НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри
_____комп'ютерних наук_____
(назва кафедри)

_____ /_____Голуб Б.Л._____/
(підпис) (ПІБ)

"___" _____ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему

"Програмне забезпечення для пошуку та оренди житла»

Спеціальність 121 – «Інженерія програмного забезпечення»

Гарант освітньої програми
_____к.т.н, доцент_____ _____ _____Вайганг Г.О._____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи
___асистент _____ _____ _Баранова Т. А._
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Консультант бакалаврської кваліфікаційної роботи
___к.т.н, доцент _____ _____ _ Даков С.Ю.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ _____Дзюба Назар Євгенійович__
(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____комп'ютерних наук_____
(назва кафедри)

___к.т.н., доцент_____ _____Голуб Б.Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"___" _____ 2025 р.

З А В Д А Н Н Я
на виконання бакалаврської кваліфікаційної роботи студенту
Дзюба Назар Євгенійович

Спеціальність 121 – «Інженерія програмного забезпечення»

1. Тема бакалаврської кваліфікаційної роботи «Програмне забезпечення для пошуку та орен житла» затверджена наказом ректора НУБіП України від 16.12.2024 №2248«с»
2. Термін подання завершеної роботи на кафедру_____
(рік, місяць, число)
3. Вихідні дані: надання інформації про доступні для оренди житлові місця, дати бронювання.
4. Перелік питань, що розглядаються:
   - Аналіз проблемної області
   - Моделювання предметної області
   - Проєктування програмної системи
   - Впровадження та експлуатація системи

Дата видачі завдання "__16__"____грудня_____2024__ р.

Керівник бакалаврської кваліфікаційної роботи
___асистент _____ _____ _Баранова Т. А._
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Консультант бакалаврської кваліфікаційної роботи
___к.т.н, доцент _____ _____ _ Даков С.Ю.
(науковий ступінь та вчене звання) (підпис) (ПІБ

Завдання прийняв до виконання _____ Дзюба Н. Є.

(підпис) (ПІБ)

ЗМІСТ

ВСТУП

Розвиток інформаційних технологій докорінно змінив спосіб взаємодії людей зі світом, зокрема з ринком нерухомості. Оренда житла, яка раніше вимагала значного часу та зусиль — через газетні оголошення, агентства нерухомості чи особисті контакти — стала значно зручнішою завдяки цифровим платформам. У цьому контексті програмні рішення для пошуку та оренди житла стають не лише актуальними, а й необхідними.

Актуальність. Зі зростанням мобільності населення попит на короткострокову та довгострокову оренду житла неухильно зростає. Водночас користувачі очікують зручності, швидкості та персоналізації в процесі пошуку та бронювання житла. Існуючі платформи часто мають обмеження: застарілі інтерфейси, недостатні фільтри пошуку, відсутність інтерактивних елементів, таких як карти, або механізмів рекомендацій. Це створює потребу в сучасному, функціональному та орієнтованому на користувача програмному забезпеченні, яке б задовольняло потреби як орендодавців, так і орендарів.

Об'єктом цього дослідження є процес цифровізації та автоматизації послуг пошуку та оренди житла за допомогою веб-технологій.

Метою цієї бакалаврської роботи є покращення сфери оренди житла, розробивши функціональну веб-програмну систему для пошуку та оренди житла, яка дозволяє користувачам:

- розміщувати та керувати оголошеннями про оренду з детальною інформацією;
- шукати та фільтрувати оголошення на основі різних параметрів;
- переглядати доступне житло на інтерактивній карті;

- отримувати персоналізовані рекомендації на основі своїх уподобань та попередньої активності.

Для досягнення цілей цієї роботи було проведено комплекс дослідницьких та розробницьких робіт та визначено **завдання** роботи. Процес розпочався з поглибленого аналізу існуючих цифрових платформ, призначених для пошуку та оренди житла, що дозволило визначити їхні сильні сторони та спільні обмеження. На основі цього було сформульовано ключові вимоги до майбутньої системи, включаючи як функціональні можливості, так і нефункціональні очікування, такі як продуктивність, надійність та простота використання. Наступний етап включав проєктування загальної архітектури веб-застосунку, включаючи структуру реляційної бази даних та взаємодію між компонентами. Особливу увагу було приділено визначенню ролей користувачів, розмежуванню прав та функціональності, доступних власникам нерухомості та потенційним орендарям. Після етапу проєктування зусилля з розробки були зосереджені на впровадженні основних системних модулів. До них належали функції для безпечної реєстрації та входу користувачів, створення та управління списками житла, розширений пошук з кількома фільтрами, інтеграція інтерактивної карти для візуалізації доступної нерухомості та механізм рекомендацій для підвищення релевантності пошуку. Також було враховано процес бронювання, що дозволило подавати запити та спілкуватися між користувачами.

Цей програмний комплекс реалізовано з використанням сучасних інструментів: PHP-фреймворку Symfony для бекенду, MySQL для зберігання даних та стандартних фронтенд-технологій (HTML, CSS, JavaScript) з інтеграцією картографічних сервісів (таких як Google Maps або Leaflet) для географічної візуалізації.

Результатом роботи є робочий веб-додаток, який надає зручний та ефективний інструмент для оренди житла, і який може слугувати основою для подальшого розвитку та впровадження в реальних умовах.

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Опис предметної області

Ринок оренди нерухомості зазнав значних змін з розвитком цифрових технологій. Традиційні методи пошуку орендованого житла, такі як оголошення в газетах, фізичні дошки оголошень або покладаючись на посередників, поступово замінюються онлайн-платформами, які пропонують більш зручні, ефективні та доступні рішення. Однак, незважаючи на широку цифровізацію, багато існуючих систем досі не повністю задовольняють потреби користувачів, що постійно змінюються, з точки зору функціональності, персоналізації та користувацького досвіду.

Однією з ключових проблем є фрагментація доступної інформації. Потенційні орендарі часто змушені переглядати численні веб-сайти або додатки в пошуках відповідних оголошень, багато з яких не мають актуальних даних або пропонують обмежені можливості фільтрації. Це збільшує час, необхідний для пошуку відповідного житла, і знижує загальну ефективність процесу. Крім того, орендодавці часто стикаються з труднощами під час спроби опублікувати оголошення, керувати доступністю або ефективно спілкуватися з орендарями.

Ще однією нагальною проблемою є відсутність інтелектуальних механізмів пошуку. Більшість платформ покладаються виключно на статичні фільтри (такі як місцезнаходження, ціна або кількість кімнат), які не адаптуються до вподобань чи поведінки користувачів. Як результат, користувачі можуть пропустити відповідні оголошення, які в іншому випадку відповідали б їхнім інтересам. Відсутність систем рекомендацій обмежує персоналізацію та може призвести до розчарування під час процесу пошуку [1].

Крім того, багатьом існуючим рішенням бракує інтерактивних елементів, які підвищують зручність використання. Наприклад, можливість

перегляду нерухомості на карті не завжди доступна або реалізована в обмеженій формі. Це зменшує просторове розуміння доступних варіантів та ускладнює процес прийняття рішень. Крім того, деякі системи не оптимізовані для мобільних пристроїв або мають застарілі та неінтуїтивно зрозумілі інтерфейси.

Безпека та захист даних також є критичними проблемами. Недостатні механізми автентифікації, недостатнє шифрування або відсутність інструментів модерації можуть наражати користувачів на шахрайство або неправомірне використання персональних даних. Ці ризики ще більше знижують довіру до платформ та створюють додаткові бар'єри для користувачів.

Враховуючи поточну ситуацію, існує очевидна потреба в сучасному, багатофункціональному та зручному для користувача рішенні, яке вирішує вищезазначені проблеми. Добре розроблений програмний продукт для пошуку та оренди житла повинен інтегрувати інтелектуальні рекомендації, інтерактивні карти, оновлення даних у режимі реального часу та засоби безпечного зв'язку. Така система значно покращить загальний досвід як для орендодавців, так і для орендарів, а також оптимізує процес оренди та мінімізує ризики [1].

Предметом цього дослідження є процеси та технології, пов'язані з пошуком та орендою житлової нерухомості через цифрові платформи. Ця галузь охоплює різні галузі, включаючи нерухомість, інформаційні технології, дизайн користувацького інтерфейсу та управління даними. Основними зацікавленими сторонами в цій системі є фізичні або юридичні особи, які пропонують житло в оренду (орендодавці або агентства), та ті, хто шукає житло (орендарі).

По суті, процес оренди включає кілька ключових етапів: розміщення об'єкта нерухомості, публікація деталей (таких як ціна, місцезнаходження, характеристики та наявність), параметри пошуку та фільтрації, спілкування між орендарем та орендодавцем, а в деяких випадках і бронювання або

резервування нерухомості. Традиційно ці процеси проводилися офлайн, включаючи оформлення документів, телефонні дзвінки та фізичні візити. Однак, з розвитком цифрових платформ, весь робочий процес тепер можна оптимізувати в рамках одного програмного рішення.

Сучасні платформи оренди житла прагнуть спростити та автоматизувати ці взаємодії. Орендодавці повинні мати можливість створювати оголошення про нерухомість, завантажувати медіафайли (фотографії, документи), редагувати інформацію та відстежувати відповіді. Орендарям, з іншого боку, потрібні інтуїтивно зрозумілі інструменти для перегляду оголошень, застосування фільтрів пошуку та оцінки пропозицій на основі візуального контенту, місцезнаходження та відгуків користувачів. Ключовою функцією, якої все частіше очікують користувачі, є інтеграція інтерактивних карт, що дозволяє переглядати нерухомість географічно для кращого контексту та зручності [2].

Ще одним важливим аспектом предметної області є персоналізація. Зі зростанням обсягу доступної нерухомості користувачі стикаються з проблемою інформаційного перевантаження. Системи рекомендацій, що працюють на основі даних користувачів, аналізу поведінки та машинного навчання, стають важливими для підвищення релевантності та задоволеності.

Предметна область також включає питання безпеки, захисту даних та дотримання законодавства. Оскільки користувачі часто діляться особистою інформацією та взаємодіють з незнайомими особами, система повинна включати механізми автентифікації, інструменти модерації та безпечні методи обробки даних [3].

У контексті цієї роботи предметна область досліджується з метою проєктування та розробки функціонального, безпечного та зручного веб-додатку, який відображає реальні вимоги та сучасні тенденції в оренді цифрового житла.

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

Таблиця 1.1

Опис атрибутів класів предметної області

Клас предметної області	**Атрибут**	**Опис**
Орендодавець	Назва об’єкта	Назва нерухомості або її опис (наприклад, “1-кімнатна квартира”)
	Адреса	Місцезнаходження житла (місто, район, вулиця)
	Ціна за добу	Вартість оренди за добу для даної нерухомості
	Опис	Короткий опис об’єкта (розмір кімнат, особливості тощо)
	Фото	Зображення житла (фото квартир, кімнат або будинків)
Орендар	Бажана локація	Місця або райони, в яких орендар шукає житло
	Бюджет	Максимальна сума, яку орендар готовий витратити на оренду
	Тип житла	Тип нерухомості, яку орендар шукає (квартира, будинок, кімната)
	Дата заїзду	Дата початку оренди (якщо орендар шукає на певний період)
Оголошення	Заголовок	Коротка назва оголошення про оренду (наприклад, “1-кімнатна квартира в центрі”)
	Опис	Детальний опис житла, його переваг і характеристик
	Дата публікації	Дата, коли оголошення було опубліковане на платформі
	Статус	Статус оголошення (активне, заброньоване, завершене)
	Доступність	Підтвердження, чи є житло доступним на зазначений період

1.2 Огляд існуючих рішень

За останні роки ринок цифрових платформ, що сприяють пошуку та оренді житла, значно розширився. Кілька добре зарекомендували себе сервісів встановили стандарти з точки зору функціональності, користувацького досвіду та технологічної реалізації. Огляд цих платформ допомагає визначити їхні сильні сторони, обмеження та сфери, які можна покращити або адаптувати під час розробки нової, більш орієнтованої на користувача системи.

Airbnb є світовим лідером на ринку короткострокової оренди, що об'єднує власників нерухомості (господарів) з мандрівниками (гостями), які шукають унікальне та доступне житло. Запущений у 2008 році, Airbnb змінив спосіб бронювання та пропозиції оренди, пропонуючи альтернативу традиційним готелям, дозволяючи окремим особам здавати в оренду свої будинки, квартири або навіть нетрадиційні простори, такі як будиночки на деревах та замки (Рис. 1.1) [4].

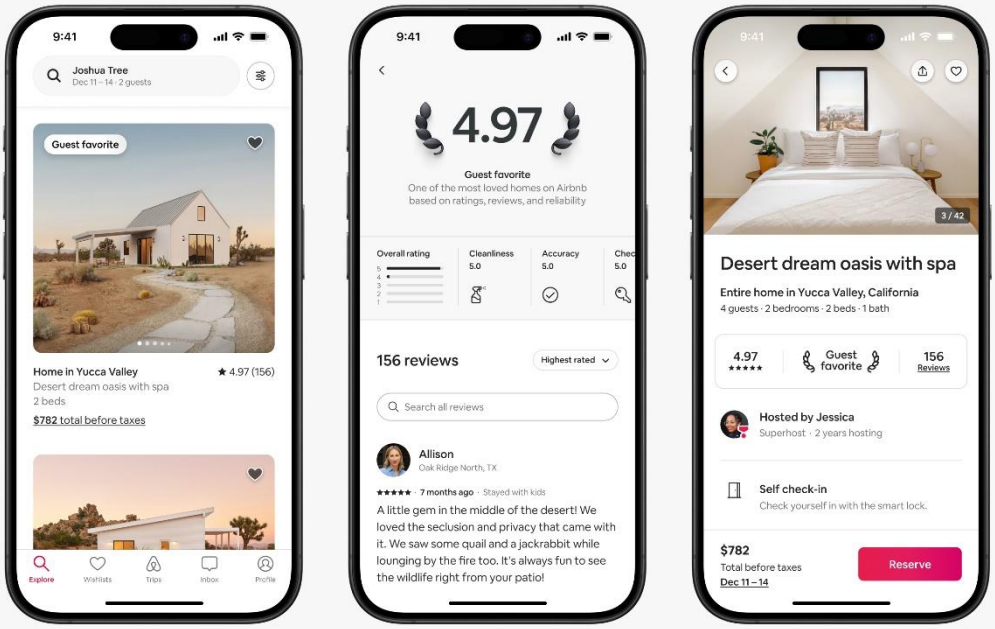

Рис. 1.1 Інформаційна система “Airbnb”

Однією з ключових переваг Airbnb є широкий вибір житла, що задовольняє різні бюджети та вподобання. Від бюджетних спільних кімнат до розкішних маєтків, платформа дозволяє користувачам знаходити варіанти житла, які відповідають їхнім потребам. Платформа значною мірою спирається на контент, створений користувачами, зокрема на відгуки та рейтинги. Ця функція сприяє довірі серед користувачів, оскільки гості можуть читати відгуки про попередні перебування, щоб допомогти їм приймати обґрунтовані рішення щодо своїх бронювань. Господарі, у свою чергу, також можуть покладатися на рейтинги, щоб переконатися, що вони відповідають очікуванням своїх гостей.

Потужна пошукова система Airbnb дозволяє користувачам фільтрувати нерухомість за різними критеріями, такими як розташування, ціна, зручності та тип житла, що полегшує їм пошук саме того, що вони шукають. Функція інтерактивної карти ще більше покращує цей пошук, пропонуючи візуальне представлення оголошень та їхньої близькості до ключових визначних пам'яток, транспортних вузлів та районів. Цей режим перегляду карти дозволяє користувачам приймати рішення на основі місцезнаходження, які найкраще відповідають їхнім планам подорожі [4].

Платформа спрощує процес бронювання та оплати, роблячи транзакції безпечними та прозорими. Airbnb обробляє платежі між гостями та господарями, гарантуючи, що господарі отримують компенсацію після перебування гостя. Ця безперебійна система усуває значну частину невизначеності, яка часто пов'язана з традиційними договорами оренди.

Для господарів Airbnb надає різноманітні інструменти для ефективного управління своїми оголошеннями. Ці інструменти допомагають встановлювати наявність, коригувати ціни та спілкуватися з потенційними гостями. Господарі також отримують доступ до цінної інформації та аналітики, що дозволяє їм оптимізувати свої пропозиції на основі уподобань гостей та моделей бронювання.

Airbnb вжила заходів для створення відчуття спільноти, дозволяючи господарям та гостям взаємодіяти безпосередньо через платформу. Вона включає такі функції, як обмін повідомленнями та системи перевіреної ідентифікації, які сприяють безпечнішому та надійнішому середовищу. Система оцінювання платформи додатково забезпечує підзвітність, оскільки як господарі, так і гості залишають відгуки після кожного перебування.

Завдяки своєму мобільному додатку Airbnb ще більше спростив для користувачів перегляд оголошень, спілкування з господарями та керування бронюванням з будь-якого місця. Додаток ще більше покращив доступність платформи, дозволяючи мандрівникам планувати та бронювати житло в дорозі.

З моменту свого створення Airbnb змінив традиційну індустрію гостинності, пропонуючи більш персоналізовану та економічно ефективну альтернативу готелям. Він надав власникам нерухомості можливість отримувати додатковий дохід, здаючи в оренду маловикористовувані приміщення. Незважаючи на свій успіх, Airbnb стикається з проблемами, зокрема з точки зору регуляторних питань та потенційного впливу на місцеві ринки житла. У деяких містах короткострокова оренда викликає занепокоєння щодо зростання цін на оренду та зменшення доступності довгострокового житла [4].

Незважаючи на ці проблеми, Airbnb продовжує впроваджувати інновації, розширюючи свої послуги за межі простого проживання. Тепер він пропонує такі «враження», як екскурсії з гідом або кулінарні майстер-класи, надаючи гостям більш повний досвід подорожей. Крім того, Airbnb також вийшов на ринок довгострокової оренди, позиціонуючи себе як конкурента не лише готелям, але й традиційним платформам оренди.

На завершення, успіх Airbnb полягає в його здатності пропонувати різноманітну та гнучку платформу розміщення, використовуючи відгуки користувачів, технології та сильне почуття спільноти для задоволення потреб як господарів, так і гостей. Постійні зусилля платформи щодо диверсифікації

своїх пропозицій та вирішення нових викликів підкреслюють її адаптивність на конкурентному ринку оренди житла для відпочинку.

Розглянемо інше програмне рішення на ринку.

Booking.com – одне з найбільших і найпопулярніших онлайн-турагентств (OTA) у світі, що спеціалізується на наданні варіантів проживання для мандрівників. Заснована в 1996 році в Нідерландах, платформа розширилася і пропонує широкий спектр послуг, включаючи бронювання готелів, оренду апартаментів та інші види короткострокового проживання. Вона працює по всьому світу, з'єднуючи мільйони мандрівників з постачальниками послуг проживання (Рис. 1.2) [5].

Рис. 1.2 Інформаційна система “Booking”

Видатною особливістю Booking.com є його комплексна пошукова система, яка дозволяє користувачам шукати житло на основі різних параметрів, таких як ціна, розташування, зручності та тип житла. Платформа також пропонує широкий вибір варіантів проживання, від бюджетних хостелів до розкішних готелів, апартаментів і навіть будинків для відпочинку. Таке різноманіття дозволяє їй обслуговувати широкий спектр клієнтів, від

мандрівників з обмеженим бюджетом до тих, хто шукає висококласний відпочинок.

Функціональність пошуку Booking.com покращена надійними інструментами фільтрації, які дозволяють користувачам звузити свої варіанти на основі певних критеріїв. До них належать кількість номерів, зірковий рейтинг готелю, ціна за ніч та оцінки гостей. Крім того, платформа відображає детальну інформацію про помешкання, включаючи фотографії, зручності та відгуки гостей, що дозволяє користувачам легко приймати обґрунтовані рішення.

Однією з ключових переваг платформи є її здатність надавати інформацію про наявність житла в режимі реального часу та миттєве підтвердження бронювання. Користувачі можуть швидко та безпечно бронювати житло, знаючи, що їхнє бронювання підтверджено миттєво, що особливо важливо для бронювань в останню хвилину. Платформа також дозволяє користувачам скасовувати або змінювати бронювання з гнучкими умовами, що приваблює мандрівників, які шукають більшої зручності.

Ще однією унікальною особливістю Booking.com є програма лояльності «Genius», яка пропонує знижки, спеціальні пропозиції та бонуси постійним користувачам. Це сприяє утриманню клієнтів, винагороджуючи тих, хто здійснює повторні бронювання на платформі.

Веб-сайт та мобільний додаток пропонують зручний інтерфейс, що сприяло їхньому широкому успіху. Додаток платформи особливо добре сприйнятий, дозволяючи користувачам керувати своїми бронюваннями, знаходити житло поблизу та отримувати сповіщення в режимі реального часу під час подорожі. Це робить Booking.com дуже зручним як для туристів, так і для ділових мандрівників [5].

Хоча Booking.com відомий переважно бронюванням готелів, він також пропонує варіанти авіаквитків, оренди автомобілів і навіть трансферу з/до аеропорту. Цей широкий спектр послуг позиціонує платформу як універсальний центр для потреб, пов'язаних з подорожами. Співпраця

компанії з тисячами постачальників послуг розміщення, від великих готельних мереж до незалежних власників, гарантує користувачам доступ до широкого спектру об'єктів нерухомості за різними ціновими категоріями.

Незважаючи на свою популярність, Booking.com стикається з конкуренцією з боку інших основних гравців туристичної галузі, таких як Expedia, Airbnb та TripAdvisor. Кожна платформа має свої унікальні пропозиції, і хоча Booking.com часто віддають перевагу для бронювання готелів та житла, такі платформи, як Airbnb, пропонують більш різноманітні варіанти, включаючи приватну оренду та місцевий досвід.

Що стосується викликів, Booking.com також має подолати різні регуляторні перешкоди в різних регіонах, оскільки багато країн мають специфічні правила щодо роботи онлайн-турагентств та використання даних клієнтів.

На завершення, Booking.com залишається домінуючим гравцем в онлайн-туристичній галузі, пропонуючи широкий вибір варіантів розміщення, конкурентні ціни та зручний функціонал. Завдяки різноманітним пропозиціям, миттєвому підтвердженню бронювання та програмі винагород лояльності, він продовжує залишатися найкращим вибором для мільйонів мандрівників у всьому світі.

1.3 Постановка завдання

Основним обов'язком команди з управління нерухомістю є відстеження важливих показників, пов'язаних з орендою житла, таких як наявність, ціни, термін оренди та кількість активних бронювань. Для цього програмне забезпечення повинно мати можливість зберігати та керувати базою даних з ключовою інформацією, необхідною для створення персоналізованих сповіщень на основі статусу нерухомості.

Основна функціональність системи полягає в тому, щоб надати користувачам (як орендарям, так і орендодавцям) легкий доступ до відповідних даних, включаючи:

- стан наявності нерухомості (чи є оголошення доступним, чи забронюванним);
- вартість оренди за ніч та пов'язані з цим витрати;
- кількість запитів на бронювання та завершених оренд;
- історичні дані про наповненість нерухомості;
- оцінки та відгуки, залишені попередніми орендарями.

Програма буде розроблена в зручному для користувача інтерфейсі на основі діалогів з використанням системи меню для навігації. Це дозволяє користувачам вводити інформацію про нерухомість, таку як описи, фотографії та ціни, та миттєво отримувати оновлення або рекомендації щодо статусу оголошень. Крім того, користувачі можуть взаємодіяти з системою в режимі реального часу, коригуючи умови оренди, додаючи нові оголошення або керуючи існуючими.

Система також пропонуватиме функцію сповіщень, яка сповіщатиме користувачів про зміни, такі як нове бронювання або зміна ціни оренди. Крім того, орендодавці зможуть скасовувати або змінювати раніше зареєстровані операції, такі як скасування бронювання або оновлення інформації про нерухомість. Ці функції розроблені таким чином, щоб бути доступними та ефективними, гарантуючи, що всі залучені сторони можуть легко керувати своєю орендою житла.

Налаштувавши програмне забезпечення таким чином, користувачі зможуть безперешкодно відстежувати свою діяльність з оренди житла та

отримувати негайні оновлення щодо ключових показників, важливих для забезпечення безперебійного процесу оренди.

1.4 Функціональні та нефункціональні вимоги

Функціональні вимоги:

1. система повинна дозволяти користувачам (як орендарям, так і орендодавцям) реєструватися та входити на платформу. Процес реєстрації повинен вимагати базової інформації, такої як електронна пошта, пароль та контактні дані. Користувачі повинні мати можливість безпечно входити в систему та виходити з неї;
2. орендодавці повинні мати можливість створювати, редагувати та видаляти оголошення про нерухомість. Кожне оголошення повинно містити такі деталі, як адреса нерухомості, опис, ціна оренди за день, доступні зручності та зображення. Орендодавці можуть налаштувати доступність нерухомості, вказавши дати, коли нерухомість доступна для оренди;
3. платформа повинна забезпечувати надійну функцію пошуку для орендарів, щоб вони могли знаходити доступну нерухомість на основі різних критеріїв, таких як місцезнаходження, ціновий діапазон, тип житла та зручності. Орендарі повинні мати можливість застосовувати розширені фільтри для уточнення результатів пошуку, такі як конкретні типи кімнат, варіанти, де дозволено проживання з домашніми тваринами, та функції доступності;
4. система повинна включати інтерактивну карту, яка візуально відображає розташування доступної нерухомості. Користувачі повинні мати можливість збільшувати та зменшувати масштаб, натискати на значки нерухомості та переглядати деталі нерухомості безпосередньо на карті;

5. система повинна пропонувати орендарям рекомендації на основі їхньої історії пошуку, уподобань та оцінок попередніх орендарів. Ця персоналізована функція допомагає орендарям швидко знаходити нерухомість, яка відповідає їхнім потребам;
6. орендарі повинні мати можливість подавати запити на бронювання нерухомості, яка їх цікавить. Система повинна повідомляти орендодавців про нові запити та дозволяти їм приймати або відхиляти бронювання. Після підтвердження бронювання інформація про наявність нерухомості повинна автоматично оновлюватися;
7. платформа повинна підтримувати обробку онлайн-платежів, щоб Орендарі могли безпечно оплачувати орендну плату. Платіжна система повинна підтримувати різні способи оплати, включаючи кредитні/дебетові картки та цифрові гаманці;
8. після закінчення терміну оренди орендарі повинні мати можливість оцінювати нерухомість та залишати відгуки. Це допоможе майбутнім орендарям приймати обґрунтовані рішення на основі досвіду інших. Аналогічно, орендодавці повинні мати можливість оцінювати орендарів після бронювання;
9. користувачі (як орендарі, так і орендодавці) повинні мати можливість керувати своїми профілями, оновлювати особисту інформацію, змінювати паролі та завантажувати фотографії профілю;
10. система повинна надсилати користувачам сповіщення про важливі події, такі як підтвердження бронювання, скасування, нові повідомлення та оновлення інформації про наявність нерухомості;
11. платформа повинна мати адміністративну панель для моніторингу та управління всією системою, включаючи облікові записи користувачів, списки об'єктів нерухомості та транзакції.

Адміністратор повинен мати можливість переглядати звіти, вирішувати суперечки та модерувати контент.

Нефункціональні вимоги:

1. система повинна мати інтуїтивно зрозумілий та зручний інтерфейс, що гарантує, що як орендарі, так і орендодавці можуть легко орієнтуватися на платформі без необхідності технічних знань. Інтерфейс має бути зрозумілим, адаптивним та простим для розуміння;
2. система має бути оптимізована для одночасної роботи з великою кількістю користувачів, з швидким часом відгуку на пошук об'єктів нерухомості, подання бронювань та оновлення оголошень. Продуктивність має бути стабільною навіть у години пікового навантаження;
3. програмне забезпечення має бути масштабованим, щоб враховувати майбутнє зростання. Зі зростанням кількості користувачів та об'єктів нерухомості платформа повинна мати можливість масштабуватися горизонтально та вертикально без значного зниження продуктивності;
4. дані користувачів, включаючи особисту інформацію та платіжні реквізити, повинні безпечно зберігатися та передаватися. Система повинна впроваджувати шифрування для конфіденційних даних та дотримуватися галузевих стандартів захисту даних, включаючи використання SSL/TLS для безпечного зв'язку;
5. система повинна бути доступна цілодобово, з мінімальним часом простою. Його слід розміщувати на надійних серверах з регулярним резервним копіюванням, щоб забезпечити високу доступність та зменшити ризик втрати даних;
6. платформа має бути сумісною з різними браузерами (Chrome, Firefox, Safari тощо) та пристроями (настільними комп’ютерами,

планшетами та смартфонами). Мобільна версія має бути адаптивною та забезпечувати подібний користувацький досвід до версії для настільних комп’ютерів.

1.5 Вимоги до інтерфейсу користувача

Інтерфейс користувача (UI) платформи пошуку та оренди житла є важливим для забезпечення безперебійного, інтуїтивно зрозумілого та приємного досвіду як для орендодавців, так і для орендарів. Він має бути розроблений таким чином, щоб користувачі могли легко орієнтуватися на платформі та отримувати доступ до необхідних функцій. Інтерфейс користувача повинен пріоритезувати простоту, доступність та адаптивність, щоб забезпечити оптимальний досвід на всіх пристроях.

Чітка та зручна система навігації має вирішальне значення, дозволяючи користувачам без зусиль досліджувати різні розділи, такі як оголошення про нерухомість, профілі, управління бронюванням та налаштування облікового запису. Макет має бути добре організованим, з легкими для розуміння підписами для ключових функцій, таких як пошук, створення оголошень та налаштування користувача. Також важливо, щоб дизайн адаптувався до різних розмірів екранів, забезпечуючи послідовний та функціональний досвід на настільних комп'ютерах, планшетах та смартфонах [6].

Оголошення про нерухомість повинні бути візуально привабливими, представляючи ключові деталі, такі як назва нерухомості, ціна, опис, доступні зручності та високоякісні зображення у зручному для читання форматі. Оголошення можуть відображатися у вигляді сітки або списку, з можливістю для користувачів сортувати та фільтрувати результати на основі різних критеріїв, таких як ціна, рейтинги або місцезнаходження. Зрозумілі, клікабельні опції, такі як «Переглянути деталі», повинні бути включені, щоб користувачі могли детальніше переглядати конкретні оголошення.

Функція пошуку має бути добре помітною та простою. Рядок пошуку має бути розташований на видному місці та дозволяти користувачам фільтрувати нерухомість на основі місцезнаходження, цінового діапазону, кількості гостей, зручностей та інших уподобань. Розширена система фільтрації дозволить користувачам звузити результати на основі певних характеристик, таких як чи дозволено проживання з домашніми тваринами, чи пропонується паркування. Крім того, у процес пошуку має бути інтегровано карту, що дозволить користувачам візуалізувати розташування нерухомості відносно бажаного району, пропонуючи більш інтерактивний спосіб перегляду доступних оголошень.

Профіль користувача та панель інструментів є важливими як для орендодавців, так і для орендарів. Орендодавцям потрібна проста, інтуїтивно зрозуміла панель інструментів для керування оголошеннями про нерухомість, відстеження бронювань та внесення необхідних змін. Орендарі, з іншого боку, повинні мати легкий доступ до історії бронювань, майбутніх бронювань та налаштувань профілю. Обидва профілі повинні дозволяти користувачам легко керувати своєю особистою інформацією та уподобаннями.

Процес бронювання має бути ефективним, з чітким покроковим процесом, який веде користувачів від вибору нерухомості до підтвердження бронювання. Після бронювання нерухомості користувачі повинні отримати чітке підтвердження, яке містить усі необхідні деталі, такі як дати заїзду/виїзду, загальна вартість та номер бронювання. Слід інтегрувати зручний календар, щоб допомогти орендарям та орендодавцям легко вибирати дати та переглядати наявність вільних місць [7].

Зосереджуючись на цих вимогах до інтерфейсу користувача, платформа буде одночасно функціональною та зручною для користувача, пропонуючи інтуїтивно зрозуміле, ефективне та доступне середовище для пошуку, оренди та управління нерухомістю. Це не лише покращить загальний досвід користувача, але й сприятиме успіху платформи на конкурентному ринку оренди житла.

2 МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Загальні відомості

UML (Уніфікована мова моделювання) – це потужний інструмент, що використовується в розробці програмного забезпечення для візуалізації, специфікації, побудови та документування різних аспектів системи. Він забезпечує стандартизований підхід, що особливо корисно в об'єктно-орієнтованій розробці програмного забезпечення. Використовуючи UML, розробники та зацікавлені сторони можуть створювати чіткі, організовані моделі, що відображають структуру та поведінку системи [8].

Діаграми UML зазвичай поділяються на різні категорії, кожна з яких виконує певну функцію в проєктованій системи. Структурні діаграми зосереджені на статичних аспектах системи, таких як її архітектура та компоненти, тоді як поведінкові діаграми підкреслюють, як система поводиться з часом та як взаємодіють її компоненти.

Структурні діаграми включають діаграми класів, які представляють різні класи в системі разом з їхніми атрибутами та методами, а також показують зв'язки між ними. Діаграми об'єктів надають знімок екземплярів цих класів у певний момент часу. Інші структурні діаграми, такі як діаграми компонентів, ілюструють модулі системи та те, як вони залежать один від одного, тоді як діаграми розгортання зосереджені на фізичних зв'язках між апаратним та програмним забезпеченням. Діаграми пакетів також є частиною структурної категорії, групуючи пов'язані класи та компоненти для відображення їхніх залежностей.

З іншого боку, поведінкові діаграми зосереджені на тому, як система функціонує динамічно. Діаграми варіантів використання, наприклад, зображують функціональні вимоги системи, показуючи взаємодію між користувачами (акторами) та системою. Діаграми послідовностей використовуються для демонстрації взаємодії об'єктів з часом шляхом

детального опису послідовності повідомлень, якими вони обмінюються. Діаграми співпраці, хоча й подібні до діаграм послідовностей, більше уваги приділяють зв'язкам між об'єктами, ніж їхній взаємодії в часовій послідовності. Діаграми машин станів використовуються для моделювання різних станів, у яких може перебувати об'єкт або система, та того, як вона переходить на основі подій. Діаграми активності пропонують візуальне представлення робочих процесів або бізнес-процесів, ілюструючи, як дії перетікають одна в іншу в системі [8].

Діаграми взаємодії, такі як діаграми зв'язку та часові діаграми, підкреслюють, як компоненти взаємодіють один з одним у системі. Діаграми зв'язку зосереджені на структурі та зв'язках між об'єктами та на тому, як вони співпрацюють через обмін повідомленнями. З іншого боку, часові діаграми корисні для моделювання послідовності станів та подій з часом, що робить їх ідеальними для систем, де важливі часові обмеження.

UML – це універсальний інструмент, що використовується на різних етапах розробки програмного забезпечення. Під час фаз аналізу та проєктування UML дозволяє командам візуалізувати архітектуру та поведінку системи, що допомагає зрозуміти вимоги та спланувати рішення. UML також служить інструментом документування, що полегшує розробникам, тестувальникам та зацікавленим сторонам розуміння структури та функціональності системи. Крім того, він діє як засіб комунікації, гарантуючи, що всі, хто бере участь у проєкті, мають чітке та послідовне розуміння дизайну системи [8].

У міру розвитку програмного забезпечення UML залишається корисним для відстеження змін та підтримки актуального представлення системи. Це особливо цінно під час модифікацій та поточного обслуговування, оскільки UML гарантує, що оновлення системи належним чином відображаються в проєктній документації.

Переваги використання UML численні. Його стандартизований підхід гарантує, що команди можуть ефективно працювати разом, незалежно від

їхнього індивідуального досвіду чи використовуваних інструментів. Візуальний характер діаграм UML допомагає спростити складні системи, роблячи їх легшими для розуміння. Крім того, він надає вичерпну документацію, до якої можна звертатися протягом усього життєвого циклу системи. Нарешті, UML є адаптивним, що робить його придатним для широкого спектру систем, від невеликих додатків до великих рішень корпоративного рівня.

На завершення, UML відіграє вирішальну роль у розробці об'єктно-орієнтованих систем. Він пропонує структурований та стандартизований метод візуалізації, документування та комунікації проєкту системи. Забезпечуючи ясність та узгодженість, UML допомагає забезпечити добре планування, розуміння та успішне виконання програмних проєктів.

2.2 Об’єктне та функціональне моделювання

2.2.1 Діаграма прецедентів. Діаграма варіантів використання — це поведінкова діаграма в UML (Уніфікована мова моделювання), яка візуально представляє взаємодію між користувачами системи (акторами) та самою системою. Вона зосереджена на зображенні функціональних вимог до системи, показуючи ключові завдання, які система виконує у відповідь на дії користувача. Мета цієї діаграми — запропонувати загальне уявлення про функціональність системи, підкреслюючи, як користувачі (або інші системи) взаємодіють із системою для досягнення певних цілей [9].

На цих діаграмах актори представляють різні сутності, які взаємодіють із системою. Це можуть бути окремі особи, такі як клієнти чи адміністратори, або навіть зовнішні системи, які взаємодіють із програмою. З іншого боку, варіанти використання символізують конкретні дії або послуги, які система пропонує у відповідь на потреби акторів. Кожен

варіант використання пов'язаний з певною функцією або завданням, таким як реєстрація користувача, бронювання або обробка транзакції.

Компоненти діаграми варіантів використання включають:

- актори представлені фігурками-паличками та розташовані поза межами системи;
- варіанти використання зображуються у вигляді овалів та описують завдання, які виконує система;
- межі системи визначають межі системи та охоплюють варіанти використання, вказуючи, що входить до сфери функціональності системи;
- асоціації – це лінії, що з'єднують учасників з варіантами використання, вказуючи на зв'язок або взаємодію між ними.

Замість того, щоб зосереджуватися на детальному потоці дій, діаграми варіантів використання слугують широким оглядом функцій системи. Вони виділяють основні взаємодії та ключові послуги, що пропонуються системою своїм користувачам, що полегшує зацікавленим сторонам розуміння того, як система відповідає потребам своїх користувачів. Ці діаграми допомагають визначити необхідні функції та гарантують, що команда розробників узгодить систему з очікуваннями користувачів.

Розроблена діаграма прецедентів використання представлена на рис. 2.1.

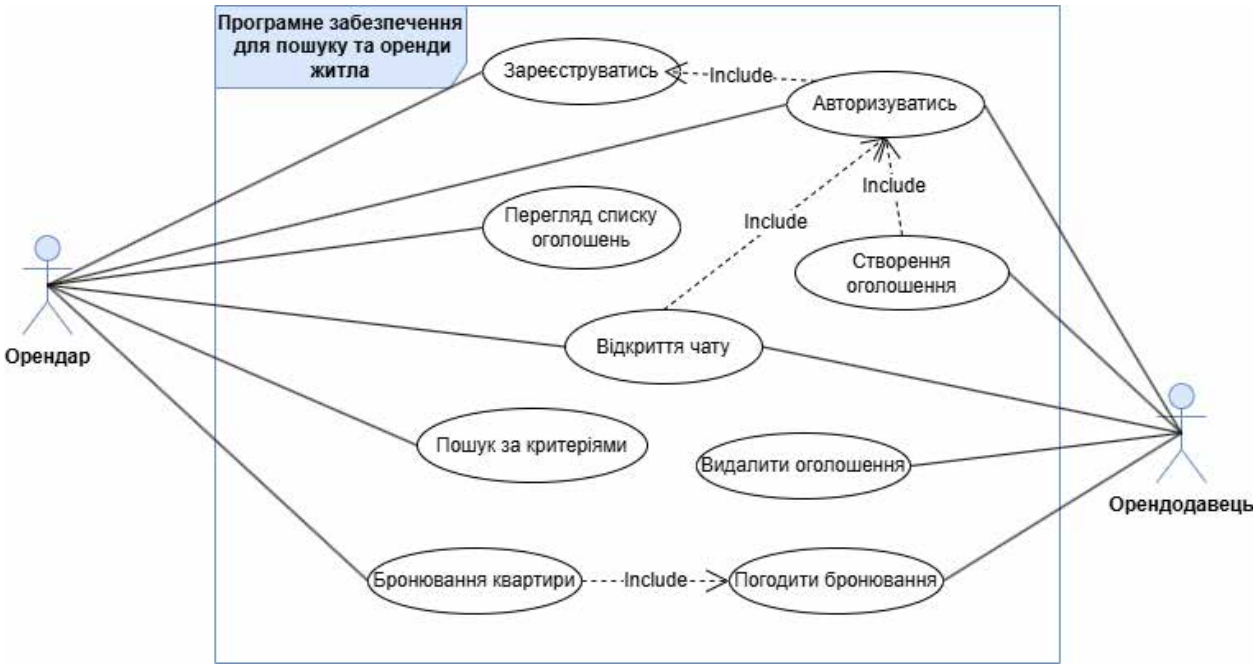

Рис. 2.1 Діаграма прецедентів

Створена діаграма прецедентів містить акторів:

- “Орендар”;
- “Орендодавець”.

Актор «Орендар» включає такі прецеденти:

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

Актор «Орендодавець» включає такі прецеденти:

- створення оголошення;

- видалити оголошення;
- погодити бронювання.

Прецеденти певним чином залежать одне від одного.

Розглянемо детальніше вищеописані прецеденти.

Варіант використання: Перегляд списку оголошень

Актор: Орендар

Опис: Приклад використання «Перегляд списку оголошень» описує процес, за допомогою якого орендар отримує доступ та переглядає список доступних оголошень про оренду житла. Орендар може використовувати різні фільтри, щоб звузити результати пошуку на основі певних уподобань, таких як ціна, розташування, тип нерухомості та інші зручності. Цей процес дозволяє орендарям переглядати та досліджувати варіанти оренди, щоб знайти відповідну нерухомість.

Основний потік:

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

Альтернативні потоки:

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

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

Розглянемо ще один сценарій використання.

Варіант використання: Підтвердження бронювання

Актор: Орендодавець

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

Основний потік:

1. орендодавець отримує сповіщення про новий запит на бронювання від орендаря;
2. орендодавець входить у систему та переходить до розділу «Запити на бронювання»;
3. система відображає список усіх запитів на бронювання, що очікують розгляду, включаючи такі деталі, як ім'я орендаря, дати та запитувана нерухомість;
4. орендодавець вибирає запит на бронювання для перегляду;
5. система показує детальну інформацію про бронювання, включаючи контактну інформацію орендаря, запитувані дати, ціну за день та будь-які додаткові умови чи примітки, надані орендарем;
6. орендодавець переглядає деталі та вирішує, чи підтверджувати чи відхиляти бронювання;
7. якщо орендодавець підтверджує бронювання;
8. система оновлює статус бронювання на «Підтверджено»;
9. орендар отримує підтвердження успішного бронювання;
10. система надсилає підтвердження як орендодавцю, так і орендарю, включаючи деталі бронювання, такі як дати, загальна вартість та контактну інформацію;
11. якщо орендодавець відхиляє бронювання;
12. система оновлює статус бронювання на «Відхилено»;
13. орендар отримує повідомлення про відхилення з причиною (якщо вона вказана) та йому рекомендується ознайомитися з іншими доступними оголошеннями.

Альтернативні потоки:

1. якщо запит на бронювання вже був підтверджений або відхилений іншим орендодавцем або після певного періоду, система повідомить орендодавця та вимкне можливість підтвердження або відхилення запиту;

2. якщо орендодавець помітить будь-яку відсутню або неповну інформацію в запиті на бронювання (наприклад, відсутні контактні дані), він може попросити орендаря надати відсутню інформацію, перш ніж продовжити підтвердження;
3. якщо запитувані дати перетинаються з існуючим бронюванням, система попередить орендодавця, і орендодавець може або відхилити бронювання, або запропонувати орендарю альтернативні дати.

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

2.2.2 Діаграма послідовності. Діаграма послідовності – це тип діаграми UML (уніфікованої мови моделювання), яка використовується для візуалізації послідовності взаємодій між різними об'єктами в системі з плином часу. Вона зосереджена на порядку обміну повідомленнями та допомагає проілюструвати, як процеси працюють та взаємодіють [10].

У контексті програмного забезпечення для пошуку та оренди житла діаграма послідовності показує динамічну поведінку системи, наприклад, як орендар надсилає запит на бронювання, як система його обробляє та як реагує орендодавець.

Кожен учасник (наприклад, Орендар, Веб-інтерфейс, Служба бронювання, Орендодавець) представлений вертикальною лінією сполучення, а повідомлення, якими вони обмінюються, показані у вигляді горизонтальних стрілок. Діаграма починається з початкової дії користувача та відстежує кожну взаємодію крок за кроком, що робить корисним як для

розробників, так і для зацікавлених сторін розуміння того, як система поводиться в реальних сценаріях.

Підсумовуючи, діаграма послідовності забезпечує чітке та структуроване уявлення про те, як компоненти в системі співпрацюють для виконання певного завдання, забезпечуючи прозорість та логічну обґрунтованість функціонального потоку.

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

- “Орендар”;
- “Оголошення”;
- “Бронювання”;
- “Чат”.

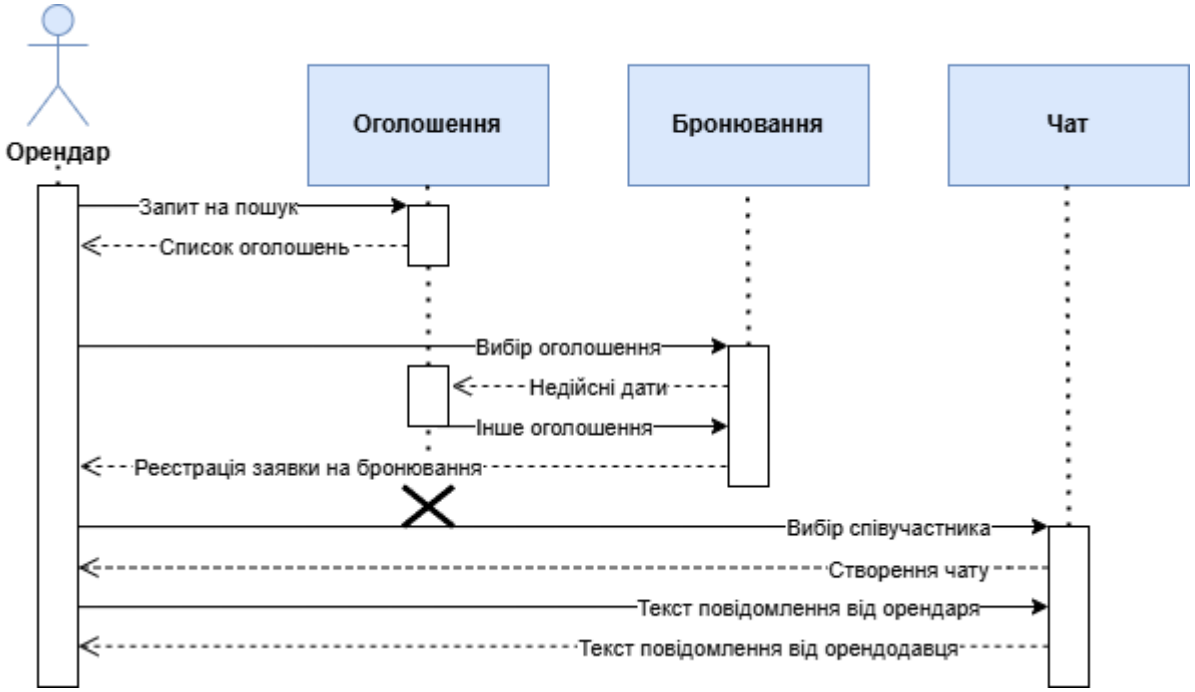

Рис. 2.2 Діаграма послідовності

Орендар розпочинає процес пошуку, звертаючись до системи оголошень, яка надає перелік доступних варіантів. Ознайомившись із пропозиціями, він вибирає відповідне оголошення, після чого система

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

2.2.3 Діаграма активності. Діаграма діяльності – це візуальне представлення, яке використовується в UML (Уніфікована мова моделювання) для демонстрації динамічних аспектів системи, зосереджуючись на послідовності та потоці дій у межах певного процесу. Вона особливо ефективна для зображення логіки бізнес-процедур та операційних робочих процесів у чіткій та структурованій формі [11].

У веб-застосунку, розробленому для пошуку та оренди житла, діаграми діяльності можуть допомогти проілюструвати ключові процеси, такі як створення оголошень про нерухомість, пошук житла, обробка бронювань та керування функціями, пов'язаними з користувачами, такими як реєстрація або вхід. Кожна дія в системі зазвичай зображується як заокруглений прямокутник, а перехід від одного кроку до наступного відображається за допомогою стрілок. Точки прийняття рішень, такі як умовні дії на основі статусу користувача, позначені ромбами, а одночасні операції, такі як завантаження фотографій під час введення деталей оголошення, позначені за допомогою смуг синхронізації.

Ці діаграми цінні, оскільки вони пропонують цілісне уявлення про те, як користувачі взаємодіють із системою, дозволяючи як розробникам, так і зацікавленим сторонам зрозуміти потік операцій ще до написання будь-якого коду. Ця ясність допомагає виявити потенційні проблеми, такі

як непотрібна складність або логічні прогалини, на ранній стадії проєктування.

Візьмемо як приклад сценарій бронювання. Процес може розпочатися зі входу орендаря на платформу, після чого він шукає підходящу нерухомість, переглядає деталі обраного оголошення та надсилає запит на бронювання. Потім система повідомляє орендодавця, який може або схвалити, або відхилити запит, завершуючи процес.

Розроблена діаграма активності представлена на рис. 2.3

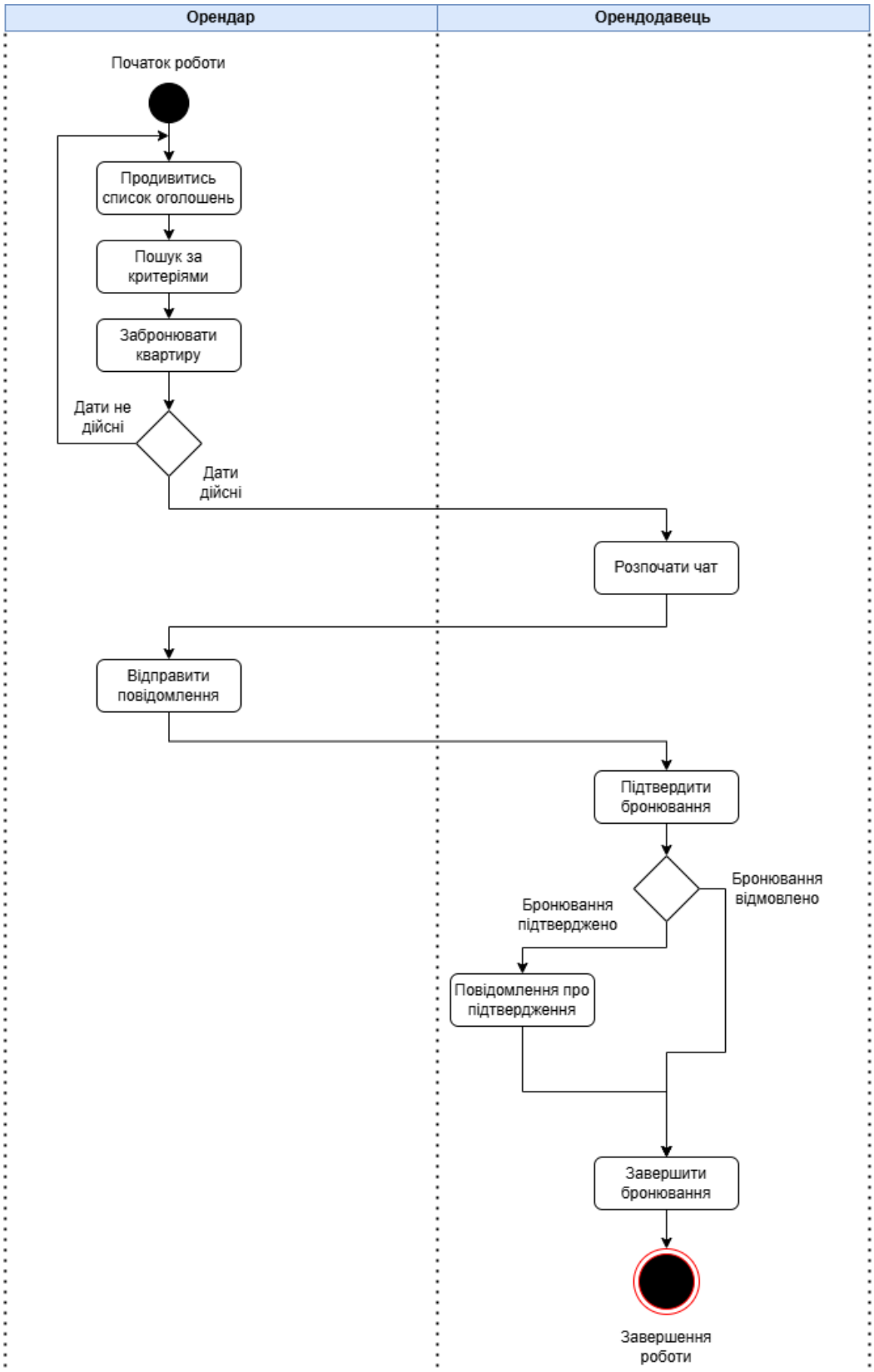

Рис. 2.3 Діаграма активності

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

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

2.3 Абстракції предметної області

Під час розробки програмних систем абстракції предметної області слугують основою для перетворення складних реальних областей на структуровані та зрозумілі моделі. Ці абстракції допомагають визначити ключові компоненти, їхні характеристики та способи їхньої взаємодії, дозволяючи розробникам зосередитися на тому, що є важливим для вирішення поточної проблеми. У контексті платформи оренди житла це включає ізоляцію основних понять, таких як користувачі, орендована нерухомість, оголошення, бронювання та місцезнаходження. Кожен з цих елементів представлений як логічна сутність у системі. Користувачі зазвичай поділяються на дві основні категорії: орендодавці, які пропонують нерухомість в оренду, та орендарі, які шукають підходяще житло. Нерухомість визначається такими атрибутами, як адреса, ціна оренди, доступні зручності та зображення. Оголошення діють як цифрові представлення доступної оренди, надаючи орендарям необхідну інформацію для прийняття обґрунтованих рішень. Бронювання фіксують транзакційний аспект платформи, відстежуючи дати бронювання, ідентифікаційні дані користувачів та статуси бронювання. Завдяки цьому процесу абстракції програмне забезпечення стає спрощеною моделлю реального середовища, що полегшує його створення, обслуговування та розширення. Воно забезпечує чіткість у проєктуванні системи та полегшує комунікацію між розробниками, клієнтами та зацікавленими сторонами. Зрештою, абстракція дозволяє застосунку відображати реальні робочі процеси, зберігаючи при цьому чисту та ефективну внутрішню структуру.

Рис. 2.4 Абстракції предметної області

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

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

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

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

2.4 Діаграма класів

Діаграма класів – одна з найфундаментальніших діаграм в уніфікованій мові моделювання (UML), яка використовується для представлення статичної структури програмної системи. Вона ілюструє класи системи, їх атрибути та

методи, а також зв'язки між ними. Діаграми класів є вирішальними на етапі проєктування розробки, оскільки вони допомагають візуалізувати та організувати компоненти програми до початку фактичного кодування [12].

У контексті веб-платформи для пошуку та оренди житла, діаграма класів відображає основні елементи домену, такі як користувачі, нерухомість, оголошення та бронювання. Наприклад, клас Користувач може бути узагальнений на два типи: Орендодавець та Орендар, кожен з яких має певні ролі та дозволи. Клас Нерухомості може включати такі атрибути, як адреса, ціна за день, опис та колекцію фотографій. Клас Оголошення пов'язує нерухомість з її доступністю та умовами оренди. Бронювання представлятиме бронювання та може включати дати бронювання, статус (наприклад, очікує, підтверджено) та посилання як на орендаря, так і на оголошення.

Зв'язки між класами також є важливими в діаграмах класів. Асоціації показують, як об'єкти пов'язані (наприклад, орендодавець може мати багато об'єктів нерухомості, нерухомість може мати багато бронювань). Успадкування використовується, коли один клас має спільні риси з іншим (наприклад, Орендар та Орендодавець успадковують від Користувача). Агрегація та композиція представляють зв'язки "ціле-частина", наприклад, нерухомість складається з кількох зображень або Бронювання містить платіжні дані.

Загалом, діаграма класів пропонує високорівневий, але детальний план архітектури системи. Вона підтримує краще планування, допомагає виявляти можливості повторного використання, забезпечує узгодженість та закладає основу для реалізації схеми бази даних та логіки програми.

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

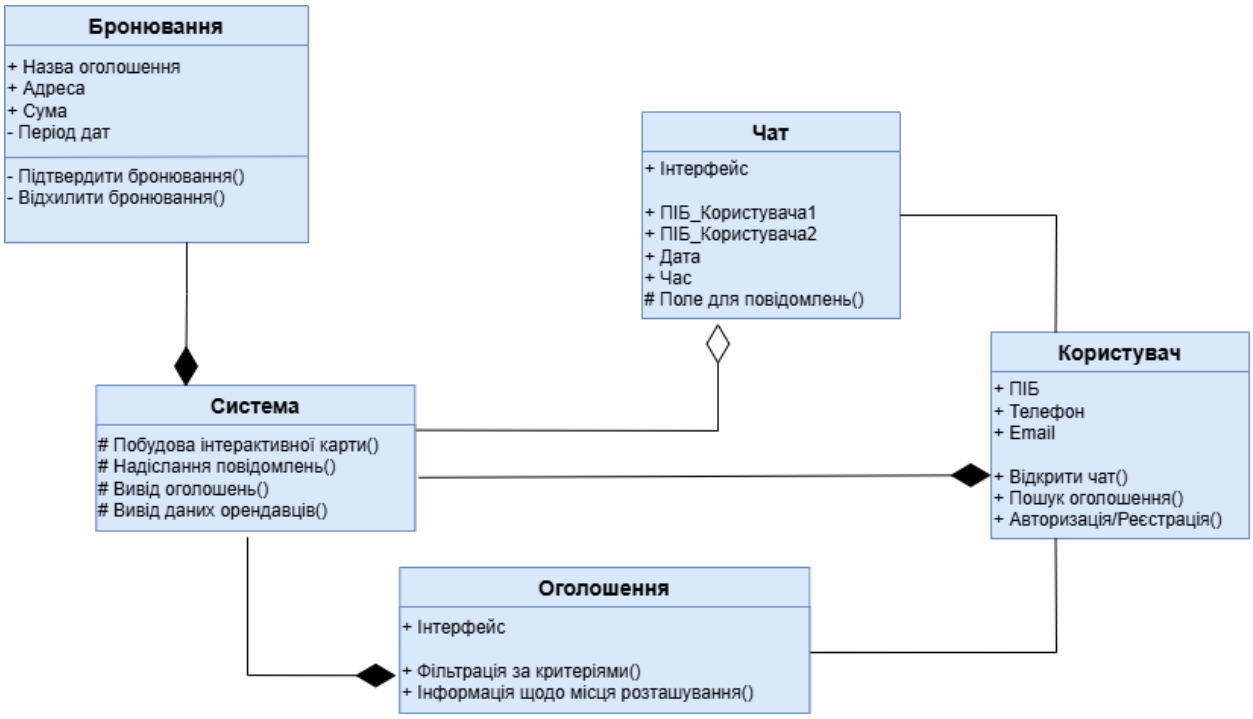

Рис. 2.5 Діаграма класів

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

3 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Логічна модель даних

Логічна модель даних — це структуроване представлення елементів даних у програмній системі, що зосереджується на організації даних та зв'язках між різними сутностями, незалежно від того, як дані будуть фізично зберігатися або реалізовуватися в базі даних. Вона служить мостом між концептуальним розумінням предметної області та технічним проєктом, який буде наступним [12].

У контексті програмної системи для пошуку та оренди житла, логічна модель даних окреслює всі основні сутності, такі як користувачі, нерухомість, оголошення, бронювання, відгуки та повідомлення, а також їхні атрибути та взаємозв'язки. Кожна сутність представляє реальну концепцію. Наприклад, сутність «Користувач» включає такі атрибути, як ім'я, електронна пошта, номер телефону та роль (орендар або орендодавець). Сутність «Нерухомість» містить такі деталі, як місцезнаходження, опис, ціна за день та статус доступності. «Оголошення» пов'язує нерухомість з її умовами оренди та статусом видимості на платформі. «Бронювання» фіксує такі дані, як дати заїзду/виїзду, ідентифікатори орендаря та орендодавця, а також статус бронювання. Додаткові сутності, такі як «Відгук» або «Повідомлення», можуть бути додані для підтримки зворотного зв'язку та комунікації з користувачами.

Модель також визначає зв'язки між сутностями. Наприклад, один користувач (орендодавець) може володіти багатьма об'єктами нерухомості, тоді як кожен об'єкт може мати кілька оголошень або бронювань. Орендар може мати кілька бронювань, і кожне бронювання пов'язане з одним об'єктом нерухомості. Ці зв'язки ілюструються за допомогою обмежень первинного та зовнішнього ключа, але на логічному рівні акцент робиться на чіткості зв'язків

між даними та бізнес-правил, а не на технічних деталях, таких як табличні індекси або формати зберігання.

Мета логічної моделі даних полягає в забезпеченні повного та правильного представлення потреб у даних перед переходом до впровадження бази даних. Вона дозволяє розробникам, зацікавленим сторонам та аналітикам перевіряти вимоги до даних, виявляти невідповідності та вдосконалювати бізнес-логіку на ранніх етапах процесу проєктування.

Логічна модель системи представлена на рисунку 3.1.

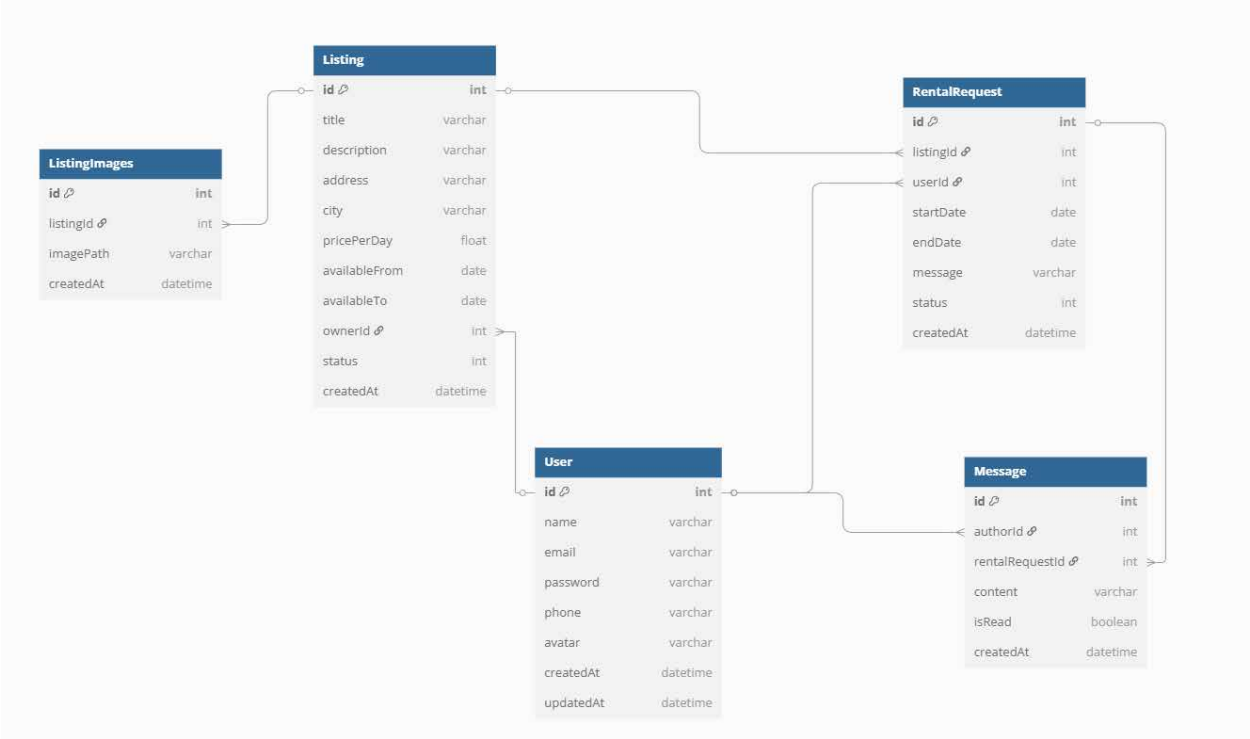

Рис. 3.1 ER-діаграма “homyway_db”

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

Таблиця **User:**

1. id: Унікальний ідентифікатор користувача (первинний ключ);
2. name: Ім'я користувача;
3. email: Електронна пошта;
4. password: Пароль користувача;
5. phone: Телефон користувача;

6. avatar: Шлях до аватарки користувача;
7. createdAt: Дата та час створення запису;
8. updatedAt: Дата та час останнього оновлення запису.

Ця таблиця зберігає дані користувачів системи

Таблиця **Listing:**

1. id: Унікальний ідентифікатор оголошення (первинний ключ);
2. title: Заголовок оголошення;
3. description: Опис оголошення (необов'язкове);
4. address: Адреса об'єкта, що здається в оренду;
5. city: Місто, де знаходиться об'єкт;
6. pricePerDay: Ціна оренди за день;
7. availableFrom: Дата, з якої об'єкт доступний для оренди (необов'язкове);
8. availableTo: Дата, до якої об'єкт доступний для оренди (необов'язкове);
9. ownerId: Ідентифікатор користувача (власника), який створив оголошення;
10. status: Статус оголошення (наприклад, активне чи неактивне);
11. createdAt: Дата та час створення оголошення.

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

Таблиця **ListingImages:**

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

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

Таблиця **RentalRequest:**

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

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

Таблиця **Message:**

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

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

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

Кожен користувач може створювати кілька оголошень, що пов'язано через поле ownerId у таблиці Listing, яке є зовнішнім ключем, що посилається

на ідентифікатор користувача з таблиці User. Таким чином, один користувач може бути власником кількох об'єктів, що здаються в оренду.

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

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

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

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

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

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

3.2 Вибір системи управління базою даних та її реалізація

Для бакалаврської роботи на тему «Програмне забезпечення для пошуку та оренди житла» вибір відповідної системи керування базами даних (СКБД) є критичним рішенням, яке впливає на ефективність та масштабованість програмного забезпечення. Програмне забезпечення призначене для спрощення процесу пошуку, розміщення оголошень та оренди житла, що включає обробку різних типів даних, таких як користувачі, нерухомість, запити на оренду та зв'язок між користувачами.

Розглядаючи ідеальну СКБД для цієї житлової платформи, слід враховувати кілька факторів. По-перше, система потребує надійної підтримки структурованих даних, включаючи дані користувачів, оголошення про нерухомість, запити на оренду та повідомлення. Реляційна база даних є природним вибором для цього типу даних завдяки своїй здатності ефективно керувати зв'язками між різними сутностями. Реляційні бази даних, такі як MySQL та PostgreSQL, є дуже ефективними для таких застосувань, оскільки вони дозволяють створювати складні моделі даних та надають розширені можливості запитів.

Масштабованість та продуктивність також є важливими питаннями, особливо з огляду на те, що кількість оголошень та користувачів з часом зростає. MySQL — це добре відома реляційна база даних, яка добре обробляє масштабні програми з великим обсягом читання, що робить її придатною для застосування, орієнтованого на пошук нерухомості. Здатність MySQL масштабуватися, зберігаючи при цьому продуктивність, особливо з індексацією та оптимізацією запитів, робить його ідеальним вибором для такого програмного забезпечення [14].

Ще одним ключовим міркуванням є необхідність узгодженості та цілісності даних. Транзакції оренди та взаємодія з користувачами вимагають системи, яка забезпечує точність та узгодженість даних. MySQL, як і інші реляційні бази даних, підтримує властивості ACID (атомарність, узгодженість,

ізоляція, довговічність), що гарантує надійну обробку всіх транзакцій бази даних, навіть у разі збоїв системи. Це особливо важливо для управління запитами на оренду та обліковими записами користувачів, де неправильні або відсутні дані можуть призвести до серйозних проблем.

Крім того, система покладатиметься на складні запити для фільтрації властивостей на основі кількох критеріїв пошуку, і їй потрібно буде керувати зв'язками між різними сутностями, такими як користувачі, оголошення та запити на оренду. Операції JOIN та розширені можливості фільтрації, що надаються MySQL, роблять його добре пристосованим для таких завдань.

Безпека є ще одним критичним фактором, враховуючи конфіденційний характер транзакцій оренди. MySQL включає надійні функції безпеки, такі як автентифікація користувачів, шифрування даних та механізми контролю доступу, що гарантує захист даних від несанкціонованого доступу.

Враховуючи всі ці фактори, MySQL стає чудовим вибором для базової СУБД програмного забезпечення для оренди житла. Він поєднує масштабованість, продуктивність, безпеку та простоту використання, що є важливим для системи такого типу.

Впровадження програмного забезпечення розпочнеться з проєктування схеми бази даних. Вона включатиме таблиці для зберігання користувачів, оголошень про нерухомість, запитів на оренду, повідомлень та, можливо, відгуків. Таблиця «Користувачі» зберігатиме важливі дані як для орендарів, так і для власників нерухомості, такі як контактна інформація та ролі (орендарь або власник). Таблиця «Оголошення» міститиме детальну інформацію про нерухомість, таку як її розташування, ціна та доступність. Таблиця «Запити на оренду» відстежуватиме запити, зроблені користувачами, тоді як таблиця «Повідомлення» зберігатиме повідомлення між користувачами щодо запитів на оренду.

Зв'язки між таблицями будуть визначені за допомогою зовнішніх ключів для забезпечення цілісності даних. Наприклад, кожен запит на оренду буде пов'язаний з користувачем, а кожен запис матиме власника. Аналогічно,

повідомлення будуть пов'язані як із запитами на оренду, так і з користувачами, що гарантує, що система підтримує логічні зв'язки між сутностями.

Після налаштування схеми бази даних, серверна частина програми буде розроблена з використанням серверних технологій, таких як PHP, Node.js або Python. Вони взаємодіятимуть з базою даних MySQL для виконання таких завдань, як реєстрація користувачів, пошук нерухомості, подання запитів на оренду та обмін повідомленнями. Фронтенд програми надаватиме інтуїтивно зрозумілий інтерфейс, де користувачі зможуть переглядати оголошення, подавати запити на оренду та спілкуватися з власниками нерухомості. Це можна розробити з використанням сучасних фронтенд-технологій, таких як React, Vue.js або простого HTML/CSS/JavaScript [15].

Для забезпечення безпечної автентифікації користувачів система реалізує хешування паролів та контроль доступу на основі ролей, щоб розрізняти орендарів та власників нерухомості. Нарешті, ретельне тестування буде необхідним, щоб переконатися, що всі функції працюють правильно, і що система може обробляти зростаючу кількість користувачів та оголошень без зниження продуктивності.

На завершення, MySQL є оптимальним вибором для цього програмного забезпечення для пошуку та оренди житла. Його реляційна структура, масштабованість, підтримка складних запитів та безпеки роблять його ідеальною основою для створення надійної, ефективної та зручної платформи.

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

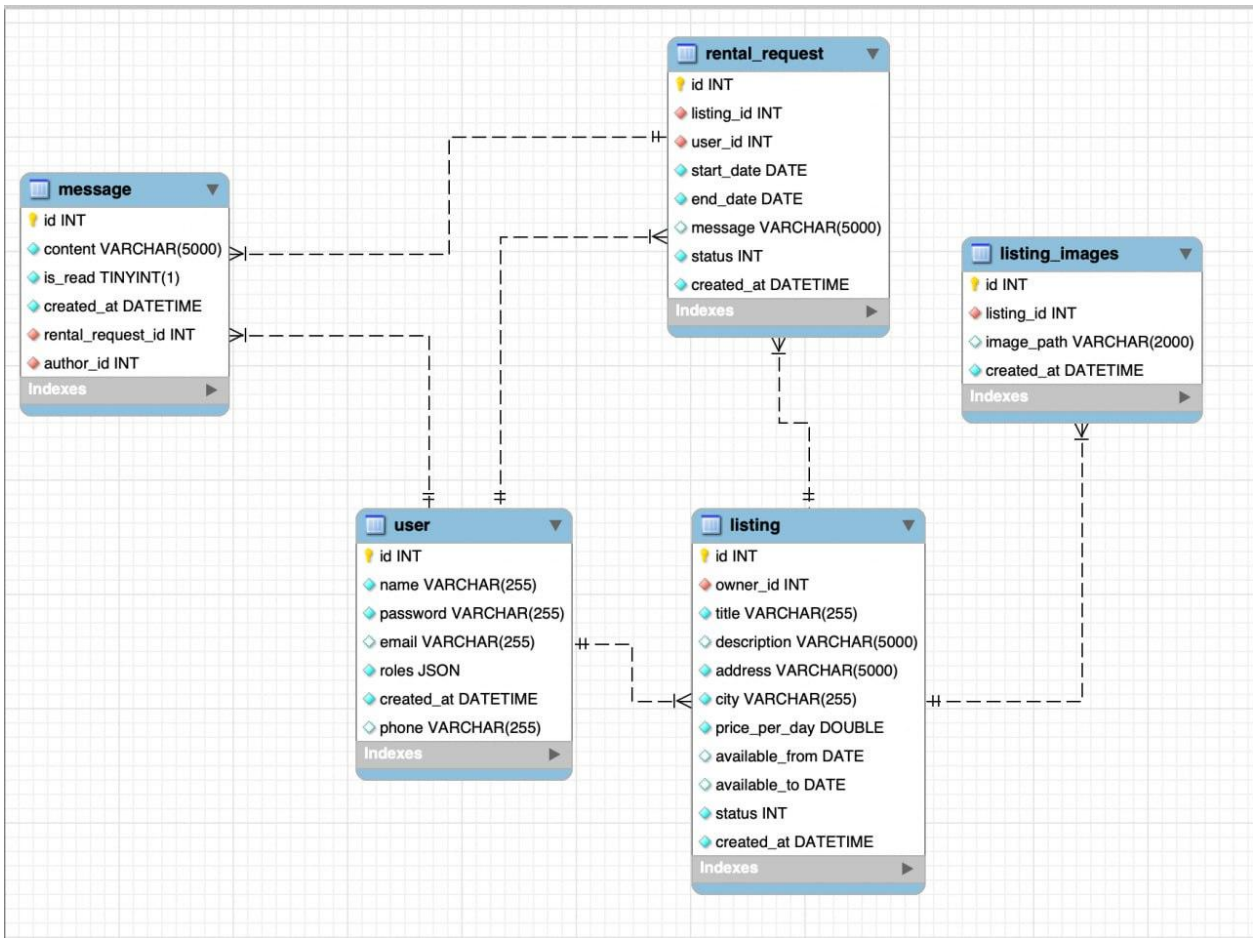

Рис. 3.2 База даних

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

Додатково, база має таблицю платежів, яка відстежує фінансові транзакції, зокрема суми, способи оплати та підтвердження надходження коштів. І таблиця чату, що містить історію повідомлень між орендарем і орендодавцем, забезпечуючи ефективну комунікацію між сторонами.

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

3.3 Архітектура програмного забезпечення

Архітектура програмного забезпечення платформи пошуку та оренди житла відіграє вирішальну роль у визначенні продуктивності, масштабованості та взаємодії користувача системи. Вона структурована на кількох рівнях, кожен з яких призначений для виконання певних завдань, забезпечуючи безперебійну та ефективну роботу платформи. В основі цієї архітектури лежать фронтенд, бекенд та база даних, які працюють разом, щоб забезпечити безперебійний взаємодію з користувачем.

Фронтенд-рівень відповідає за те, з чим безпосередньо взаємодіють користувачі. Він зосереджений на інтерфейсі користувача (UI) та загальному досвіді (UX). Для створення захопливого та адаптивного інтерфейсу користувача зазвичай використовуються сучасні фронтенд-фреймворки, такі як React, Vue.js або Angular. Ці фреймворки сприяють ефективному рендерингу та управлінню станом різних компонентів, таких як форми пошуку нерухомості, картки оголошень та форми запиту на оренду. Маршрутизація на стороні клієнта гарантує, що користувачі можуть переміщатися по програмі без необхідності перезавантаження сторінки, тоді як бібліотеки управління станом, такі як Redux або Vuex, допомагають підтримувати узгодженість даних у всій програмі [16].

З іншого боку, бекенд-рівень керує бізнес-логікою, обробкою запитів, автентифікацією користувачів та зв'язком з базою даних. Побудований на надійних серверних фреймворках, таких як PHP з Symfony, Node.js з Express або Python з Django, бекенд слугує основою платформи. Він надає API, зазвичай RESTful або GraphQL, щоб фронтенд міг взаємодіяти з системою. Ці API обробляють важливі операції, такі як реєстрація користувачів, керування оголошеннями, подання запитів на оренду та надсилання повідомлень. Крім того, бекенд забезпечує дотримання бізнес-правил, наприклад, забезпечення обробки лише дійсних запитів на оренду або коректного оновлення оголошень залежно від наявності.

Безпека всередині бекенду також є пріоритетом, і зазвичай впроваджуються протоколи автентифікації, такі як JWT (JSON Web Tokens) або OAuth2. Ці методи гарантують безпечну автентифікацію та авторизацію користувачів, надаючи їм доступ до відповідних функцій на основі їхніх ролей — незалежно від того, чи є вони орендарями, чи власниками нерухомості. Бекенд також включає механізми контролю доступу на основі ролей, гарантуючи, що користувачі мають дозвіл на доступ або зміну лише тих даних, на які вони мають дозвіл.

Рівень бази даних – це місце, де зберігаються всі критично важливі дані, включаючи профілі користувачів, списки оголошень про нерухомість, запити на оренду та повідомлення. Зазвичай використовується реляційна система керування базами даних (RDBMS), така як MySQL або PostgreSQL. Ці системи зберігають структуровані дані в таблицях зі зв'язками, визначеними зовнішніми ключами. Наприклад, кожен користувач може створювати кілька оголошень, і кожне оголошення може мати кілька запитів на оренду. Ці зв'язки гарантують, що дані залишаються узгодженими та доступними за потреби. Використання зовнішніх ключів допомагає підтримувати цілісність посилань, щоб дані в пов'язаних таблицях, таких як Listings та RentalRequests, залишалися коректно пов'язаними.

Проєктування схеми бази даних також має вирішальне значення для успіху програми. Наприклад, таблиці можуть містити Користувачів, Оголошення, RentalRequests, Повідомлення та Зображення Оголошень. Кожна таблиця налаштована для зберігання певних даних – особистих даних користувачів, описів оголошень та їхньої наявності, повідомлень, пов'язаних із запитами на оренду, та зображень нерухомості. Структура цих таблиць гарантує, що всі взаємодії ефективно записуються та доступні. Цілісність цих даних підтримується шляхом застосування властивостей ACID (атомарність, узгодженість, ізоляція, довговічність) у базі даних, що забезпечує надійність та запобігає таким проблемам, як пошкодження даних [16].

Безпека вбудована в кожен рівень архітектури. Від шифрування конфіденційної інформації, такої як паролі та дані користувачів, до забезпечення безпечного зв'язку через HTTPS, протоколи безпеки суворо дотримуються. Використання методів шифрування, таких як bcrypt або Argon2, гарантує безпечне зберігання паролів користувачів. Система розроблена для захисту від несанкціонованого доступу, як за допомогою керування доступом на основі ролей, так і шляхом забезпечення належної автентифікації всіх викликів API.

Коли справа доходить до обробки збільшеного трафіку, розгортання та масштабованість платформи є важливими міркуваннями. Платформу можна розгортати в хмарних середовищах, таких як AWS, Google Cloud або Azure, які забезпечують інфраструктуру, необхідну для ефективного масштабування програми. Балансування навантаження реалізовано для розподілу вхідних запитів між кількома серверами, гарантуючи, що система може обробляти великі обсяги трафіку, зберігаючи при цьому швидкість реагування. Крім того, базу даних можна масштабувати за допомогою таких методів, як реплікація та шардінг, щоб забезпечити швидкий та ефективний доступ до даних, навіть зі зростанням бази користувачів.

На завершення, архітектура платформи пошуку та оренди житла – це комплексна, добре структурована система, розроблена для забезпечення продуктивності, безпеки та безперебійного користувацького досвіду.

Рішення про використання 4-рівневої архітектури для програмного забезпечення для пошуку та оренди житла було зумовлене кількома ключовими факторами, зосередженими на масштабованості, зручності обслуговування, гнучкості та простоті розробки. 4-рівнева архітектура поділяє програму на чотири окремі рівні: рівень презентації, рівень бізнес-логіки, рівень доступу до даних та рівень бази даних. Таке розділення обов'язків гарантує, що кожна частина системи є незалежною, що призводить до чистішого коду, кращої продуктивності та можливості масштабування за потреби.

Рівень презентації обробляє користувацький інтерфейс (UI) та відповідає за те, як користувачі взаємодіють із системою. Він забезпечує інтуїтивно зрозумілий, адаптивний та зручний для користувача досвід. У програмі для пошуку та оренди житла це важливо, оскільки користувачам потрібно легко переглядати оголошення, подавати запити на оренду та керувати своїми профілями. Ізолюючи інтерфейс користувача від логіки серверної частини, рівень презентації дозволяє покращити дизайн користувацького інтерфейсу та ефективнішу оптимізацію продуктивності на стороні клієнта, таку як динамічне рендеринг даних та оновлення в режимі реального часу. Для цього рівня можна використовувати такі технології, як React або Angular, що забезпечує гнучкість для покращення інтерфейсу користувача незалежно від серверної частини [16].

Рівень бізнес-логіки, також відомий як сервісний рівень, — це місце, де знаходиться основна логіка програми. Цей рівень обробляє запити користувачів, виконує перевірки та забезпечує дотримання бізнес-правил. Наприклад, він обробляє схвалення запитів на оренду, обчислює доступність оголошень та перевіряє, чи мають користувачі правильні дозволи для виконання певних дій. Відокремлюючи бізнес-логіку від інших рівнів, легше вносити зміни до функціональності програми, не впливаючи на інтерфейс користувача чи базу даних. Такий підхід також дозволяє розробникам зосередитися на впровадженні бізнес-правил модульним, повторно використовуваним та тестованим способом, покращуючи загальну якість коду та його підтримку.

Рівень доступу до даних розташований між бізнес-логікою та базою даних, забезпечуючи абстракцію для взаємодії з базою даних. Цей рівень відповідає за запити до бази даних, отримання, оновлення та зберігання даних. Він взаємодіє з базою даних за допомогою таких методів, як SQL-запити, або через фреймворк об'єктно-реляційного відображення (ORM), такий як Doctrine на PHP або Hibernate на Java. Відокремлення доступу до даних від бізнес-логіки гарантує, що базову структуру бази даних можна змінювати або

оптимізувати, не впливаючи на бізнес-правила чи інтерфейс користувача. Це також спрощує тестування, дозволяючи використовувати макетні бази даних або джерела даних для модульного тестування.

Нарешті, рівень бази даних – це місце, де зберігаються всі постійні дані. Він містить фактичні структури даних, такі як таблиці та зв'язки, які містять всю інформацію про користувачів, оголошення, запити на оренду та повідомлення. Реляційна система керування базами даних (RDBMS), така як MySQL або PostgreSQL, може використовуватися для зберігання структурованих даних з чіткими зв'язками, забезпечуючи узгодженість та цілісність. Таке розділення рівня бази даних гарантує, що зміни в сховищі даних або запитах не впливають на бізнес-логіку або інтерфейс користувача програми.

Чотирирівнева архітектура була обрана, оскільки вона сприяє принципам модульності та розділення завдань. Кожен рівень має свою певну роль, зменшуючи залежності та дозволяючи розробникам працювати над різними частинами програми, не втручаючись один в одного. Це призводить до легшого обслуговування, оскільки помилки або покращення можна ізолювати в межах певних рівнів. Крім того, цей підхід підвищує масштабованість. У міру зростання програми стає легше масштабувати окремі рівні незалежно. Наприклад, покращення продуктивності на рівні бази даних можна внести, не впливаючи на інтерфейс користувача, або нові функції можна додати до рівня бізнес-логіки, не змінюючи рівень представлення.

Крім того, ця архітектура спрощує тестування. Модульні тести та інтеграційні тести можна застосовувати до кожного рівня окремо, гарантуючи, що всі компоненти функціонують належним чином, перш ніж вони взаємодіють один з одним. Завдяки чітким межам між рівнями розробники можуть легко ізолювати проблеми та виправляти їх цілеспрямовано, що підвищує швидкість та ефективність процесу розробки.

Загалом, вибір 4-рівневої архітектури для цього програмного забезпечення для пошуку та оренди житла забезпечує чисте, модульне та

масштабоване рішення. Це забезпечує більшу гнучкість у розробці, легше обслуговування та більш керований шлях зростання програми, оскільки потрібні нові функції або оптимізації.

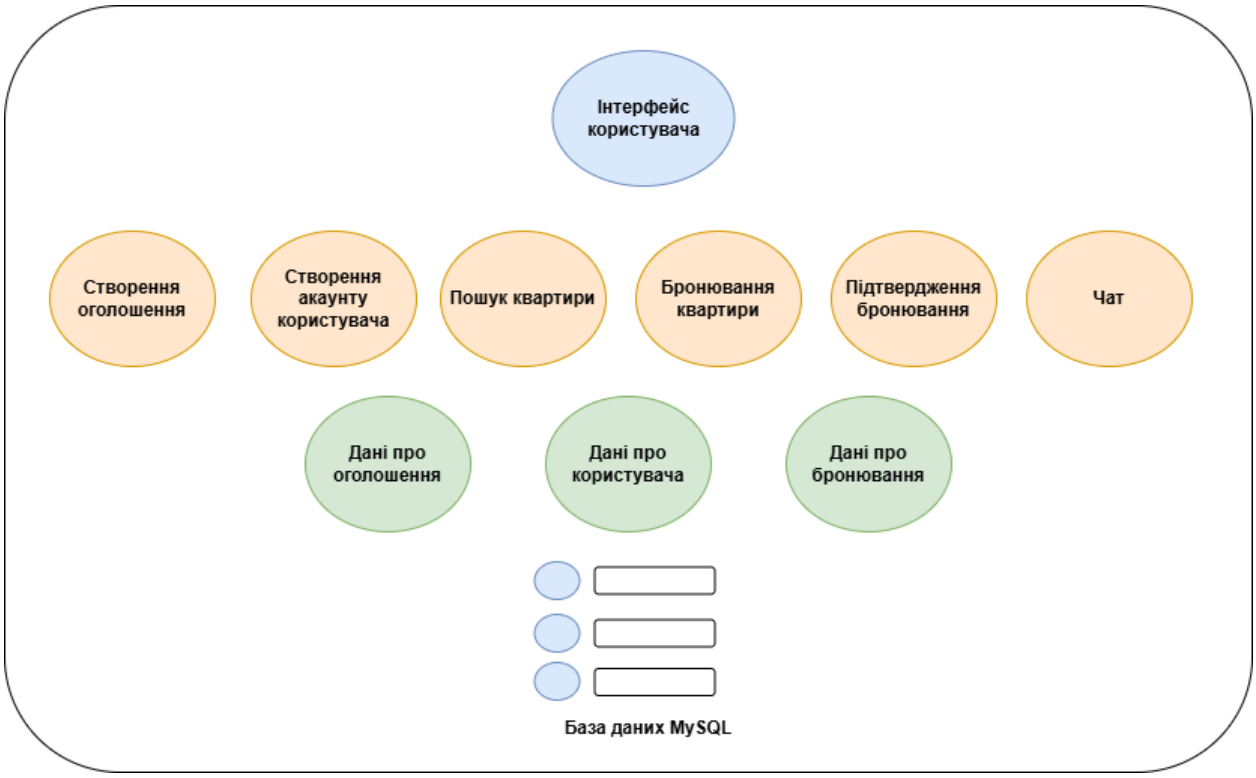

Рис. 3.3 Багатошарова архітектура

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

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

Взаємозв'язки між компонентами гарантують логічний і структурований процес. Наприклад, після вибору оголошення інформація передається в модуль бронювання, а далі – до платіжної системи, яка обробляє фінансові операції та підтверджує транзакції. Додатково, інтеграція із

системою чату забезпечує прозорий обмін повідомленнями між орендарем та орендодавцем.

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

3.4 Організаційна структура програмного забезпечення

3.4.1 Діаграма пакетів. Діаграма пакета – це тип UML-діаграми, яка візуалізує загальну структуру системи шляхом організації пов’язаних класів у пакети. Вона показує, як ці пакети взаємодіють один з одним, пропонуючи загальне уявлення про архітектуру системи. Цей тип діаграми особливо корисний під час керування складними програмами, оскільки він дозволяє використовувати більш модульний підхід, розбиваючи систему на менші, керовані одиниці. Для програми пошуку та оренди житла діаграма пакета може допомогти структурувати різні компоненти та виділити зв’язки між ними [17].

У контексті цієї програми пакет рівня представлення включає всі компоненти, відповідальні за взаємодію з користувачем. Це включає контролери інтерфейсу користувача, які керують подіями та взаємодією користувачів, представлення, що представляють візуальні елементи, такі як оголошення про нерухомість або запити на оренду, та компоненти, які обробляють загальний потік інтерфейсу користувача.

Рівень бізнес-логіки обробляє основні функції та правила системи. У цьому рівні розташовуються модулі, пов’язані з управлінням орендою, управлінням оголошеннями, управлінням користувачами та сповіщеннями. Ці компоненти є важливими для роботи програми, забезпечуючи обробку запитів на оренду, керування користувачами своїми профілями, оновлення оголошень та отримання користувачами сповіщень про важливі дії, такі як схвалення або оновлення повідомлень.

Рівень доступу до даних зосереджений на взаємодії між програмою та базою даних. Цей пакет містив би репозиторії, які абстрагують логіку виконання операцій CRUD, керування підключеннями до бази даних та використання системи ORM (об'єктно-реляційного відображення), такої як Doctrine. Рівень ORM відповідає за відображення таких сутностей, як користувачі та списки, у відповідні таблиці бази даних, оптимізуючи пошук та маніпулювання даними.

Рівень бази даних містить фактичні моделі даних або сутності, що використовуються програмою. Це включало б пакети для таких сутностей, як користувачі, списки, запити на оренду та повідомлення. Ці моделі визначають структуру даних, що зберігаються в базі даних, та забезпечують узгодженість під час взаємодії із системою зберігання.

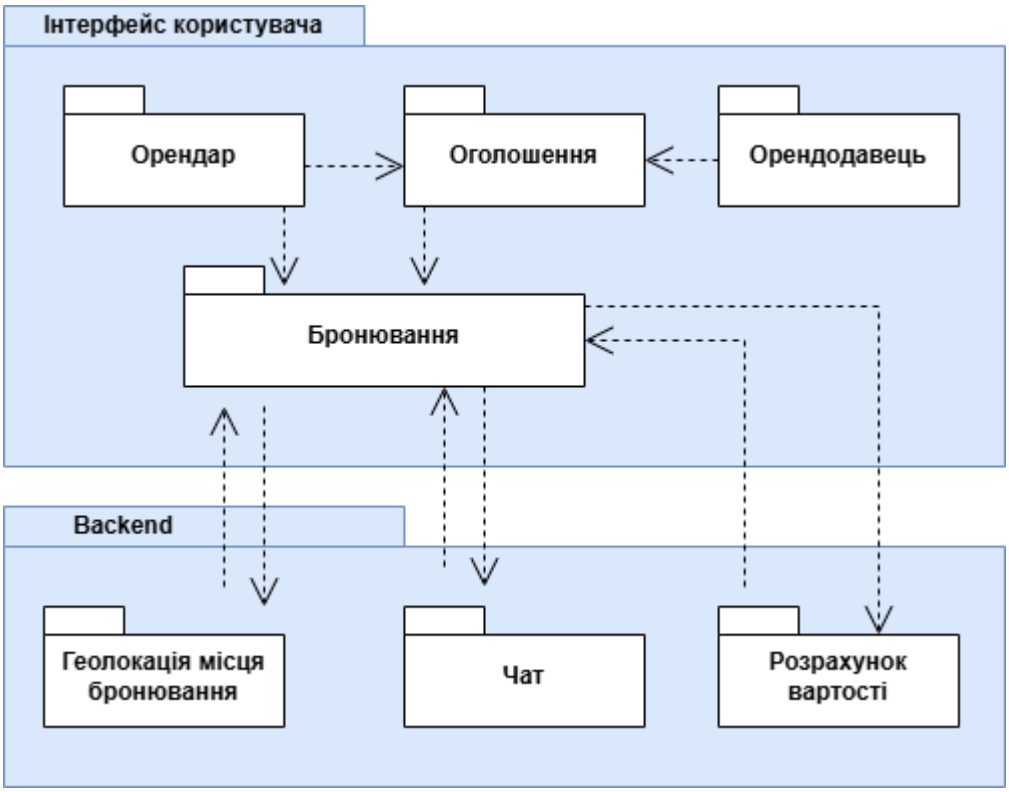

Рис. 3.4 Діаграма пакетів

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

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

Крім того, представлено пакет платежів, який відповідає за обробку транзакцій, розрахунок вартості оренди та інтеграцію з фінансовими сервісами. Пакет комунікації містить механізми обміну повідомленнями, гарантуючи зручний зв’язок між орендарем та орендодавцем. Завершує архітектуру пакет користувачів, де зберігаються дані про зареєстрованих осіб, їхні профілі та права доступу.

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

3.5 Вибір інструментарію для створення програмного забезпечення

Для розробки програмного забезпечення, спрямованого на пошук та оренду житла, ретельний вибір інструментів має вирішальне значення для забезпечення створення надійної, масштабованої та ефективної системи. Обрані інструменти спеціально підібрані для задоволення різних аспектів процесу розробки програмного забезпечення, від розробки серверної частини до створення користувацького інтерфейсу, а також для забезпечення безперебійного управління даними. Нижче наведено огляд обраних інструментів та обґрунтування їхнього вибору.

PHP, широко використовувана мова серверних сценаріїв, була обрана основною мовою програмування для розробки серверної частини програми. Вона добре підходить для створення динамічних та адаптивних веб-додатків, пропонуючи міцну основу для серверної логіки та обробки даних. Відкритий

код PHP та велика спільнота розробників забезпечують значні ресурси, підтримку та попередньо створені бібліотеки, що значно скорочує час розробки.

Symfony, надійний PHP-фреймворк, доповнює PHP, надаючи набір компонентів та інструментів PHP, що можуть бути повторно використані, для створення складних веб-додатків. Він сприяє використанню найкращих практик кодування, таких як архітектура MVC (Model-View-Controller), та включає вбудовані інструменти для маршрутизації, управління сеансами та шаблонизації. Завдяки обширній документації, масштабованості та гнучкості Symfony, він є ідеальним вибором для створення масштабованого та зручного програмного забезпечення, такого як система пошуку та оренди житла.

Для керування даними програми було обрано MySQL, широко використовувану реляційну систему керування базами даних (RDBMS). Вона є високоефективною, надійною та підтримує SQL, стандартну мову для запитів до реляційних баз даних. MySQL пропонує швидке отримання та маніпулювання даними, що є важливим для житлової платформи, де користувачі часто шукатимуть, фільтруватимуть та взаємодіятимуть зі списками, запитами на оренду та повідомленнями. Крім того, MySQL добре підтримується Symfony та безперебійно працює з Doctrine ORM (Об'єктно-реляційне відображення), забезпечуючи безперебійну взаємодію з базою даних.

Doctrine ORM — це інструмент об'єктно-реляційного відображення для PHP, який спрощує взаємодію з базами даних, дозволяючи розробникам працювати з сутностями бази даних як з об'єктами PHP. Ця абстракція зменшує потребу в прямих SQL-запитах, роблячи кодову базу чистішою, зручнішою для обслуговування та простішою для розуміння. Doctrine ORM тісно інтегрована з Symfony, що дозволяє безперебійно обробляти дані, перевіряти їх та автоматично генерувати схеми. Використовуючи Doctrine, розробники можуть ефективно працювати з реляційною базою даних, не

турбуючись про низькорівневі деталі бази даних, що спрощує розробку та підвищує продуктивність.

Фронтенд застосунку побудовано за допомогою HTML, CSS та JavaScript, які є основними технологіями для створення веб-інтерфейсів. HTML забезпечує структурну основу для веб-сторінок, визначаючи такі елементи, як заголовки, форми та посилання. CSS використовується для стилізації цих елементів, забезпечуючи візуальну привабливість, адаптивність та зручність використання застосунку. Завдяки сучасним дизайнерським фреймворкам, таким як Bootstrap, або користувацькому стилюванню, CSS дозволяє створювати інтуїтивно зрозумілий макет, який покращує взаємодію з користувачем.

JavaScript, мова сценаріїв на стороні клієнта, додає інтерактивності застосунку. Він дозволяє динамічно оновлювати сторінку без необхідності повного перезавантаження сторінки, що важливо для таких функцій, як пошук у реальному часі, сповіщення про повідомлення та надсилання форм. Бібліотеки та фреймворки JavaScript, такі як jQuery або Vue.js, можна використовувати для покращення функціональності та керування складними взаємодіями інтерфейсу користувача. Поєднання цих трьох технологій гарантує, що застосунок для оренди житла буде як функціональним, так і візуально привабливим.

Visual Studio Code (VS Code) – це потужний, легкий та високонастроюваний редактор коду, обраний для процесу розробки. VS Code пропонує інтуїтивно зрозумілий інтерфейс користувача та надійні функції, такі як підсвічування синтаксису, IntelliSense, інструменти налагодження та інтеграція з Git, що робить його ідеальним для розробки на PHP, HTML, CSS та JavaScript. Він також підтримує широкий спектр розширень, що підвищують продуктивність, таких як PHP IntelliSense, підтримка Doctrine та GitLens для покращеного керування версіями. Гнучкість та багатий набір функцій VS Code гарантують, що розробники можуть ефективно працювати над завданнями розробки як на серверній, так і на фронтендній стороні.

4 ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЯ СИСТЕМИ

4.1 Вимоги до апаратного та програмного забезпечення

Для розробки надійного програмного рішення для пошуку та оренди житла важливо визначити вимоги як до апаратного, так і до програмного забезпечення, щоб забезпечити безперебійну функціональність та оптимальний користувацький досвід. Ці специфікації застосовуються як до середовища розробки, так і до виробничого середовища, враховуючи потреби програми на різних етапах її життєвого циклу.

Щодо середовища розробки, апаратне забезпечення повинно бути здатним обробляти ресурсомісткі завдання, такі як кодування, тестування та налагодження. Для забезпечення безперебійної роботи з інструментами розробки рекомендується багатоядерний процесор, такий як Intel i5 або AMD Ryzen 5. Щонайменше 8 ГБ оперативної пам'яті є важливим для запобігання уповільненням під час роботи інтегрованих середовищ розробки (IDE), локальних серверів або систем баз даних. Для зберігання даних має бути твердотільний накопичувач (SSD) з мінімальною ємністю 256 ГБ, щоб забезпечити швидше завантаження та зменшити затримки. Стабільне підключення до Інтернету зі швидкістю щонайменше 10 Мбіт/с важливе для завантаження залежностей, доступу до репозиторіїв та роботи локальних серверів або інструментів розробки. Для найкращого досвіду рекомендується дисплей Full HD (1920x1080) для покращення видимості та чіткості під час кодування. Що стосується операційної системи, то ОС на базі Linux, така як Ubuntu, ідеально підходить для веб-розробки, хоча PHP та Symfony також безперебійно працюватимуть у Windows або macOS.

У робочому середовищі обладнання має бути масштабованим для підтримки кількох користувачів та обробки більших навантажень. Для

управління підвищеним обчислювальним навантаженням необхідний процесор серверного класу, такий як Intel Xeon або AMD EPYC. Для забезпечення безперебійної роботи за високого трафіку рекомендується 16 ГБ або більше оперативної пам'яті, а для сайтів з дуже високим трафіком потрібна ще більша ємність. Високопродуктивний SSD-накопичувач з обсягом пам'яті щонайменше 1 ТБ забезпечить швидкий доступ до даних програми, списків оренди та інформації, пов'язаної з користувачами. Для забезпечення швидкої передачі даних та мінімальних затримок для користувачів потрібне надійне інтернет-з'єднання зі швидкістю щонайменше 100 Мбіт/с. Серверна ОС на базі Linux, така як Ubuntu або CentOS, оптимально підходить для роботи програми, забезпечуючи стабільність та безпеку. Крім того, безпечна система резервного копіювання, як локально, так і через хмарне сховище, є важливою для захисту даних та забезпечення швидкого відновлення у разі збоїв системи.

Щодо вимог до програмного забезпечення, інструменти та технології розробки становитимуть основу застосунку. Основною мовою програмування буде PHP, з версією 8.0 або вище, рекомендованою для його найновіших функцій та оптимізацій. Symfony, потужний PHP-фреймворк, буде використовуватися для структурування серверної частини, пропонуючи компоненти багаторазового використання та ефективну маршрутизацію. MySQL служитиме системою керування реляційними базами даних (RDBMS) для обробки зберігання критично важливих даних, таких як профілі користувачів, списки оренди та запити. Doctrine ORM, інтегрована з Symfony, керуватиме операціями з базою даних та спрощуватиме взаємодію з нею, дозволяючи розробникам працювати з високорівневими сутностями замість сирих SQL-запитів. Для фронтенду будуть використовуватися стандартні веб-технології — HTML, CSS та JavaScript — для створення зручного інтерфейсу, який забезпечує інтерактивний досвід. Visual Studio Code (VS Code) буде основним редактором коду, пропонуючи ряд розширень для покращення робочого процесу розробки, таких як автодоповнення коду, налагодження та підтримка контролю версій.

Контроль версій буде здійснюватися за допомогою Git, що дозволить розробникам відстежувати зміни коду та ефективно співпрацювати без конфліктів. Composer, менеджер залежностей PHP, спростить процес керування зовнішніми бібліотеками та пакетами, необхідними для програми. Для розробки фронтенду Node.js та npm будуть використовуватися для керування бібліотеками JavaScript та інструментами створення, забезпечуючи сумісність та оптимізуючи код для продакшену. Крім того, Docker може бути впроваджений для створення ізольованих середовищ для програми, покращуючи узгодженість на різних етапах розгортання.

Що стосується середовища продакшену, то для обробки HTTP-запитів та обслуговування як статичного, так і динамічного контенту буде використовуватися веб-сервер, такий як Apache або Nginx. Для безпеки та шифрування буде встановлено SSL-сертифікат, щоб забезпечити безпечний зв'язок між користувачами та сервером. Інструменти моніторингу, такі як New Relic або Datadog, будуть використовуватися для відстеження продуктивності програми та виявлення будь-яких проблем, які можуть вплинути на взаємодію з користувачем. Для забезпечення цілісності даних буде впроваджено автоматизоване програмне забезпечення для резервного копіювання, щоб запобігти втраті даних, гарантуючи захист інформації про користувачів та інших важливих даних.

Підсумовуючи, описані вище апаратні та програмні характеристики забезпечать міцну основу для розробки, розгортання та підтримки програми пошуку та оренди житла. Використовуючи сучасні інструменти розробки, такі як PHP, Symfony, MySQL та інші веб-технології, додаток зможе обробляти великі обсяги трафіку.

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

Рис. 4.1 Діаграма розгортання

4.2 Тестування системи

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

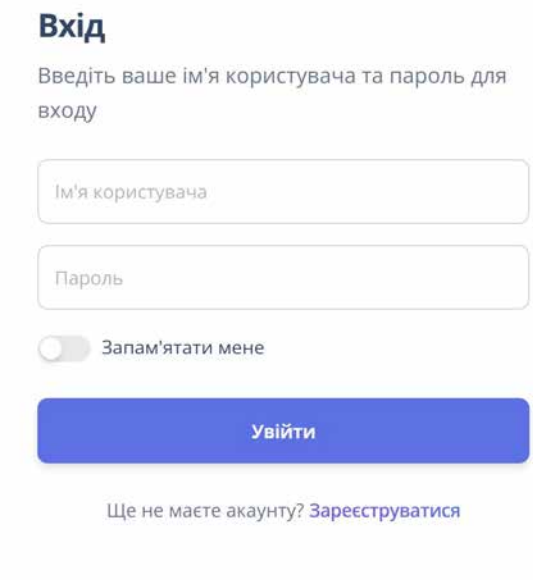

Рис. 4.2 Форма авторизації

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

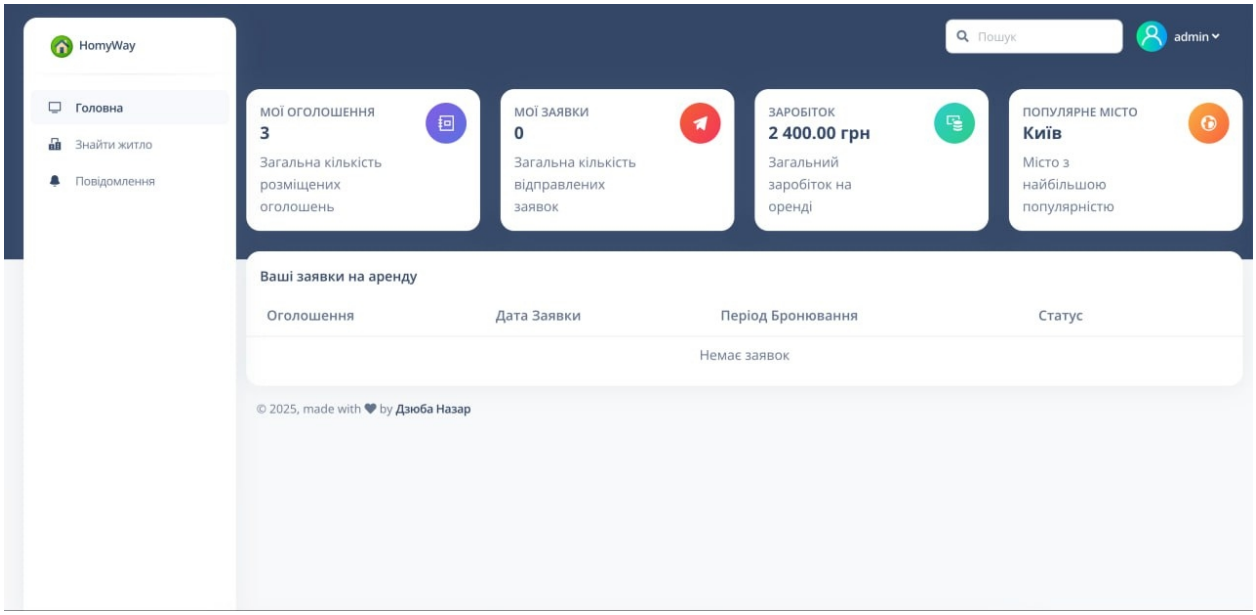

Рис. 4.3 Головна сторінка

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

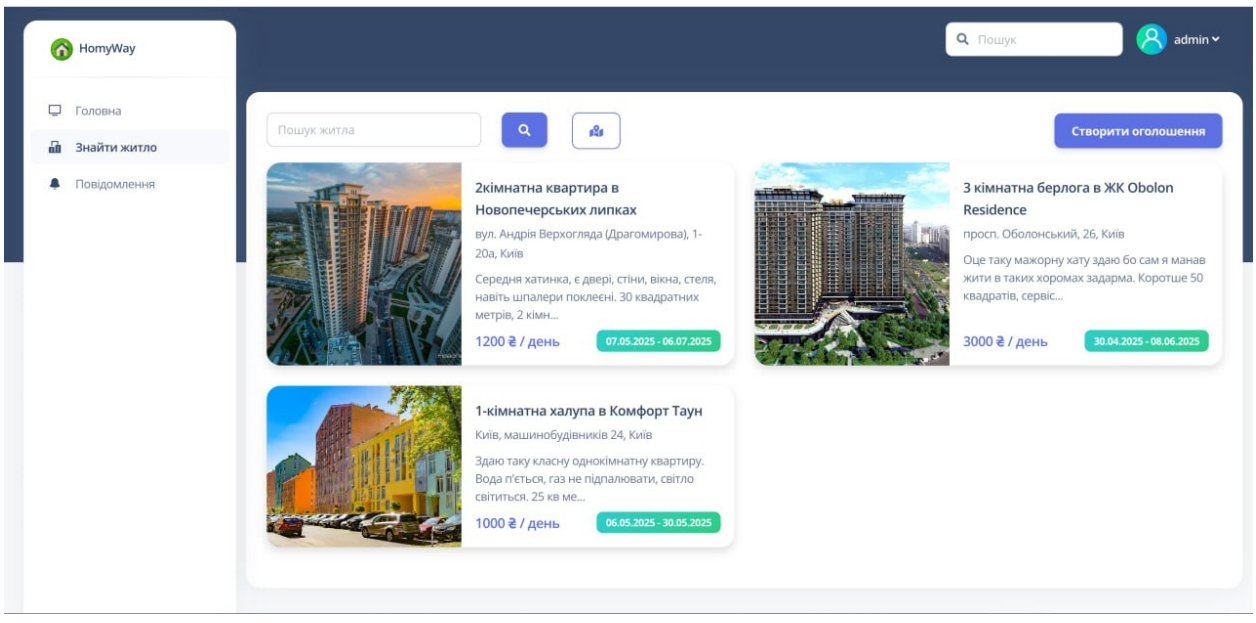

Рис. 4.4 Сторінка “Знайти житло”

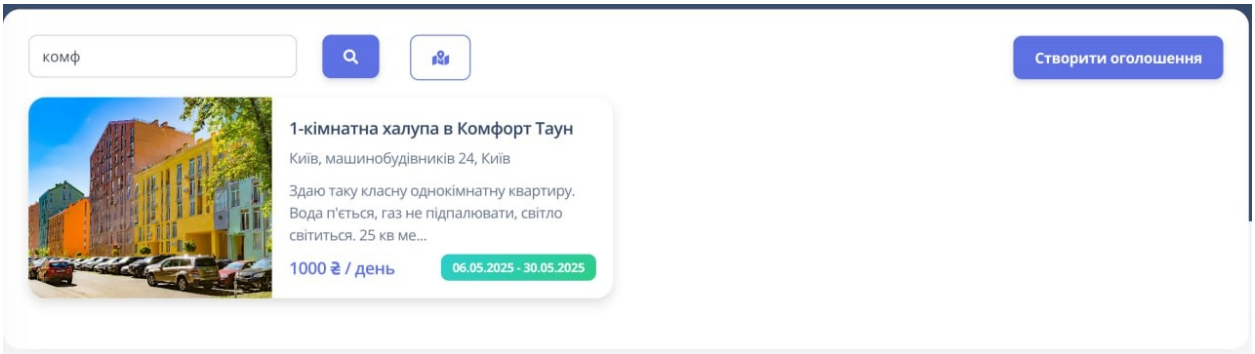

Рис. 4.5 Пошук за критеріями

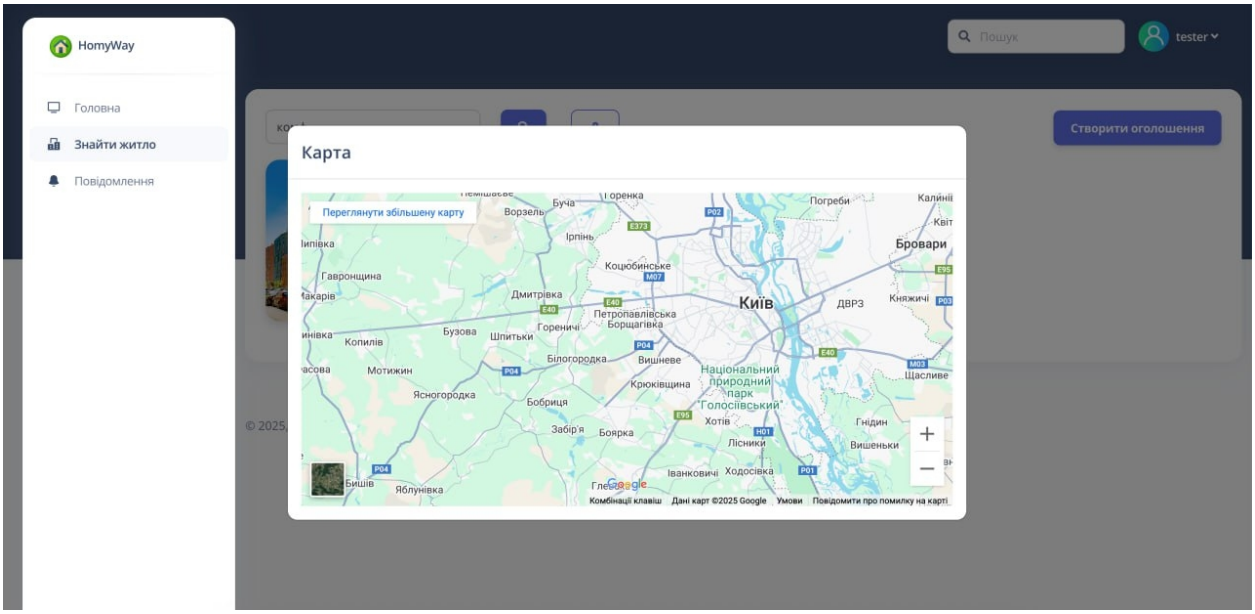

Рис. 4.6 Інтерактивна карта

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

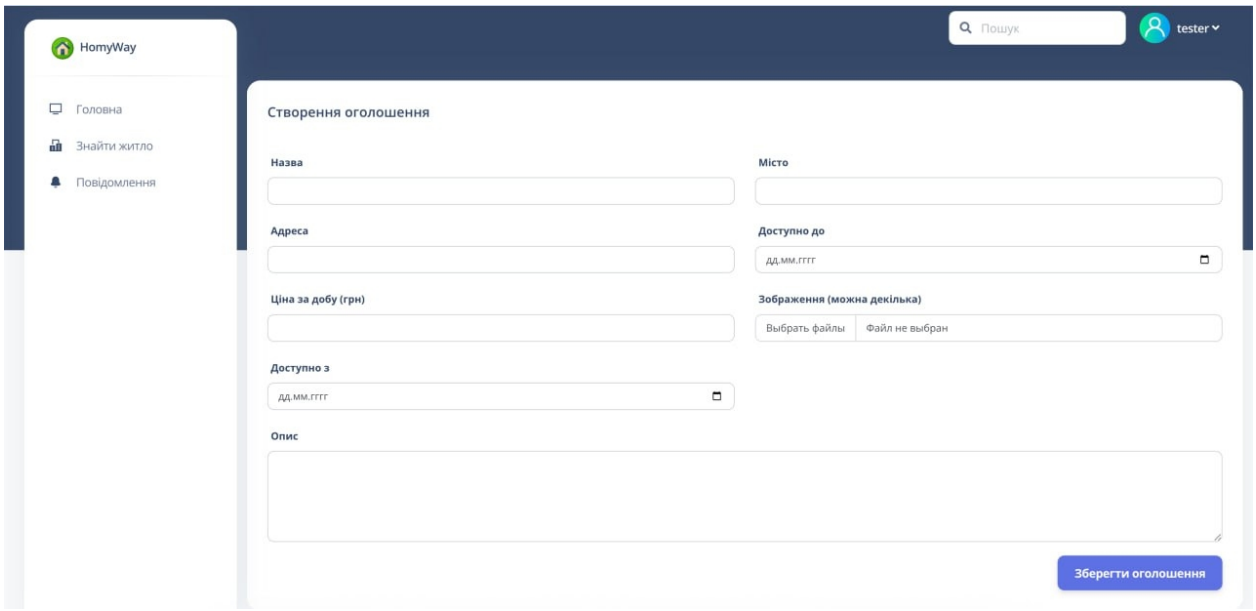

Рис. 4.7 Сторінка створення оголошення

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

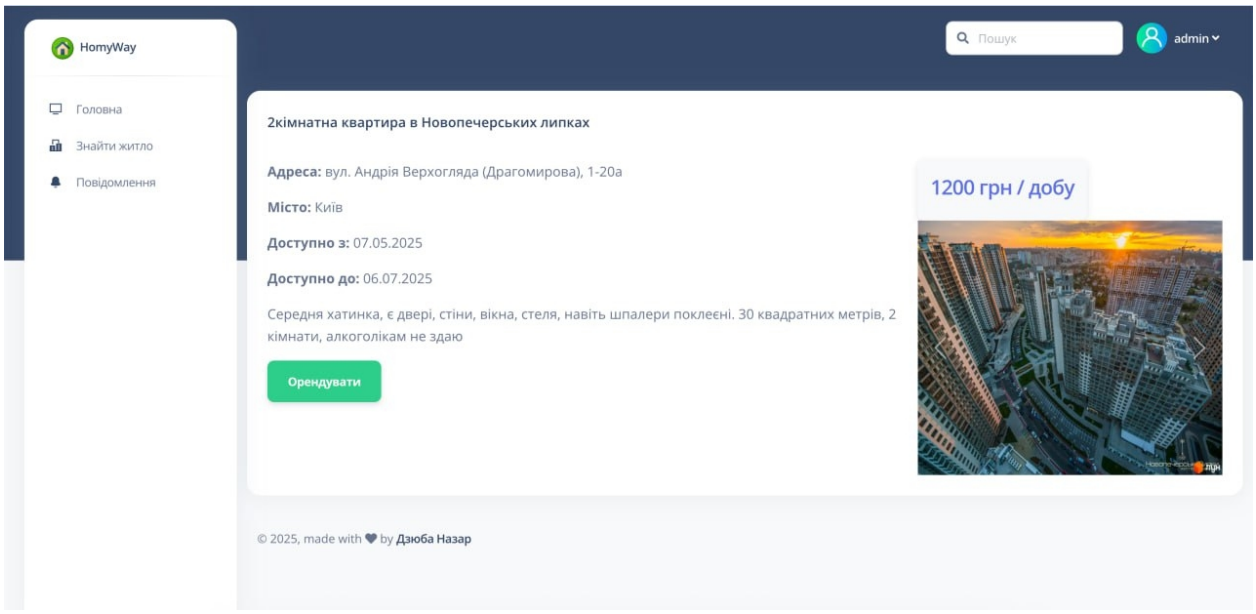

Рис. 4.8 Сторінка новоствореного оголошення

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

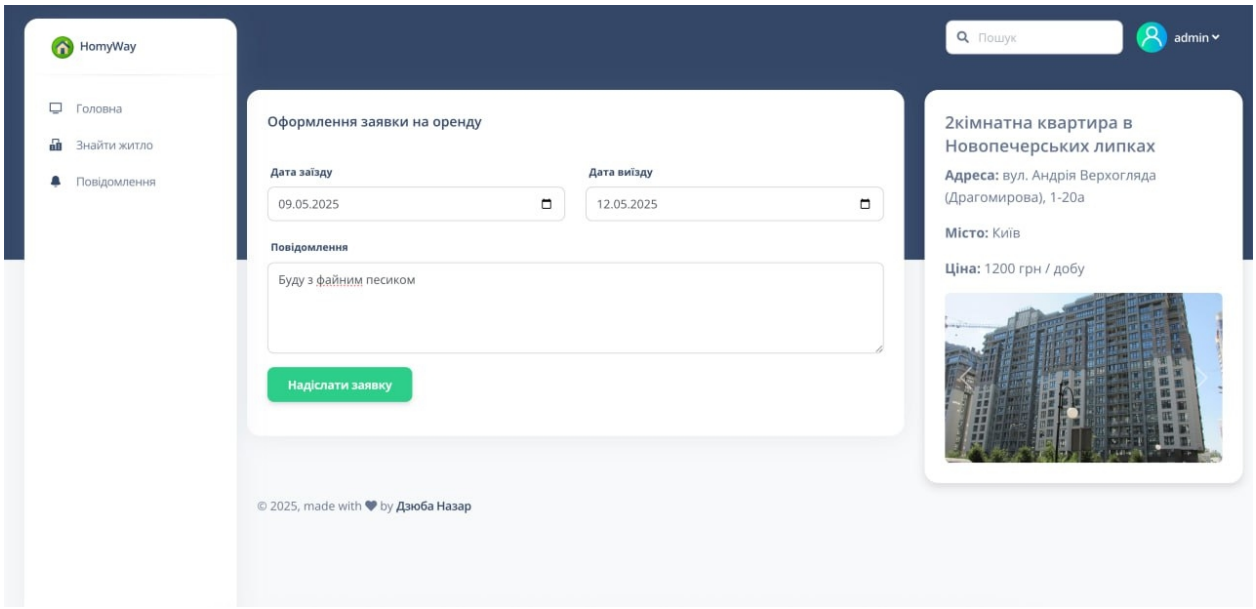

Рис. 4.9 Оформлення заявки на оренду

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

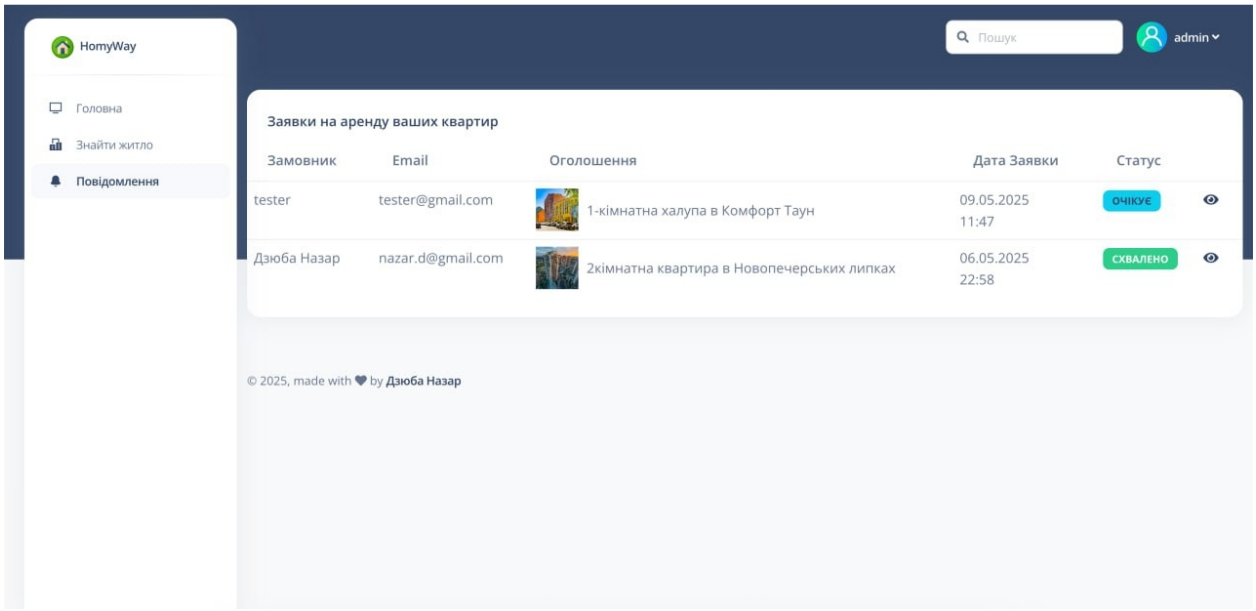

Рис. 4.10 Сторінка “Повідомлення”

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

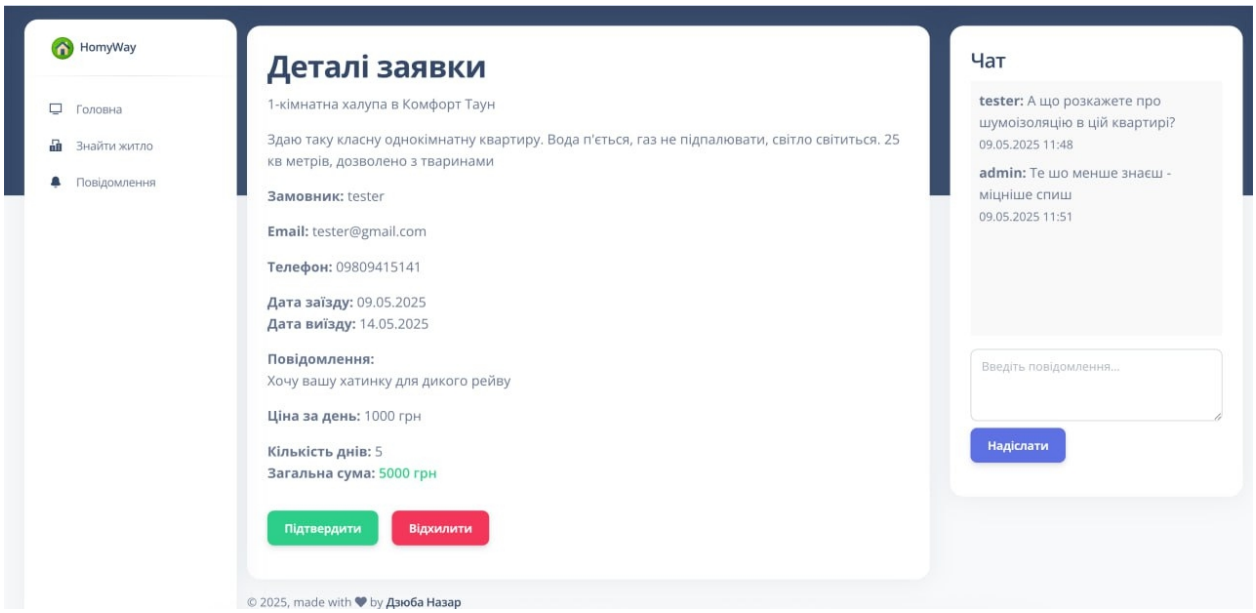

Рис. 4.11 Деталі заявки на бронювання

ВИСНОВКИ

На завершення, розробка програмного забезпечення для пошуку та оренди житла відповідає на значну потребу сучасного ринку нерухомості, пропонуючи як орендарям, так і орендодавцям безперебійну, ефективну та зручну платформу для управління нерухомістю та орендних операцій. Обрані технології — PHP, Symfony, MySQL, Doctrine ORM, HTML, CSS та JavaScript — забезпечують надійну основу для створення масштабованого, безпечного та високофункціонального додатку. Ці інструменти, разом з архітектурними рішеннями та дизайном бази даних, забезпечують здатність системи обробляти великий трафік, забезпечувати обробку даних у режимі реального часу та надавати інтуїтивно зрозумілий інтерфейс користувача.

Робота розпочалася з вивчення предметної області та формулювання проблеми дослідження. Існуючі рішення були переглянуті, щоб виявити прогалини та можливості для вдосконалення програмного забезпечення. Це послужило основою для наступних етапів роботи.

Було створено логічну модель даних, що представляє вимоги до інформації та зв’язки в програмному забезпеченні. Ця модель лягла в основу проєктування програмної системи. Розроблено фізичну модель даних, що забезпечує ефективне зберігання та пошук даних за допомогою вибраної системи керування базами даних.

Рішення використовувати 4-рівневу архітектуру — презентація, додаток, домен та доступ до даних — забезпечує чіткий розподіл обов'язків, що спрощує обслуговування, тестування та масштабування програмного забезпечення. Така структура сприяє модульності, дозволяючи незалежні оновлення та вдосконалення різних частин системи без порушення роботи інших рівнів. Крім того, дизайн бази даних з ретельно продуманими зв'язками між таблицями, такими як User, Listing, RentalRequest та Message, гарантує

організованість та узгодженість потоку даних, що сприяє ефективним запитам та безперебійній взаємодії з користувачами.

Вимоги до апаратного та програмного забезпечення, викладені в роботи, гарантують, що система оптимізована як для середовища розробки, так і для виробничого середовища. Завдяки масштабованій серверній архітектурі та безпечним, надійним програмним інструментам, програма готова обробляти реальні вимоги, від перегляду оголошень та подання запитів на оренду до управління зв'язком між користувачами.

Цей програмний проєкт не лише відповідає основним потребам користувачів, які шукають та пропонують нерухомість в оренду, але й забезпечує надійну та безпечну платформу для управління цими транзакціями. Впровадження сучасних веб-технологій забезпечує гнучкість системи та її здатність розвиватися відповідно до майбутніх тенденцій на ринку житла, що робить її актуальним та ефективним рішенням для користувачів сьогодні та в майбутньому. Зрештою, цей проєкт пропонує комплексний підхід до створення програмного забезпечення для пошуку та оренди житла, а його успішна розробка та впровадження нададуть значні переваги як кінцевим користувачам, так і керуючим нерухомістю.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гілмор, Р., Діксон, М. Системи управління нерухомістю та майном: інструменти для управління житловою та комерційною нерухомістю. — Routledge, 2019.
2. Хьюз, Р. Цифрова трансформація в нерухомості: роль технологій у пошуку та оренді нерухомості. — Springer, 2021.
3. Міллер, Дж. Розвиток нерухомості: практичний посібник. — McGraw-Hill Education, 2017.
4. Airbnb – [Електронний ресурс] – Режим доступу: https://www.airbnb.com.ua/stays/monthly
5. Booking – [Електронний ресурс] – Режим доступу: https://www.booking.com/index.ua.html
6. Сайденберг, А., Ліор, С. Технології нерухомості: інновації в пошуку та оренді нерухомості. — Wiley, 2020.
7. Стюарт, Б. Створення платформ нерухомості: програмні рішення для оренди та продажу нерухомості. — Apress, 2018.
8. Лейнг, Р., Трухільйо, Р. Управління нерухомістю в цифрову епоху. — Palgrave Macmillan, 2019.
9. Танненбаум, С. Хмарні обчислення для управління нерухомістю: сучасний підхід. — Wiley, 2020.
10. Сасман, Л. Технології нерухомості: тенденції та застосування в системах пошуку та оренди нерухомості. — Pearson, 2021.
11. Гаррісон, Дж., Рід, А. Сучасні рішення для нерухомості: управління списками житла, орендою та транзакціями за допомогою програмного забезпечення. — Cambridge University Press, 2018.
12. Кім, Д. Системи розумного житла: використання програмного забезпечення для підвищення ефективності оренди нерухомості. — Elsevier, 2020.

13. Doctrine ORM – [Електронний ресурс] – Режим доступу: https://www.doctrine-project.org/projects/orm.html
14. Документація Symfony – [Електронний ресурс] – Режим доступу: https://symfony.com/doc/current/index.html
15. Документація MySQL – [Електронний ресурс] – Режим доступу: https://dev.mysql.com/doc/
16. Посібник з PHP – [Електронний ресурс] – Режим доступу: https://www.php.net/manual/en/
17. Документація Git – [Електронний ресурс] – Режим доступу: https://git-scm.com/doc
18. Посібник з HTML W3Schools – [Електронний ресурс] – Режим доступу: https://www.w3schools.com/html/
19. Посібник з CSS W3Schools – [Електронний ресурс] – Режим доступу: https://www.w3schools.com/css/
20. Посібник з JavaScript W3Schools – [Електронний ресурс] – Режим доступу: https://www.w3schools.com/js/
21. "«Важливість систем управління корпоративним навчанням» – [Електронний ресурс] – Режим доступу: https://elearningindustry.com/importance-corporate-learning-management-system
22. «Найкращі практики управління корпоративним навчанням» – [Електронний ресурс] – Режим доступу: https://www.simplilearn.com/corporate-training-best-practices-article
23. Інструмент для створення діаграм Draw.io – [Електронний ресурс] – Режим доступу: https://app.diagrams.net/
24. ПРОЄКТУВАННЯ ER-ДІАГРАМИ – [Електронний ресурс] – Режим доступу: https://nationalteam.worldskills.ua/skills/proektirovanie-er-diagrammy/

25. Діаграма варіантів використання (UseCase diagram) – [Електронний ресурс] – Режим доступу: https://flexberry.github.io/ua/fd_use-case-diagram.html
26. ПРОЄКТУВАННЯ USE CASE ДІАГРАМИ. ВИЗНАЧЕННЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ СИСТЕМИ – [Електронний ресурс] – Режим доступу: https://nationalteam.worldskills.ua/skills/proektirovanie-use-case-diagrammy-opredelenie-funktsionalnykh-vozmozhnostey-sistemy/
27. What is Sequence Diagram? – [Електронний ресурс] – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/
28. UML - Activity Diagrams – Tutorialspoint – [Електронний ресурс] – Режим доступу: https://www.tutorialspoint.com/uml/uml_activity_diagram.htm
29. Побудова діаграми класів – [Електронний ресурс] – Режим доступу: https://flexberry.github.io/ua/gpg_class-diagram.html
30. UML-діаграми класів – [Електронний ресурс] – Режим доступу: https://prog-cpp.ua/uml-classes/
31. Entity Relationship Diagram - Data Modeling – [Електронний ресурс] – Режим доступу: https://www.visualparadigm.com/VPGallery/datamodeling/EntityRelationshipDiagram.html
32. UML 2 Tutorial - Package Diagram - Sparx Systems – [Електронний ресурс] – Режим доступу: https://sparxsystems.com/resources/tutorials/uml2/package-diagram.html
33. Документація Bootstrap (офіційний сайт) – [Електронний ресурс] – Режим доступу: www.getbootstrap.com/documentation.

ДОДАТОК А

Фрагменти програмного коду. Функція створення оголошення

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

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

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

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

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

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

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

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

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

            $entityManager->flush();
        }

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

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

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

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

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

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

ДОДАТОК Б

Фрагменти програмного коду. Функціонал бронювання житла

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

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

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

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

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

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

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

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

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

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

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

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

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

```
}
```