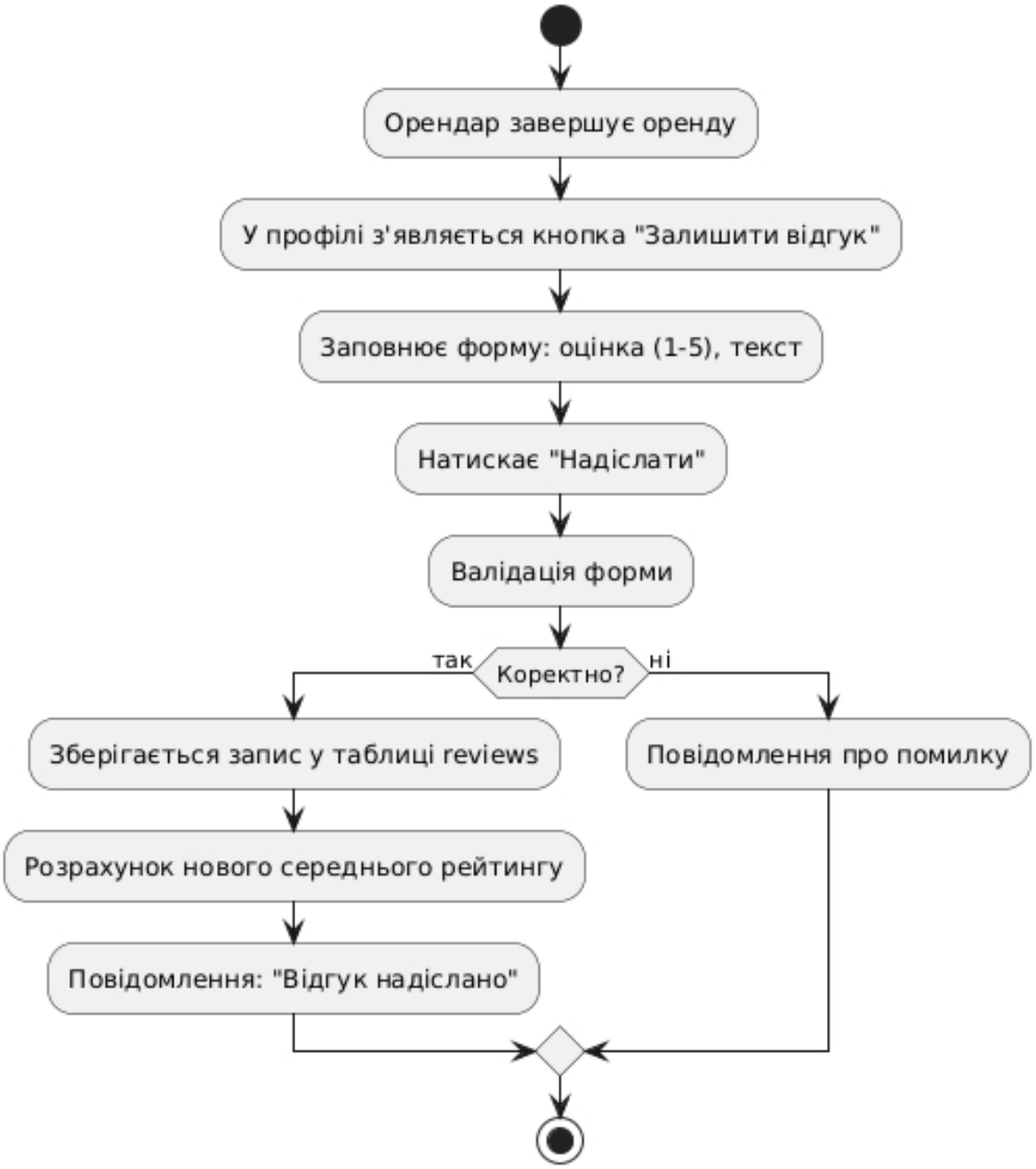


Рис. 3.6. Надання оцінки/відгуку після оренди

4. Установка PWA на смартфон

Початок: Алгоритм стартує з відкриттям користувачем мобільної версії сайту, який відповідає PWA стандартам завдяки файлам manifest.json і service worker.

Перевірка підтримки: Браузер оцінює, чи сайт підходить для інсталяції. Якщо так, з’являється стандартний банер або користувач може самостійно вибрати опцію «Додати на головний екран».

Підтвердження інсталяції: Користувач підтверджує намір встановити застосунок.

Збереження ярлика: Система створює на головному екрані пристрою ярлик з відповідною іконкою та назвою програми.

Мобільна адаптація: Коли сайт відкривається з ярлика, він функціонує як автономний застосунок без адресного рядка в режимі standalone.

Повідомлення користувачу: Деякі браузери можуть відображати повідомлення: «Додаток встановлено».

Кінець: Алгоритм завершується після успішного встановлення програми на пристрій користувача.

Мета: Забезпечити легку інсталяцію та доступ до програми як мобільного додатку без необхідності використовувати App Store чи Google Play.

Рис. 3.7. Блок-схема алгоритму установки PWA на смартфон

5. Пошук житла з фільтрами

Початок: Користувач переходить на сторінку пошуку оголошень і бачить форму для фільтрації.

Введення параметрів: У формі користувач зазначає потрібний район, тип житла, кількість кімнат, ціновий діапазон, дату доступності та інші критерії.

Натискання кнопки «Знайти»: Після заповнення форми надсилається GET-запит із зазначеними параметрами.

. Пошук у базі даних: ORM-запит проводить обробку в таблиці застосовуючи задані фільтри, включно з умовою статусу "approved".

Повернення результатів: Результати пошуку повертаються в вигляді списку, який демонструється користувачу із функцією пагінації або нескінченної прокрутки.

Повідомлення при відсутності даних: Якщо відповідних записів не знайдено, користувач бачить повідомлення: «Житло не знайдено за обраними параметрами».

Кінець: Алгоритм завершується після виведення списку результатів або відповідного повідомлення.

Мета: Забезпечити користувача зручним інструментом для пошуку житла відповідно до його індивідуальних вимог.

Рис. 3.8. Блок-схема алгоритму пошук житла з фільтрами

7. Надсилання запиту на бронювання

Початковий етап: Користувач обирає потрібне оголошення та натискає кнопку «Забронювати».

Заповнення форми: Вказуються дати початку і завершення оренди, а також за бажанням додається повідомлення для орендодавця.

Перевірка доступності: Система перевіряє можливість бронювання, виключаючи конфлікти з іншими активними запитами.

Збереження даних: На рівні Laravel здійснюється запис нового бронювання у таблицю bookings зі статусом pending (очікування підтвердження).

Оповіщення орендодавця: Орендодавцю надходить повідомлення про новий запит через особистий кабінет або електронну пошту.

Інформація для користувача: На екрані з'являється повідомлення: «Запит на бронювання надіслано. Очікуйте підтвердження».

Завершення процесу: Система переходить у стан чекання відповіді від орендодавця.

Головна мета: Організувати прозорий процес оренди між сторонами із запобіганням конфліктам і дублюванням бронювань.

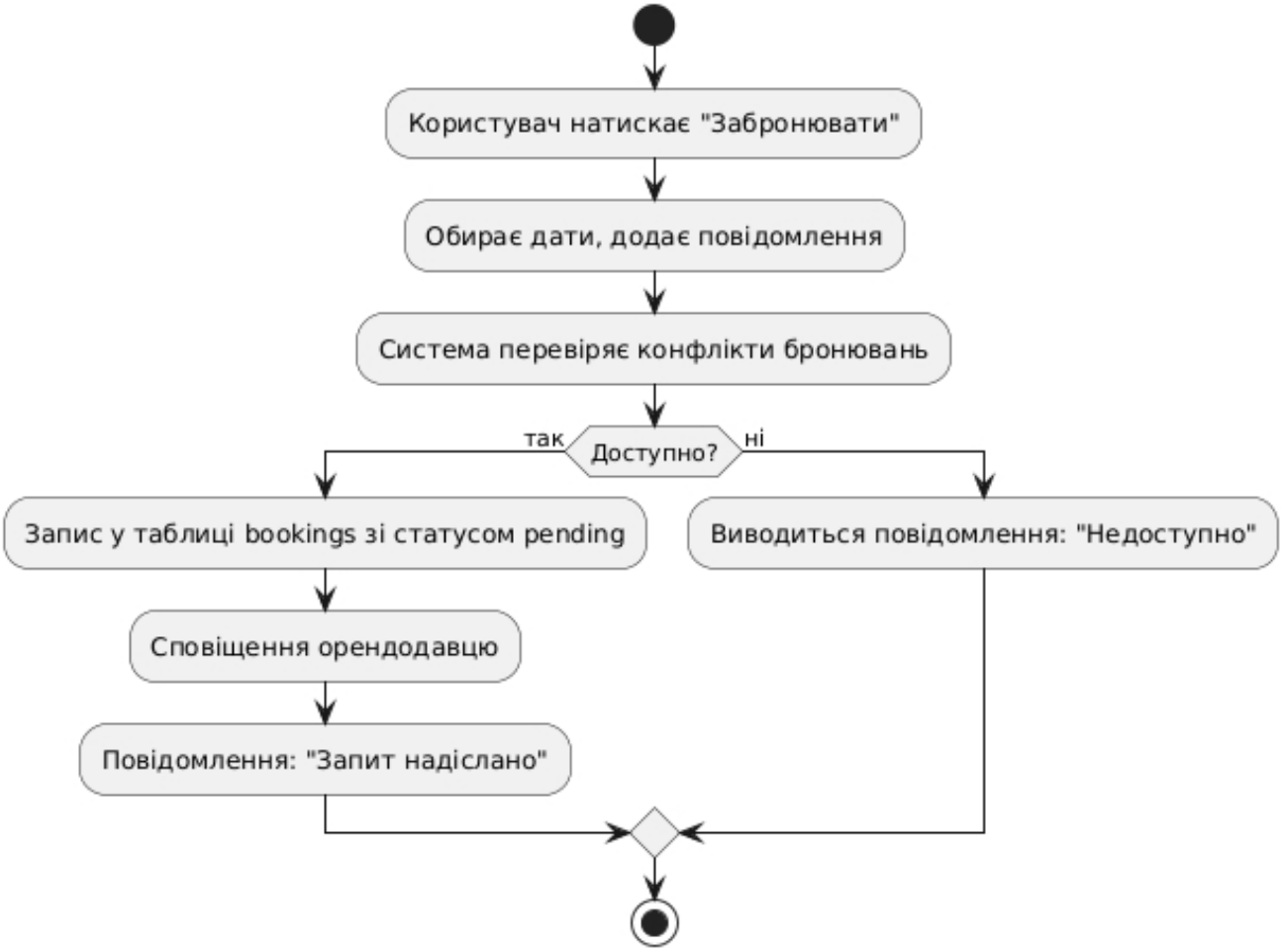


Рис. 3.9. Блок-схема алгоритму надсилання запиту на бронювання

3.5. Висновки до розділу 3

Розробка мобільного додатку для орендарів і власників житла з системою рейтингів проходила через кілька ключових етапів, що забезпечили створення складного і водночас функціонального прогресивного вебдодатку (PWA).

Під час етапу проектування та побудови інформаційної бази даних була розроблена професійна структура на основі СУБД MySQL у середовищі Laravel. Створено таблиці для управління даними про користувачів, об’єкти житла, бронювання, рейтинги, повідомлення та інші пов’язані сутності. Фізична модель бази враховувала зовнішні ключі, типи зв’язків та необхідні атрибути для ефективного збереження й обробки даних.

Для технічної реалізації системи використано сучасний технологічний стек: Laravel (PHP), MySQL, Blade, Bootstrap 5, JavaScript. Це забезпечило надійну серверно-клієнтську архітектуру з підтримкою CSRF-захисту, автоматичного оброблення сесій, зручної маршрутизації та інтеграції з ORM (Eloquent). Для повної реалізації PWA-функцій були налаштовані manifest.json, service worker і можливість додавання застосунку на головний екран мобільних пристроїв.

Ключова функціональність модулями включає реєстрацію та авторизацію користувачів, створення й редагування оголошень про оренду, пошук і фільтрацію житла за різними параметрами, процеси бронювання, перегляд історії бронювань, взаємні рейтинги після завершення орендних угод, а також інструменти адміністративної модерації. Для всіх цих функцій були створені алгоритмічні блок-схеми й контролери для інтеграції моделей і користувацьких інтерфейсів.

Особливу увагу приділено інтерфейсу, адаптованому для мобільних пристроїв, а також локалізації українською мовою. Система рейтингів була впроваджена як важливий механізм покращення довіри між користувачами через одноразове оцінювання після завершення орендної угоди.

У результаті створене програмне забезпечення повністю відповідає сучасним вимогам мобільних додатків. Воно є кросплатформним, зрозумілим у використанні, захищеним, масштабованим і готовим до подальших оновлень. Крім того, передбачена можливість розширення функціоналу через додавання нових модулів, таких як система сповіщень, чат або інтеграція онлайн-платежів.

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУТАЦІЇ СИСТЕМИ

4.1. Тестування системи

Процес тестування мобільного додатку для орендодавців та орендарів житла із інтегрованою рейтинговою системою включав кілька етапів, спрямованих на забезпечення його стабільності, безпеки та відповідності очікуванням кінцевих користувачів.

На першому етапі проведено модульне тестування. Окремі функціональні частини застосунку, такі як маршрути, контролери, сервіси та моделі в Laravel, перевірялися з використанням фреймворку PHPUnit. Серед сценаріїв тестування: реєстрація користувачів, створення оголошень, додавання рейтингу, авторизація та редагування профілю. Це дозволило переконатися у коректності роботи кожного модуля й уникнути критичних помилок ще на початковому етапі розробки.

На наступному етапі здійснено інтеграційне тестування, яке перевіряло злагоджену роботу різних компонентів додатку. Серед них: Blade-шаблони користувацького інтерфейсу, HTTP-контролери, middleware, база даних MySQL, механізми авторизації Laravel та рейтинговий функціонал. Тестувалися такі процеси, як правильне оновлення рейтингу у профілі після відгуку, збереження оголошення у базі даних з необхідними параметрами та робота фільтрів під час пошуку житла.

Інтерфейс програми тестувався вручну на різних мобільних пристроях та розмірах екранів для перевірки адаптивності й зручності взаємодії. Інструменти Chrome DevTools і Lighthouse допомогли проаналізувати швидкодію додатку, час відгуку, ефективність рендерингу та відповідність вимогам PWA. Було підтверджено можливість коректного встановлення програми на смартфони, її функціонування в офлайн-режимі за допомогою

Service Worker та доступність ключових функцій навіть при слабкому інтернет-з’єднанні.

Для повноцінного тестування життєвого циклу користувача використовувався інструмент Laravel Dusk. Автоматизовані тести симулювали повний спектр дій користувачів: реєстрацію, пошук житла, бронювання та залишення відгуків по завершенню оренди. Тести проводилися в середовищі браузера для перевірки відповідності фактичної роботи додатку попередньо визначеним сценаріям.

Особлива увага приділялася тестуванню безпеки. Проведено аудит програми для виявлення вразливостей до XSS (скрипти в полях опису), SQL-ін’єкцій (через параметри URL), CSRF (з перевіркою токенів) і налаштування CORS-запитів. Усі форми в додатку забезпечені CSRF-токенами, а також системами валідації та обробки введених даних.

Фінальний етап — приймальне тестування — проводився за участю майбутніх користувачів додатку (зокрема студентів, викладачів та рієлторів). Вони оцінювали роботу інтерфейсу в реальних сценаріях, таких як створення профілю, публікація оголошень, перегляд доступного житла чи оцінювання взаємодій з іншими користувачами. Їхній зворотний зв'язок дозволив виявити недоліки в дизайні й покращити окремі аспекти UX (наприклад, розташування кнопок чи локалізацію тексту).

Завдяки всебічному підходу до тестування, що охоплює як автоматизовані, так і ручні методи, вдалося забезпечити високий рівень якості програмного продукту. Додаток став стабільним, безпечним і максимально зручним для щоденного користування як на десктопах, так і на мобільних пристроях.

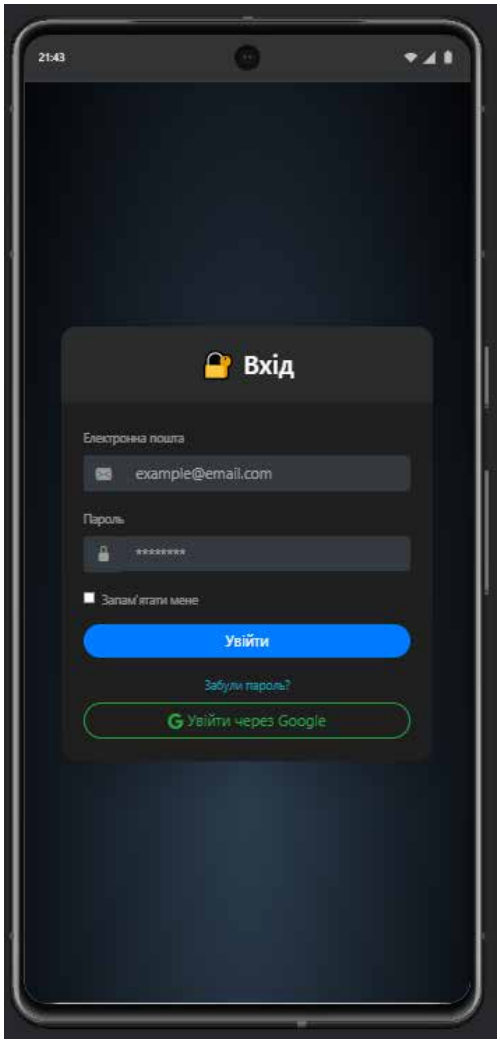
Рисунок 4.1 ілюструє екран авторизації мобільного додатку. На зображенні представлено інтерфейс форми входу, де користувач може вказати свою електронну пошту та пароль для входу в систему. Також передбачена опція “Запам’ятати мене” для зручності наступних авторизацій. Нижче розташовані кнопка входу та посилання для відновлення забутого пароля. Окрім стандартного входу за допомогою пошти, користувачу доступна альтернатива — авторизація через Google. Завдяки лаконічному дизайну та зрозумілій структурі, інтерфейс забезпечує швидкий та інтуїтивний доступ до основних функцій додатку.

роботи.

.

На рисунку 4.1 зображена реєстрація

Рисунок 4.2 ілюструє екран авторизації. На зображенні представлено інтерфейс форми входу, де користувач може вказати свою електронну пошту та пароль для входу в систему. Також передбачена опція “Запам’ятати мене” для зручності наступних авторизацій. Нижче розташовані кнопка входу та посилання для відновлення забутого пароля. Окрім стандартного входу за допомогою пошти, користувачу доступна альтернатива — авторизація через Google. Завдяки лаконічному дизайну та зрозумілій структурі, інтерфейс забезпечує швидкий та інтуїтивний доступ до основних функцій додатку.



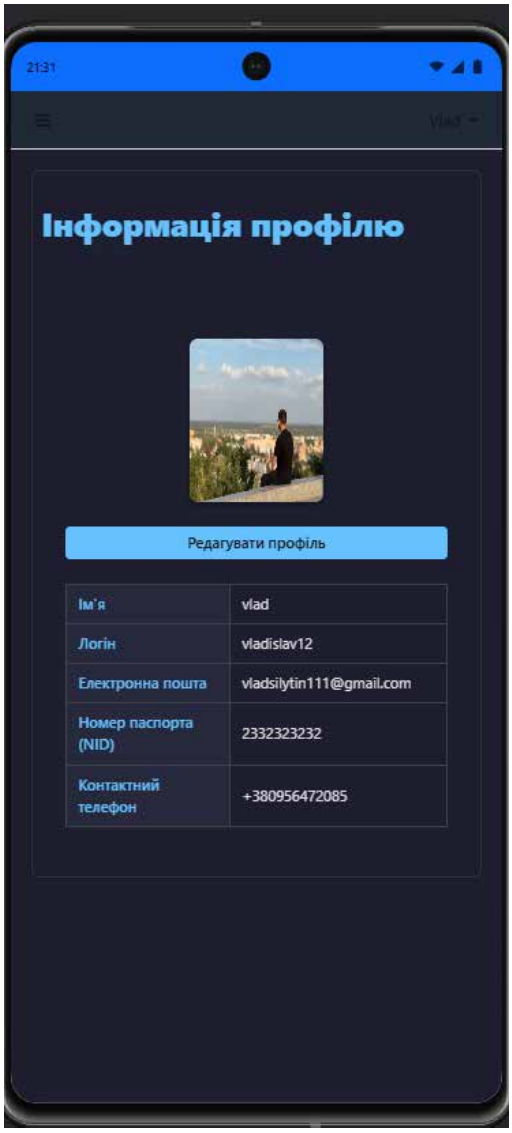
На рисунку 4.2 зображено вхід

Рисунок 4.3 демонструє сторінку історії бронювань. У верхній частині екрана відображено назву розділу “Історія бронювань” із зазначенням кількості записів у дужках. Основний блок містить таблицю з інформацією про всі попередні та поточні бронювання користувача.Записи таблиці відображають детальну історію проживання, з інформацією про точні адреси, дати перебування та відповідні фінансові дані. Завдяки структурованому й зрозумілому поданню інформації, користувач може швидко переглянути свої бронювання та проаналізувати історію взаємодій із сервісом..

.

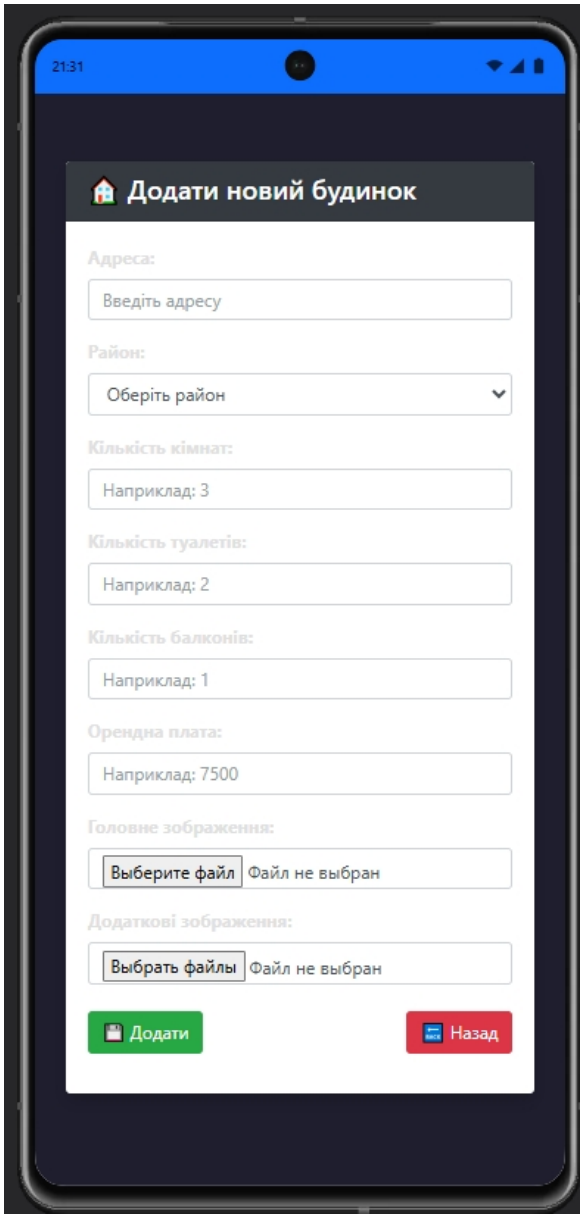
На рисунку 4.3зображено історію бронювань

Рисунок 4.4 демонструє екран із детальною інформацією профілю користувача у мобільному додатку. У верхній частині розміщено заголовок “Інформація профілю”, під яким знаходиться фотографія користувача, що персоналізує інтерфейс. Нижче розташована кнопка “Редагувати профіль”, яка дозволяє змінювати особисті дані. Основна частина екрану представлена у вигляді таблиці, де відображаються основні персональні дані: ім’я, логін, електронна пошта, номер паспорта (NID) та контактний телефон. Завдяки структурованому розміщенню інформації користувач може швидко переглянути свої дані та за потреби ініціювати їх редагування.



На рисунку 4.3 зображено профіль користувача

Рисунок 4.4 ілюструє екран додавання нового будинку у мобільному додатку. На зображенні представлена форма, що містить декілька полів для введення основної інформації про нерухомість: адресу, район (із можливістю вибору з випадаючого списку), кількість кімнат, туалетів, балконів, а також вартість оренди. Окремо виділені поля для завантаження головного зображення та додаткових фотографій об’єкта. Внизу форми розташовані дві кнопки: зелена “Додати” для підтвердження введених даних і червона “Назад” для повернення до попереднього екрана.



На рисунку 4.4 зображено додавання нового будинку

4.2. Вимоги до апаратного та програмного забезпечення

Для забезпечення стабільної роботи мобільного додатку Rent Trust, який виступає як прогресивний вебзастосунок (PWA) на базі серверного фреймворку Laravel, необхідно окреслити мінімальні та рекомендовані вимоги до апаратного й програмного забезпечення. Це сприятиме безперебійності функціонування системи, ефективній обробці запитів користувачів і можливості масштабування у разі зростання навантаження.

Огляд діаграми розгортання

На представленій діаграмі розгортання (рис. 4.4) відображено основні компоненти, що забезпечують роботу додатку Rent Trust:

Клієнтський пристрій (Browser/PWA): смартфони або планшети користувачів, які відкривають додаток через браузер або встановлюють його безпосередньо на пристрій.

CDN/Static Hosting: використовується для швидкої доставки статичних файлів, таких як зображення, CSS та JavaScript.

Laravel Web Server: серверна частина, побудована на PHP 7+ із використанням Laravel Framework, що обробляє HTTP-запити.

MySQL Database Server: база даних, яка зберігає всю інформацію про користувачів, оголошення, відгуки, повідомлення тощо.

Service Worker / Cache: забезпечує автономне функціонування застосунку, локальне кешування ресурсів та обробку push-сповіщень.

Адміністративна панель: вебінтерфейс для управління контентом і здійснення модерації оголошень.Огляд діаграми розміщення.

Діаграма розгортання містить графічнезображення процесорів, пристроїв і зв’язків між ними [10, с.34].

Рис. 4.5. Діаграма розгортання

Таблиця 4.1 – Апаратні та програмні вимоги

Компонент	**Мінімальні вимоги**	**Рекомендовані вимоги**
Користувацький пристрій	CPU: 2 ядра RAM: 2 ГБ Браузер: Chrome/Firefox	CPU: 4 ядра RAM: 4 ГБ Браузер: Chrome/Firefox/Edge
Сервер Laravel	CPU: 2 ядра RAM: 4 ГБ PHP 7.2, Composer, MySQL	CPU: 4 ядра RAM: 8 ГБ PHP 7.2+, Laravel , MySQL

MySQL-сервер	5 ГБ пам’яті 1 ядро	SSD-диск, 10+ ГБ пам’яті, реплікація, регулярні бекапи
PWA/Service Worker	Підтримка браузером HTTPS та Cache API	Повна підтримка Push API, Web App Manifest, Offline Cache
ОС розробника	Windows 10	Ubuntu 22.04+, macOS 12+, Windows 11
CI/CD (Vercel)	Git, локальні коміти	GitHub/GitLab CI, автоматичні деплої з Git

Мережева інфраструктура:

- Для комфортної роботи з PWA клієнтська пропускна здатність повинна складати не менше 5 Мбіт/с.
- Серверна пропускна здатність у середовищі продакшен має бути від 100 Мбіт/с.
- Затримка між клієнтом і сервером повинна становити до 100 мс, що забезпечить швидкий відгук системи.

Операційна система та середовище:

- Laravel версії 6 і вище
- PHP 7.2 або новіша версія
- MySQL 6.0 і новіше
- Composer 2.5+
- Сумісність із останніми версіями браузерів: Google Chrome, Firefox, Microsoft Edge

Резервне копіювання та моніторинг:

- Усі з'єднання між клієнтом і сервером мають виконуватися виключно через HTTPS.
- Регулярне оновлення залежностей (composer update, npm audit) для мінімізації вразливостей.
- Реалізація механізмів аутентифікації: CSRF, JWT або Session-based.
- Налаштування системи логування помилок через Laravel Log або використання зовнішніх сервісів, таких як Sentry.
- Організація резервного копіювання бази даних MySQL за допомогою cron-скриптів чи інструментів на кшталт mysqldump.

Дотримання наведених технічних вимог до апаратного та програмного забезпечення забезпечить стабільну й захищену роботу мобільного додатку Rent Trust. Використання сучасних вебтехнологій та підтримка PWA дозволяють системі масштабуватися, функціонувати в хмарних середовищах і забезпечувати зручність для користувачів на різних типах пристроїв.

4.3. Склад інсталяційного пакету

Оскільки програмне забезпечення Rent Trust створено у формі мобільного вебзастосунку з підтримкою технології Progressive Web App (PWA), користувачам не обов’язково завантажувати та встановлювати його на свої пристрої. Система функціонує за принципом клієнт-сервер і доступна через будь-який веббраузер, що забезпечує її сумісність як зі смартфонами, так і зі стаціонарними пристроями без потреби в додаткових налаштуваннях.

Процес впровадження системи починається з підготовки хостингового середовища для бекенд- та фронтенд-компонентів. Серверна сторона системи розроблена з використанням стеку Laravel + PHP і розгортається на вебсервері (наприклад, Apache або Nginx) з підтримкою PHP версії 7.x. База даних MySQL може бути налаштована окремо або використовуватися як частина хмарної інфраструктури.

Інтерфейс користувача розроблений за допомогою Blade-шаблонів Laravel, а його стиль створений із застосуванням Bootstrap з підтримкою

адаптивного дизайну та темного режиму. Крім того, система включає PWA-модулі, такі як manifest.json і service worker, що дозволяють користувачам додати застосунок на головний екран свого смартфона і використовувати його в режимі офлайн.

Функціонал аутентифікації, рейтингової системи, управління оголошеннями та адміністративної модерації реалізовано засобами Laravel із застосуванням контролерів, middleware та модулі для конфігурації ролей і доступів.

Основні елементи інсталяційного пакета:

- composer.json – файл, що визначає залежності PHP-проєкту, серед яких Laravel, пакети безпеки та базові сервіси.
- package.json – конфігурація JavaScript-залежностей, таких як Bootstrap, Laravel Mix та модулі для PWA.
- webpack.mix.js – файл для налаштування збірки фронтенду, включаючи компіляцію SASS і JavaScript.
- public/ – публічна директорія вебзастосунку, що містить згенеровані файли, такі як manifest.json і service-worker.js.
- resources/views/ – шаблони Blade для користувацького інтерфейсу, адаптовані до мобільного дизайну, таблиці, кнопки та профілі.
- resources/js/ – JavaScript-компоненти й функціонал, включно з реалізацією PWA та інтерактивними елементами.
- resources/sass/ – стилі CSS із підтримкою темної теми та мобільної адаптації.
- routes/web.php – маршрути для управління вебсистемою.
- app/Http/Controllers/ – контролери, що обробляють запити користувачів, виконують CRUD-операції та працюють з рейтингами.
- app/Models/ – моделі Eloquent для роботи з базою даних (наприклад, User, Housing, Rating, Region).

- database/migrations/ – файли для міграцій, які створюють структуру таблиць у базі даних.
- config/ – конфігураційні файли для сервісів, бази даних, сесій і поштових параметрів.
- .env – файл середовищної конфігурації, у якому зберігаються параметри доступу до бази даних.

У разі застосування систем автоматизованого розгортання, таких як GitHub Actions разом із Laravel Forge або Deployer, інсталяційний пакет може автоматично створюватися та розгортатися після внесення змін до основної гілки репозиторію. До стандартного списку команд зазвичай належать: composer install, npm run prod, php artisan migrate --force.

Розміщення та доступність:

Застосунок може бути розгорнутий:

- на віртуальному приватному сервері (VPS) з передвстановленими PHP, MySQL, Nginx/Apache;
- на хмарному Laravel-хостингу, що підтримує PWA (наприклад, Laravel Forge, Heroku);
- локально — для розробки через Laravel Valet, Homestead або Docker.

Після першого завантаження користувач має можливість встановити вебзастосунок на свій мобільний пристрій через PWA-механізм — без встановлення через App Store.

Отже, Інсталяційний пакет Rent Trust являє собою організований репозиторій Laravel-проєкту, що об’єднує серверну логіку, інтерфейс, оптимізований для мобільних пристроїв, та підтримує прогресивні веб-застосунки (PWA). Його вирізняє легкість перенесення, масштабування й автоматизації розгортання, що забезпечує зручність, високу продуктивність та сучасний користувацький досвід.

4.4. Висновки до розділу 4

У цьому розділі описано процес інсталяції, налаштування та тестування мобільного додатку Rent Trust, створеного у форматі прогресивного вебзастосунку (PWA) з використанням фреймворку Laravel. Деталізовано кроки підготовки середовища розробки, конфігурації локального веб-сервера, запуску проєкту та перевірки його основних функцій у реальних умовах експлуатації.

Результати тестування підтвердили коректну роботу ключових модулів системи, серед яких: реєстрація та автентифікація користувачів, створення й редагування оголошень про житло, пошук із фільтрами, надсилання запитів на бронювання і залишення відгуків після завершення оренди. Особливу увагу було приділено тестуванню адаптивності інтерфейсу, підтримки темної теми та можливості встановлення PWA на мобільні пристрої.

Аналіз технічних вимог засвідчив, що система є невибагливою до ресурсів. Для розробки та тестування використовувався локальний сервер на базі PHP і MySQL, що забезпечило повний контроль над усіма компонентами застосунку і гнучкість у налаштуванні.

Описано також структуру інсталяційного пакета. На відміну від традиційних десктопних програм, Rent Trust не потребує класичної інсталяції. Після компіляції та розгортання програма може функціонувати як звичайний вебсайт і бути встановленою на смартфон через браузер. Завдяки використанню сучасних вебтехнологій структура пакета лишається простою, зрозумілою й підготовленою до CI/CD-інтеграції в майбутньому.

Загалом Rent Trust продемонстрував високу стабільність роботи, відповідність технічним вимогам і готовність до повноцінної експлуатації. Прийняті архітектурні рішення забезпечують масштабованість системи, зручність підтримки та перспективи для подальшого розвитку..

ВИСНОВКИ

У рамках виконання дипломної роботи було створено повноцінний мобільний застосунок Rent Trust, розроблений для зручної взаємодії між орендарями та орендодавцями житлової нерухомості. Основною метою проєкту було забезпечення безпечного, прозорого й адаптивного рішення, що сприяє ефективному процесу оренди завдяки системам оцінювання, рейтингу та зворотного зв’язку між користувачами. Заявлені цілі були повністю досягнуті: реалізований застосунок відповідає заданим функціональним і нефункціональним вимогам, демонструє стійку роботу та готовий до практичного використання.

У процесі розробки проведено аналіз існуючих сервісів оренди житла. Було виявлено їхні недоліки, серед яких недостатня зручність використання на мобільних пристроях і обмежена прозорість стосовно надійності користувачів. На основі цього аналізу була спроєктована оригінальна архітектура, орієнтована на мобільне використання. Завдяки формату Progressive Web App (PWA) користувачі можуть встановлювати додаток безпосередньо через браузер, обминаючи традиційні платформи, такі як App Store та Google Play.

Застосунок створений на сучасному технологічному стеку: для серверної частини використано PHP із фреймворком Laravel, для зберігання даних — MySQL, а інтерфейс реалізовано за допомогою Blade і Bootstrap із підтримкою темної теми та мобільної адаптивності. Основні функції включають реєстрацію й автентифікацію користувачів, створення та перегляд оголошень із системою фільтрів, рейтинги й відгуки, а також інструменти для адміністративної модерації контенту.

Особлива увага приділена підтримці повного спектра PWA-функцій: це включає реалізацію manifest.json, service worker, кешування контенту й можливість встановлення додатку на домашній екран смартфона. Такий підхід

забезпечує швидкий доступ до платформи та високу продуктивність навіть за низької швидкості інтернет-з’єднання.

Кожен розділ проєкту логічно пов’язаний із наступним: від формулювання завдання й аналізу предметної області до моделювання бази даних, розробки інтерфейсу, побудови діаграм класів і кооперацій, а також створення блок-схем процесів. Проведене тестування засвідчило стабільну роботу системи, відповідність очікуваним сценаріям використання й коректну інтеграцію між усіма компонентами.

Перспективи розвитку застосунку залишаються значними. Серед потенційних напрямків удосконалення — впровадження інтеграції з платіжними сервісами для бронювання житла, розробка чатів для комунікації між орендарями та орендодавцями, пошук житла за допомогою геолокації, автоматичні сповіщення та адаптація застосунку до повноцінної нативної мобільної версії.

Загалом, розроблене програмне забезпечення характеризується технічною зрілістю, зручністю для кінцевого користувача й відповідністю сучасним стандартам у сфері цифрових послуг. Rent Trust поєднує інтуїтивний дизайн, адаптивність, функціональність і безпеку, що робить його перспективною платформою для ефективної взаємодії на ринку оренди житла.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Tkachuk, V. "PWA, як перспективний напрямок об'єднання веб та мобільних технологій." COMPUTER-INTEGRATED TECHNOLOGIES: EDUCATION, SCIENCE, PRODUCTION 46 (2022): 83-87.
2. Кулябов Д.С. Введение в формальные методы описания бизнес-процессов : [учеб.пособ.] / Д.С. Кулябов, А.В. Королькова. – М. : РУДН, 2008. – 173 с.
3. Гайна Г. А. Основи проектування баз даних: навч. посіб. Київ: КНУБА, 2005. 204 с.
4. Bulatetskaya, L.,et al. "Методичні особливості вивчення концептуального проектування баз даних при підготовці майбутніх фахівців."COMPUTER-INTEGRATED TECHNOLOGIES: EDUCATION, SCIENCE, PRODUCTION 41 (2020): 5-9.
5. Кісь, Я. П., Л. Б. Чирун, and В. М. Фольтович. "Особливості застосування методу контент-аналізу для опрацювання інтернет-газети." Вісник Національного університету Львівська політехніка. Інформаційні системи та мережі 805 (2014): 124-136.
6. Полотай, О. І. "Використання діаграми класів UML для запровадження освітніх ІТ-проектів у ВНЗ." *Торгівля, комерція, підприємництво* 13 (2011): 111-115.
7. Schmuller J. Sams Teach Yourself UML in 24 Hours 3rd Edition. Sams Publishing, 2004.504 p.
8. Кулябов Д.С. Введение в формальные методы описания бизнес-процессов : [учеб.пособ.] / Д.С. Кулябов, А.В. Королькова. – М. : РУДН, 2008. – 173 с.
9. Єфремов, М. Ф., Ю. М. Єфремов, and В. М. Єфремов. "Проектування програмного забезпечення з використанням UML." (2016).

10. Schmuller J. Sams Teach Yourself UML in 24 Hours 3rd Edition. Sams Publishing, 2004.504 p