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 онлайн сервісу для купівлі-продажу нерухомості»

Спеціальність 121 – «Інженерія програмного забезпечення»

Гарант освітньої програми
к.т.н., доц. _____ Вайганг Г.О
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи
к.е.н., ст.викладач _____ Ніколаєнко Д.В.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Андрєєв Богдан Олексійович
(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
Андрєєв Богдан Олексійович

Спеціальність 121 – «Інженерія програмного забезпечення»

1. Тема бакалаврської кваліфікаційної роботи «Програмне забезпечення онлайн сервісу для купівлі-продажу нерухомості» Затверджена наказом ректора НУБіП України № 2248 "С" від 16.12.2024
2. Термін подання завершеної роботи на кафедру 2025 . 05 . 25
3. Вихідні дані: надання інформації про доступні для купівлі об'єкти нерухомості, їх розташування та умови продажу.
4. Перелік питань, що розглядаються:
   - Аналіз проблемної області
   - Моделювання предметної області
   - Проєктування програмної системи
   - Впровадження та експлуатація системи

Керівник кваліфікаційної роботи

к.е.н., ст.викладач		Ніколаєнко Д.В.
(науковий ступінь та вчене звання)	(підпис)	(ПІБ)

Завдання прийняв до виконання		Андрєєв Б.О.
	(підпис)	(ПІБ)

Дата отримання завдання "___"_____ 2025 р.

ЗМІСТ

ВСТУП

У сучасну цифрову епоху процес купівлі-продажу нерухомості значно еволюціонував завдяки швидкому розвитку інформаційних технологій. Традиційні методи пошуку нерухомості через газети чи фізичні агентства поступово замінюються більш ефективними веб-рішеннями. Зростаючий попит на зручність, доступність та швидкість на ринку нерухомості призвів до появи численних онлайн-платформ, які дозволяють користувачам публікувати, переглядати та керувати оголошеннями про нерухомість з будь-якої точки світу.

Незважаючи на наявність великих веб-сайтів з нерухомості, багато платформ перевантажені рекламою, не мають персоналізованих інструментів комунікації або не враховують належним чином потреби окремих користувачів. Це створює можливість для розробки більш спеціалізованих та зручних послуг, адаптованих до потреб приватних покупців та продавців.

Ця бакалаврська робота присвячена розробці веб-додатку, який забезпечує простий та ефективний спосіб для користувачів публікувати оголошення про нерухомість, переглядати доступні об'єкти та спілкуватися безпосередньо через платформу. Система дозволяє продавцям створювати детальні оголошення з фотографіями та важливими параметрами, такими як ціна, площа та місцезнаходження. Покупці, у свою чергу, можуть використовувати потужні інструменти фільтрації, щоб знаходити нерухомість, яка відповідає їхнім критеріям, та надсилати пропозиції власникам нерухомості. Вбудована система чату дозволяє користувачам безпечно обговорювати деталі транзакцій на платформі.

Програмне забезпечення розроблено з використанням мови програмування PHP, веб-фреймворку Symfony та бази даних MySQL. Ці технології були обрані за їхню продуктивність, гнучкість та придатність для створення надійних веб-додатків.

Основною **метою** цієї роботи є покращення поточної індустрії продажу нерухомості за допомогою розробки програмного забезпечення для веб-платформи, яка дозволяє користувачам купувати та продавати нерухомість онлайн. Система має на меті забезпечити функціональне, безпечне та зручне рішення для розміщення оголошень про нерухомість, управління пропозиціями та полегшення комунікації між покупцями та продавцями.

Ринок нерухомості продовжує рухатися до цифровізації. Зі зростанням кількості користувачів, які віддають перевагу пошуку та пропонуванні нерухомості онлайн, існує очевидна потреба в спеціалізованому програмному забезпеченні, яке спрощує цей процес. Багато існуючих платформ або перевантажені контентом, або мають складні інтерфейси користувача, або не мають важливих функцій, таких як прямі канали зв'язку між сторонами. Тому розробка індивідуального онлайн-сервісу, який вирішує ці проблеми, є дуже **актуальною** та практичною в сучасному контексті. Він відображає сучасні технологічні тенденції та відповідає реальним потребам користувачів.

Об'єктом дослідження є процес взаємодії між покупцями та продавцями нерухомості в цифровому середовищі за допомогою веб-технологій.

Предметом дослідження є методи та програмні засоби, що використовуються для реалізації веб-застосунку для онлайн-купівлі та продажу нерухомості, включаючи управління оголошеннями про продаж, функції пошуку, подання пропозицій та обмін повідомленнями на платформі.

Цей проєкт демонструє, як сучасні програмні інструменти можуть бути застосовані для вирішення практичних проблем у сфері нерухомості, пропонуючи масштабоване, ефективне та орієнтоване на користувача рішення. Робота включає повний цикл розробки програмного забезпечення — від аналізу вимог та проєктування системи до впровадження, тестування та оцінки результатів.

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Опис предметної області

В останні роки ринок нерухомості зазнав значного зсуву в бік цифрових платформ, що відображає ширшу тенденцію цифровізації в усіх галузях. З широким доступом до Інтернету та зростаючою перевагою онлайн-сервісів, все більше людей обирають пошук та рекламу нерухомості через веб-додатки. Ця трансформація принесла більшу зручність, швидший доступ до інформації про нерухомість та зменшила залежність від посередників, таких як агенти з нерухомості.

Онлайн-платформи нерухомості стали важливими інструментами для покупців та продавців. Вони дозволяють користувачам переглядати велику кількість оголошень, порівнювати ціни, оцінювати характеристики нерухомості та приймати обґрунтовані рішення. Однак, незважаючи на зростання кількості платформ на ринку, багато існуючих рішень не забезпечують безперебійного та зручного користувацького досвіду. Великі, усталені сервіси часто зосереджуються на комерційних оголошеннях та агентствах нерухомості, що ускладнює для окремих осіб навігацію в інтерфейсі або самостійне керування власними оголошеннями. Крім того, спілкування між покупцями та продавцями зазвичай обмежене, що вимагає від користувачів покладатися на зовнішні програми для обміну повідомленнями або телефонні дзвінки для завершення угод [1].

Такі сучасні системи, як Zillow, Dom.Ria, Realtor.com та інші, пропонують потужні функції, але вони часто перевантажені рекламою та даними, що може бути непосильним для звичайних користувачів. Цим платформам також може бракувати персоналізації, вони не можуть адаптуватися до конкретних потреб приватних продавців або покупців, які купують нерухомість вперше. У багатьох випадках користувачі повідомляють

про труднощі з керуванням своїми оголошеннями або узгодженням пропозицій безпосередньо в системі [2].

Детальніше розглянемо очікування користувачів, і вони виявляють попит на платформи, які надають пріоритет простоті, прозорості та безпечній взаємодії. Покупці хочуть швидко знаходити нерухомість за допомогою розширених інструментів фільтрації, тоді як продавці шукають інтуїтивно зрозумілий процес публікації оголошень, завантаження зображень та виділення ключових деталей. Ключовою функцією, якої часто бракує на багатьох платформах, є вбудована система обміну повідомленнями, яка забезпечує пряме спілкування між двома сторонами в режимі реального часу.

Крім того, на ринку існує прогалина для легких платформ, які обслуговують окремих користувачів, надаючи їм усі необхідні інструменти, не перевантажуючи їх функціями корпоративного рівня. Користувачі хочуть мати більше контролю над процесом купівлі-продажу в рамках єдиної платформи, яка підтримує весь життєвий цикл транзакції — від пошуку до комунікації та узгодження.

З огляду на ці проблеми та обмеження існуючих рішень, існує очевидна потреба в новому програмному продукті, адаптованому до потреб приватних осіб, які бажають купити або продати нерухомість. Така система повинна поєднувати простоту використання з основними функціями, дозволяючи користувачам керувати оголошеннями та пропозиціями нерухомості в безпечному та зручному середовищі. Усуваючи недоліки існуючих платформ та зосереджуючись на дизайні, орієнтованому на користувача, запропоноване рішення має потенціал для покращення загальних вражень від операцій з нерухомістю в цифровому просторі.

Ринок нерухомості є динамічною та важливою частиною економіки, де обмін житловою, комерційною та промисловою нерухомістю відіграє життєво важливу роль як в житті окремих осіб, так і в ширших фінансових системах. Традиційно процеси купівлі-продажу нерухомості здійснювалися через особисті зустрічі, паперову документацію та посередників, таких як агенти чи

ріелтори. Однак з розвитком цифрових технологій, особливо розвитку веб-платформ, ці процеси все більше переходять в онлайн, що суттєво змінює те, як люди шукають, оцінюють та укладають угоди з нерухомістю.

Сучасні веб-додатки для нерухомості спрямовані на спрощення взаємодії між власниками нерухомості та потенційними покупцями, пропонуючи централізований цифровий простір для управління рекламою нерухомості та проведення транзакцій. Ці платформи зазвичай дозволяють користувачам реєструватися, публікувати детальні оголошення про нерухомість, завантажувати високоякісні фотографії та описувати ключові характеристики, такі як розмір, розташування, стан та ціна. Водночас покупці отримують переваги від потужного інтерфейсу пошуку, який допомагає їм фільтрувати нерухомість на основі відповідних критеріїв та швидко порівнювати різні варіанти.

Окрім базової функціональності списків, розширені платформи також надають функції прямого зв'язку, дозволяючи користувачам надсилати пропозиції щодо купівлі та ставити запитання, не виходячи з системи. Цей прямий контакт між покупцем і продавцем сприяє прозорості та пришвидшує процес переговорів. Інтегрована система обміну повідомленнями допомагає вести історію спілкування, зменшує залежність від зовнішніх додатків та організовує всі обговорення в межах платформи [3].

Безпека та зручність використання – два найважливіші аспекти програмного забезпечення, розробленого в цій галузі. Платформа повинна гарантувати безпечне зберігання особистих даних користувачів, контактної інформації та деталей нерухомості, а також захищене середовище.

У контексті цієї роботи предметна область включає проєктування та розробку веб-додатку, який підтримує повний цикл угоди з нерухомістю, від розміщення оголошення про нерухомість до спілкування з потенційними покупцями. Система буде побудована за допомогою PHP з фреймворком Symfony та MySQL як реляційною базою даних. Ці технології дозволять створити гнучку, масштабовану та ефективну платформу, яка підтримує

безперебійну взаємодію з користувачем, надійне управління даними та чітку комунікацію, що зрештою покращить досвід онлайн-торгівлі нерухомістю.

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**
User	id	Унікальний ідентифікатор користувача в системі.
	name	Ім'я користувача (може бути використано для особистої ідентифікації).
	password	Пароль користувача для доступу до системи.
	email	Адреса електронної пошти користувача для комунікацій та відновлення пароля.
PropertyListing	id	Унікальний ідентифікатор оголошення про продаж нерухомості.
	title	Назва оголошення, короткий опис нерухомості (наприклад, "Квартира в центрі міста").
	description	Детальний опис об'єкта, включаючи характеристики та стан нерухомості.
	price	Ціна об'єкта нерухомості.
	location	Місцезнаходження об'єкта (місто, район, вулиця).
	area	Площа об'єкта (в квадратних метрах).
	images	Список URL-адрес зображень нерухомості.
Offer	id	Унікальний ідентифікатор пропозиції на покупку нерухомості.
	propertyId	Ідентифікатор нерухомості, на яку зроблено пропозицію.
	buyerId	Ідентифікатор покупця, який зробив пропозицію.
	offerPrice	Пропонована ціна для об'єкта нерухомості.

Ця таблиця описує основні атрибути класів предметної області, що використовуються для створення веб-застосунку для купівлі та продажу нерухомості.

1.2 Огляд існуючих рішень

За останнє десятиліття ринок нерухомості зазнав значної цифрової трансформації, зумовленої зростанням кількості онлайн-платформ, які спрощують операції з нерухомістю для користувачів у всьому світі. Ці платформи сильно відрізняються за функціональністю, інтерфейсом користувача, цільовою аудиторією та технологічною архітектурою. Щоб краще зрозуміти ситуацію та визначити можливості для інновацій, важливо проаналізувати деякі з найвідоміших існуючих рішень.

ЛУН.ua є одним із провідних українських онлайн-сервісів для пошуку нерухомості, який спеціалізується переважно на первинному ринку житла. Ця платформа надає користувачам широкий доступ до актуальних оголошень про продаж та оренду квартир у новобудовах, а також оснащена потужними інструментами для детального аналізу ринку нерухомості в Україні [4].

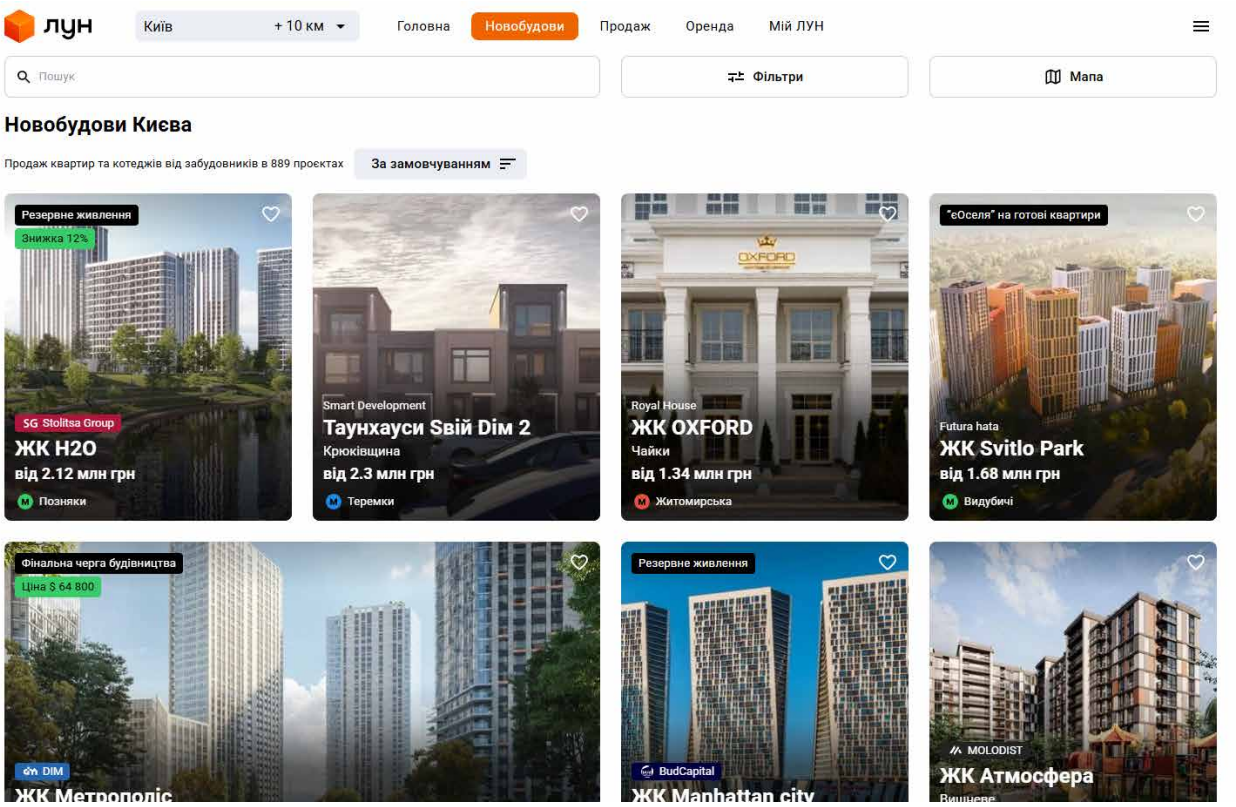

Рис. 1.1 Інформаційна система “Лун”

ЛУН.ua відрізняється своєю здатністю збирати і обробляти величезний обсяг даних — щоденно з понад 160 джерел платформа агрегує близько 2,5 мільйона оголошень. Це забезпечує користувачам найсвіжішу інформацію про об’єкти нерухомості у більш ніж двадцяти містах країни. Інтерактивна карта новобудов, яка є одним із ключових інструментів сервісу, дозволяє легко орієнтуватися у розташуванні житла, переглядати дані про забудовників, цінові пропозиції та строки здачі об’єктів.

Платформа також робить акцент на безпеці і надійності, тісно співпрацюючи з юридичними фірмами для перевірки дозвільної документації та фінансування будівництва. Регулярне оновлення фотографій будівництва з однакових точок дає можливість користувачам оцінити реальний прогрес та стан об’єктів. При цьому, ЛУН.ua підтримує чистий інтерфейс без нав’язливої реклами, що підвищує зручність використання сайту [4].

Крім веб версії, сервіс пропонує мобільний додаток, що розширює можливості користувачів, дозволяючи шукати житло в будь-який час, зберігати вподобані об’єкти та отримувати сповіщення про нові пропозиції. У

2020 році компанія провела ребрендинг, виокремивши пошук вторинного житла у сервіс Flatfy, щоб зосередитися на новобудовах, у яких має найбільшу експертизу. Розширення можливостей стало можливим також завдяки придбанню порталу Rieltor.ua у 2022 році, що сприяло збільшенню бази оголошень та покращенню якості послуг.

Отже, ЛУН.ua є потужним і зручним інструментом для тих, хто шукає житло на первинному ринку України. Платформа постійно вдосконалюється, впроваджуючи сучасні технології та сервіси, щоб забезпечити максимально комфортний та безпечний процес пошуку нерухомості.

Розглянемо інше програмне рішення на ринку.

Zillow — одна з найбільших та найвідоміших онлайн-платформ нерухомості у Сполучених Штатах, розроблена для об’єднання покупців, продавців та орендарів з вичерпними списками нерухомості. Вона пропонує користувачам доступ до величезної бази даних будинків на продаж, в оренду та нещодавно проданих об’єктів, що підтримується розширеними фільтрами пошуку та ринковою аналітикою [5].

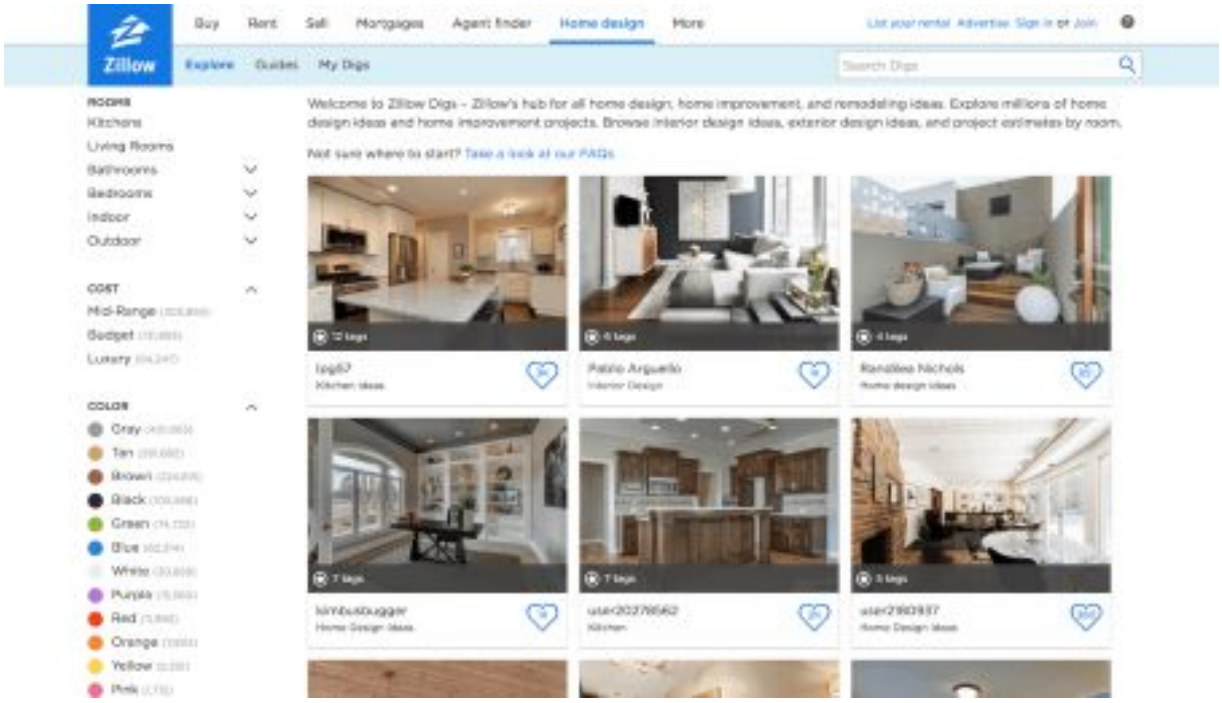

Рис. 1.2 Інформаційна система “Zillow”

Платформа надає детальну інформацію про нерухомість, включаючи фотографії, історію цін, дані про райони та оціночну вартість житла, відому як

«Zestimates». Зручний інтерфейс Zillow дозволяє покупцям налаштовувати пошук на основі таких критеріїв, як місцезнаходження, ціновий діапазон, кількість спалень тощо, що полегшує пошук будинків, що відповідають їхнім уподобанням.

Окрім списків нерухомості, Zillow сприяє прямому спілкуванню між покупцями та продавцями або агентами з нерухомості, покращуючи процес переговорів та укладання угод. Платформа також пропонує різні інструменти, такі як іпотечні калькулятори, оцінювачі доступності та довідники з вартості ремонту житла, допомагаючи користувачам приймати обґрунтовані фінансові рішення [5].

Технологічно Zillow використовує масштабовану архітектуру серверної частини, здатну щодня керувати мільйонами списків та взаємодій користувачів. Його вебсайт та мобільний додаток забезпечують безперебійний досвід, пропонуючи інтерактивні карти, збережені пошукові запити та персоналізовані рекомендації.

Незважаючи на свою сильну присутність, Zillow в основному зосереджується на ринку нерухомості США, що обмежує його використання на міжнародному рівні. Крім того, хоча «Zestimate» надає корисні оцінки, він не є офіційною оцінкою та іноді може відрізнятися від фактичної ринкової вартості.

Загалом, Zillow залишається піонером у сфері цифрових послуг нерухомості, поєднуючи великі дані, інноваційні інструменти та орієнтований на користувача підхід для спрощення процесу купівлі та продажу житла.

1.3 Постановка завдання

Основна функція системи полягає в тому, щоб дозволити користувачам ефективно керувати оголошеннями про нерухомість, пропозиціями та комунікаціями, а також взаємодіяти з ними. Для цього програмне забезпечення розроблено для ведення комплексної бази даних, що містить всю необхідну

інформацію про об'єкти нерухомості, користувачів, пропозиції та розмови. Це дозволяє користувачам отримувати своєчасні сповіщення та оновлення щодо нових оголошень, зроблених пропозицій або змін у статусі об'єктів нерухомості.

Основна програмна система розроблена для забезпечення користувачів швидкого та зручного доступу до:

- детальної інформації про об'єкти нерухомості, включаючи фотографії та описи;
- оновлень у режимі реального часу щодо пропозицій, поданих щодо об'єктів нерухомості [6];
- інтегрованого чату для безперебійного спілкування між покупцями та продавцями;
- інструментів управління для розміщення, редагування та видалення оголошень.

Програма налаштована для роботи через інтуїтивно зрозумілий інтерфейс з меню, що дозволяє користувачам виконувати всі дії в розмовній та керованій манері. Така конструкція гарантує, що користувачі можуть легко орієнтуватися в системі, відстежувати свою активність та, за необхідності, без труднощів скасовувати або змінювати свої оголошення та пропозиції.

1.4 Функціональні та нефункціональні вимоги

Функціональні вимоги:

1. користувачі повинні мати можливість створювати облікові записи, безпечно входити в систему та відновлювати паролі;
2. продавці можуть створювати, редагувати та видаляти оголошення про нерухомість, зокрема завантажувати кілька фотографій та вказувати детальну інформацію про нерухомість (розташування, ціна, розмір, тип, опис тощо);

3. покупці можуть шукати оголошення, використовуючи різні фільтри, такі як ціновий діапазон, розташування, тип нерухомості, кількість кімнат та інші відповідні критерії;
4. покупці можуть подавати пропозиції щодо купівлі нерухомості, які потім надсилаються продавцю для розгляду;
5. система повинна підтримувати функцію чату, що дозволяє здійснювати пряме спілкування між покупцями та продавцями для обговорення пропозицій та узгодження умов;
6. користувачі отримують сповіщення про нові повідомлення, пропозиції, зміни статусу оголошень або відповіді на свої пропозиції;
7. користувачі можуть оновлювати свою особисту інформацію, переглядати свої активні оголошення або пропозиції та керувати налаштуваннями облікового запису;
8. адміністратори можуть відстежувати активність користувачів, модерувати оголошення для запобігання шахрайству або неприйнятному контенту, а також керувати налаштуваннями системи.

Нефункціональні вимоги:

1. система повинна забезпечувати швидкий час реагування на пошук, завантаження оголошень та повідомлення в чаті, навіть за високого навантаження на користувачів;
2. платформа повинна обробляти зростаючу кількість користувачів та оголошень без зниження продуктивності;
3. конфіденційні дані, такі як паролі та особиста інформація, повинні зберігатися безпечно за допомогою шифрування. Система повинна захищати від поширених веб-вразливостей (наприклад, SQL-ін'єкції, XSS) [7];
4. інтерфейс користувача має бути інтуїтивно зрозумілим та доступним на різних пристроях, включаючи настільні

комп'ютери та мобільні телефони. Додаток має бути стабільним та доступним з мінімальним часом простою, забезпечуючи безперервне обслуговування для користувачів;

5. кодова база має бути добре структурованою та задокументованою, щоб майбутні розробники могли ефективно оновлювати та розширювати систему;
6. забезпечте, щоб усі введені та збережені дані залишалися узгодженими та точними в усій системі;
7. програмне забезпечення повинно відповідати відповідним правилам та стандартам захисту даних, що застосовуються до онлайн-платформ, що обробляють інформацію користувачів.

1.5 Вимоги до інтерфейсу користувача

Інтерфейс користувача онлайн-платформи нерухомості розроблений для забезпечення безперебійного та інтуїтивно зрозумілого досвіду для всіх користувачів, включаючи покупців, продавців та адміністраторів. Пріоритетом є простота та зрозумілість, що забезпечує легку навігацію та дозволяє користувачам легко виконувати ключові дії, такі як пошук нерухомості, розміщення оголошень та керування пропозиціями, без зайвих ускладнень. Чистий макет мінімізує відволікаючі фактори та природно спрямовує користувачів через систему.

Щоб забезпечити доступ з різних пристроїв, інтерфейс повністю адаптивний, автоматично підлаштовується під різні розміри екранів, включаючи настільні комп'ютери, планшети та смартфони. Ця гнучкість гарантує, що користувачі можуть зручно взаємодіяти з сервісом будь-коли та будь-де. Узгодженість підтримується на всіх сторінках завдяки єдиним елементам дизайну, таким як шрифти, кольори та розміщення компонентів інтерфейсу, що сприяє знайомому та комфортному середовищу. Меню навігації чітко відображаються та логічно структуровані, що дозволяє швидко

та легко переміщатися між розділами, такими як оголошення про нерухомість, профілі користувачів, повідомлення та налаштування.

Функція пошуку є надійною, але зручною для користувача, пропонує фільтри за ціною, місцезнаходженням, типом нерухомості, кількістю кімнат та іншими відповідними критеріями. Результати пошуку представлені чітко, що дозволяє користувачам швидко зрозуміти ключові деталі про кожен об'єкт. Самі оголошення про нерухомість демонструють високоякісні зображення, що супроводжуються вичерпними описами та важливою інформацією, включаючи ціну, розмір та місцезнаходження. Галереї зображень є інтерактивними та легкими для перегляду, що дає користувачам краще уявлення про кожну нерухомість [8].

Спілкування між покупцями та продавцями здійснюється завдяки доступній функції чату, яка підтримує обмін повідомленнями в режимі реального часу. Сповіщення про нові повідомлення розроблені таким чином, щоб бути помітними, не порушуючи робочий процес користувача. Профілі користувачів прості в навігації та дозволяють легко оновлювати особисті дані, активні оголошення та налаштування облікового запису, водночас гарантуючи безпечне поводження з конфіденційною інформацією.

Міркування щодо доступності є невід'ємною частиною дизайну, забезпечуючи підтримку користувачів з інвалідністю за допомогою таких функцій, як сумісність з програмою зчитування з екрана, навігація за допомогою клавіатури та відповідні рівні контрастності. Зворотній зв'язок надається негайно після дій користувача, підтверджуючи успішні операції або надаючи чіткі пояснення помилок. Сповіщення розроблені так, щоб бути інформативними, але ненав'язливими, підтримуючи збалансований користувацький досвід.

2 МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Загальні відомості

Уніфікована мова моделювання (UML) – це широко прийнятий стандарт для візуалізації, специфікації, побудови та документування компонентів програмних систем. Вона пропонує універсальний набір графічних нотацій, що представляють різні аспекти архітектури та поведінки системи. Забезпечуючи спільну мову, UML допомагає подолати прогалини в комунікації між розробниками програмного забезпечення, аналітиками та зацікавленими сторонами, забезпечуючи спільне розуміння структури та динаміки проєкту [9].

Однією з ключових переваг UML є її здатність представляти як статичні, так і динамічні аспекти програмного забезпечення. Статичні діаграми, такі як діаграми класів, ілюструють сутності системи, їх атрибути та зв'язки між ними. Динамічні діаграми, включаючи діаграми послідовностей та дій, зображують взаємодії та робочі процеси, що відбуваються під час роботи системи. Це поєднання дозволяє комплексно моделювати складні системи з різних точок зору.

У контексті онлайн-сервісу нерухомості UML відіграє вирішальну роль у відображенні основних бізнес-процесів та структур даних. Наприклад, діаграми варіантів використання допомагають визначити взаємодію між користувачами (покупцями, продавцями, адміністраторами) та функціональними можливостями системи, такими як розміщення оголошень про нерухомість, пошук оголошень та ведення переговорів щодо пропозицій. Діаграми класів моделюють основні об'єкти — властивості, користувачів, пропозиції — та їхні атрибути й асоціації, забезпечуючи план для проєктування баз даних та логіки застосунків.

Крім того, UML сприяє ранньому виявленню недоліків проєктування, дозволяючи розробникам моделювати поведінку системи та візуально

перевіряти вимоги. Це підвищує якість документації, полегшуючи її подальше обслуговування та розширення системи. Стандартизована нотація UML також підтримує інтеграцію з різними інструментами розробки, підвищуючи продуктивність та співпрацю між міждисциплінарними командами [10].

Загалом, використання UML у процесі розробки програмного забезпечення для онлайн-платформи нерухомості забезпечує чіткість, узгодженість та ефективність. Це забезпечує структурований підхід до перетворення абстрактних ідей у добре організований та функціональний застосунок, узгоджуючи технічну реалізацію з потребами користувачів та бізнес-цілями.

2.2 Об’єктне та функціональне моделювання

2.2.1 Діаграма прецедентів. Діаграми варіантів використання є фундаментальною частиною Уніфікованої мови моделювання (UML), яка використовується для візуального представлення взаємодії між користувачами (акторами) та системою. Вони слугують загальним планом, який окреслює функціональні вимоги до програмного застосунку, ілюструючи, що система повинна робити з точки зору користувача. На відміну від детальних схем проєктування, діаграми варіантів використання зосереджені на фіксації цілей та завдань, яких користувачі хочуть досягти за допомогою системи [11].

В онлайн-сервісі нерухомості діаграми варіантів використання допомагають визначити та впорядкувати різні ролі, такі як покупці, продавці та адміністратори. Кожен актор взаємодіє з системою через різні варіанти використання, які представляють ключові функціональні можливості, наприклад, розміщення оголошень про нерухомість, пошук оголошень, подання пропозицій про купівлю або керування обліковими

записами користувачів. Така візуалізація полегшує розуміння обсягу системи та того, як кожен тип користувача бере участь у бізнес-процесах.

Чітко визначаючи варіанти використання та акторів, ці діаграми сприяють комунікації між зацікавленими сторонами, розробниками та аналітиками. Вони забезпечують простий спосіб підтвердження вимог та забезпечення відповідності функціональності системи потребам користувачів. Крім того, діаграми варіантів використання підтримують процес розробки, слугуючи основою для створення більш детальних моделей, таких як діаграми послідовностей або діаграми діяльності, які додатково уточнюють, як відбуваються взаємодії.

Діаграми варіантів використання особливо цінні на ранніх етапах життєвого циклу проєкту, оскільки вони допомагають визначити пріоритети функцій та виявити відсутні вимоги. Вони також сприяють орієнтованому на користувача дизайну, зосереджуючись на тому, чого користувачі хочуть досягти, а не на технічних деталях реалізації. Загалом, діаграми варіантів використання сприяють формуванню чіткого, спільного розуміння цілей та меж системи, спрямовуючи розробку до створення ефективних рішень.

Розроблена діаграма прецедентів використання представлена на рис. 2.1.

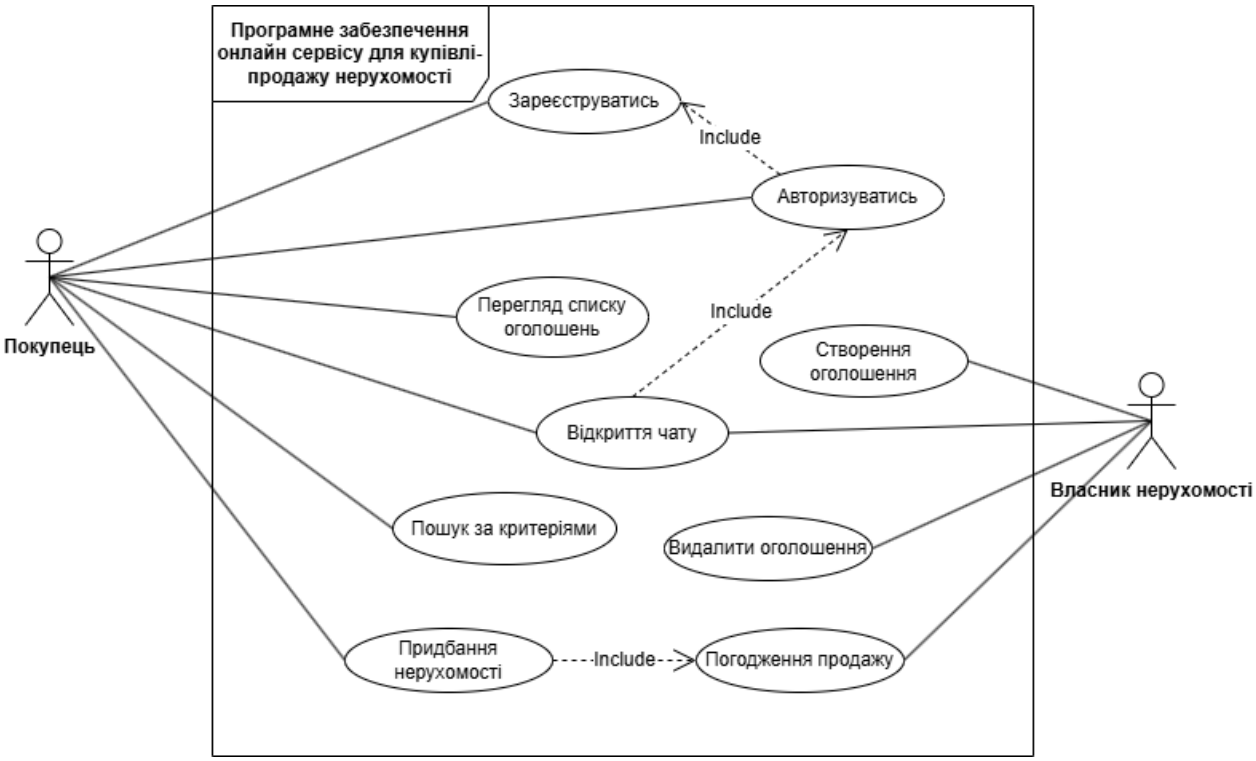

Рис. 2.1 Діаграма прецедентів

Створена діаграма прецедентів містить акторів:

- “Покупець”;
- “Власник нерухомості”.

Актор «Покупець» включає такі прецеденти:

- зареєструватись;
- авторизуватись;
- перегляд списку оголошень;
- відкриття чату;
- пошук за критеріями;
- придбання нерухомості.

Актор «Власник нерухомості» включає такі прецеденти:

- створення оголошення;
- видалити оголошення;
- погодження продажу.

Прецеденти певним чином залежать одне від одного.

Розглянемо детальніше вищеописані прецеденти.

Назва прецеденту: Огляд списку оголошень

Актор: Покупець

Опис: Цей варіант використання описує процес, за допомогою якого покупець отримує доступ та переглядає доступні оголошення про нерухомість на онлайн-платформі. Покупець шукає, фільтрує та переглядає описи оголошень про нерухомість, щоб знайти варіанти, що відповідають його критеріям.

Основний потік:

1. покупець входить до онлайн-сервісу нерухомості або отримує доступ до нього як гість;
2. покупець переходить до розділу, що відображає описи нерухомості;
3. система надає список доступних оголошень про нерухомість з короткою інформацією, такою як назва, ціна, місцезнаходження та мініатюрне зображення;
4. покупець застосовує фільтри або критерії пошуку, щоб звузити список оголошень (наприклад, місцезнаходження, ціновий діапазон, тип нерухомості);
5. система оновлює список на основі вибраних фільтрів та пошукових термінів;
6. покупець прокручує відфільтрований список, переглядаючи описову інформацію кожного оголошення;
7. покупець вибирає конкретне оголошення, щоб переглянути детальну інформацію, включаючи повний опис, фотогалерею та контактні дані продавця.

Альтернативні потоки:

1. якщо покупець не ввійшов у систему, йому може бути запропоновано зареєструватися або продовжити як гість з обмеженою функціональністю;

2. якщо жодне оголошення не відповідає критеріям фільтра покупця, система відображає повідомлення про те, що результатів не знайдено, і пропонує скинути фільтри або розширити пошук;
3. якщо покупець вирішує очистити фільтри, система повертається до відображення повного списку доступних оголошень.

Випадок використання завершується, коли покупець успішно переглянув одне або кілька оголошень про нерухомість і вирішує зробити пропозицію, зберегти оголошення на потім або вийти з платформи. Цей процес підтримує прийняття рішень покупцем, надаючи легкий доступ до відповідних варіантів нерухомості.

Розглянемо ще один сценарій використання.

Назва прецеденту: Погодження продажу

Актор: Власник нерухомості

Опис: Цей варіант використання описує процес, за допомогою якого власник нерухомості завершує продаж своєї нерухомості після отримання та перегляду пропозицій про купівлю від потенційних покупців. Він включає перегляд пропозицій, спілкування з покупцями та підтвердження продажу.

Основний потік:

1. власник входить на онлайн-платформу нерухомості та переходить до своїх об'єктів нерухомості;
2. система відображає список активних оголошень про нерухомість разом з будь-якими отриманими пропозиціями від зацікавлених покупців;
3. власник вибирає конкретну нерухомість та переглядає детальні пропозиції, включаючи інформацію про покупця та запропоновані умови;
4. власник ініціює зв'язок з одним або кількома покупцями через систему обміну повідомленнями платформи, щоб обговорити умови продажу;

5. після переговорів власник погоджується з умовами покупця;
6. власник підтверджує прийняття пропозиції через платформу, яка оновлює статус нерухомості на «Очікується продаж» або «Продано»;
7. система повідомляє покупця про рішення власника та надає необхідну інформацію для продовження процесу продажу;
8. власник за бажанням оновлює оголошення про нерухомість або видаляє його з платформи після завершення продажу офлайн.

Альтернативні потоки:

1. якщо власник не отримує пропозицій, нерухомість залишається в оголошенні, і власник може оновити оголошення, щоб покращити привабливість або скоригувати ціну;
2. якщо переговори з покупцем не вдаються, власник може відхилити пропозицію та продовжити перегляд інших пропозицій або чекати на нові;
3. якщо власник вирішить зняти нерухомість з продажу, він може деактивувати або видалити оголошення в будь-який момент до завершення продажу.

Випадок використання завершується, коли власник успішно підтверджує продаж своєї нерухомості через платформу, фактично змінюючи статус оголошення та повідомляючи залучені сторони. Цей сценарій гарантує, що власник зберігає контроль над процесом продажу та спілкуванням з покупцями до завершення угоди.

2.2.2 Діаграма послідовності. Діаграми послідовностей – це важливий тип UML-діаграм, що використовуються для моделювання динамічної поведінки системи шляхом ілюстрації взаємодії об'єктів з часом. Вони зосереджені на послідовності повідомлень, якими обмінюються системні компоненти або актори для виконання певної функції або процесу.

Це робить їх особливо корисними для розуміння та документування детального потоку операцій у програмному застосунку [12].

Діаграма послідовності візуально розташовує об'єкти або актори горизонтально зверху, а вертикальні лінії життя простягаються вниз, щоб представити їх існування з часом. Повідомлення про взаємодію відображаються у вигляді стрілок між лініями життя, що вказують на виклики методів, передачу даних або відповіді. Такий хронологічний вигляд допомагає розробникам та аналітикам відстежувати, як розгортається певний сценарій крок за кроком.

У контексті онлайн-сервісу нерухомості діаграми послідовностей можуть зображувати важливі взаємодії, такі як пошук покупцем оголошень, подання пропозиції або спілкування з власником нерухомості. Наприклад, діаграма може показувати, як запит покупця запускає пошуковий запит, як система отримує відповідні оголошення з бази даних і як результати надсилаються назад до інтерфейсу покупця.

Використання діаграм послідовностей допомагає визначити ролі різних об'єктів у процесах, зрозуміти часові рамки та залежності системи, а також уточнити вимоги до впровадження системи. Вони також допомагають виявляти потенційні проблеми в логіці робочого процесу на ранніх етапах циклу розробки.

Загалом, діаграми послідовностей надають детальне, зосереджене на часі уявлення про те, як система поводиться під час конкретних випадків використання, доповнюючи інші діаграми UML, що моделюють статичну структуру або загальну архітектуру системи.

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

- “Покупець нерухомості”;
- “Оголошення”;
- “Пропозиція”;
- “Чат”.

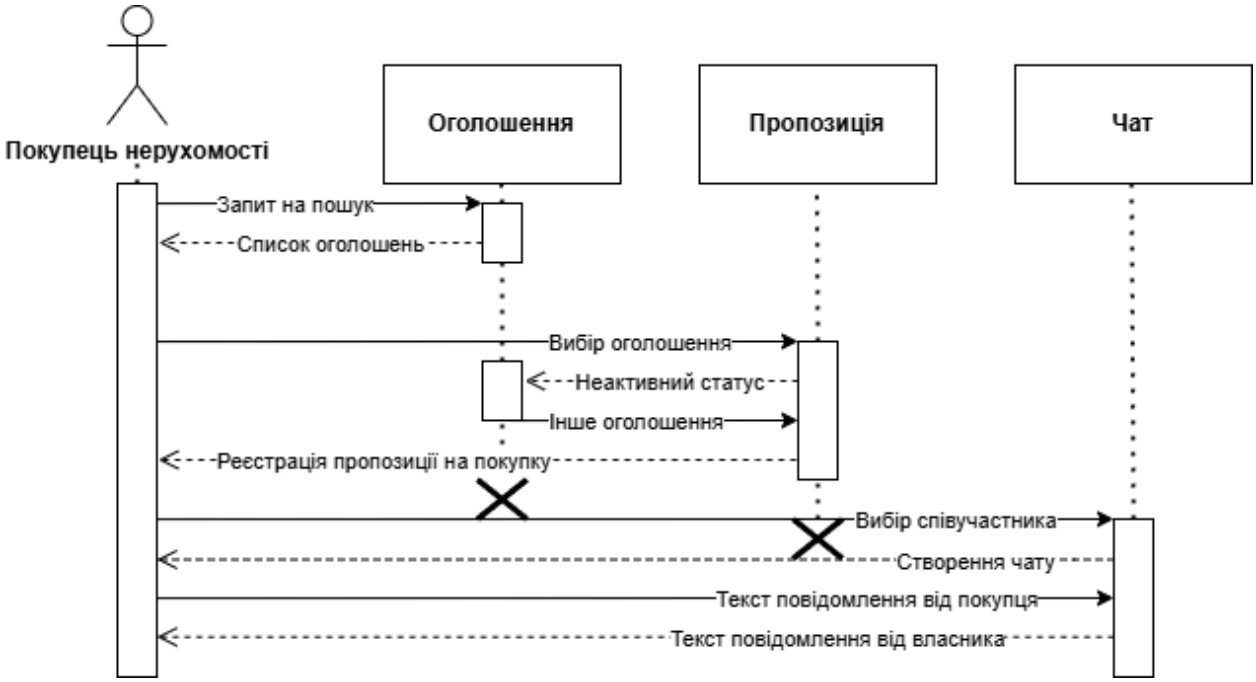

Рис. 2.2 Діаграма послідовності

Коли покупець шукає нерухомість, система надає список доступних оголошень. Потім покупець вибирає те, що його цікавить, але якщо вибране оголошення неактивне, він повинен обрати інше. Після того, як знайдено підходящий варіант, покупець надсилає пропозицію про купівлю та за потреби обирає спів покупця. На цьому етапі створюється чат, що дозволяє здійснювати пряме спілкування. Покупець надсилає повідомлення, а власник нерухомості відповідає, сприяючи подальшому обговоренню. Цей структурований процес забезпечує безперебійну взаємодію на ринку нерухомості, крок за кроком ведучи користувачів від перегляду до переговорів.

2.2.3 Діаграма активності. Діаграми діяльності – це тип UML-діаграм, призначений для моделювання потоку керування або дій у системі. Вони візуально представляють послідовність та умови координації поведінки нижчого рівня, показуючи, як різні дії прогресують та розгалужуються на основі рішень, паралельних процесів або циклів. Це

робить діаграми діяльності особливо корисними для опису робочих процесів, бізнес-процесів та складних процедур [13].

Діаграма діяльності зазвичай складається з дій (або дій), точок прийняття рішень, початкових та кінцевих вузлів та потоків керування, що з'єднують ці елементи. Вона допомагає проілюструвати, як процес розгортається від початку до кінця, включаючи всі можливі шляхи та альтернативні сценарії. На відміну від діаграм послідовності, які зосереджені на взаємодії об'єктів з часом, діаграми діяльності підкреслюють загальний робочий процес та порядок операцій.

У контексті онлайн-сервісу нерухомості діаграми діяльності можуть використовуватися для відображення таких процесів, як розміщення оголошення про нерухомість, перегляд пропозицій або завершення продажу. Наприклад, діаграма діяльності може детально описувати кроки, які продавець виконує для створення оголошення, завантаження фотографій, встановлення цін та публікації оголошення, виділяючи моменти прийняття рішень, такі як додавання додаткової інформації чи перехід до підтвердження.

Завдяки чіткій візуалізації робочих процесів, діаграми діяльності покращують комунікацію між розробниками, аналітиками та зацікавленими сторонами. Вони допомагають виявляти неефективність, надлишки або потенційні вузькі місця в процесах, сприяючи кращому проєкту системи та покращенню взаємодії з користувачем. Крім того, ці діаграми слугують корисним орієнтиром під час впровадження та тестування, щоб забезпечити правильне дотримання бізнес-логіки.

Підсумовуючи, діаграми діяльності забезпечують комплексне, орієнтоване на потоки уявлення про поведінку та процеси системи, що робить їх цінним інструментом для аналізу та проєктування функціональних аспектів програмних застосунків.

Розроблена діаграма активності представлена на рис. 2.3

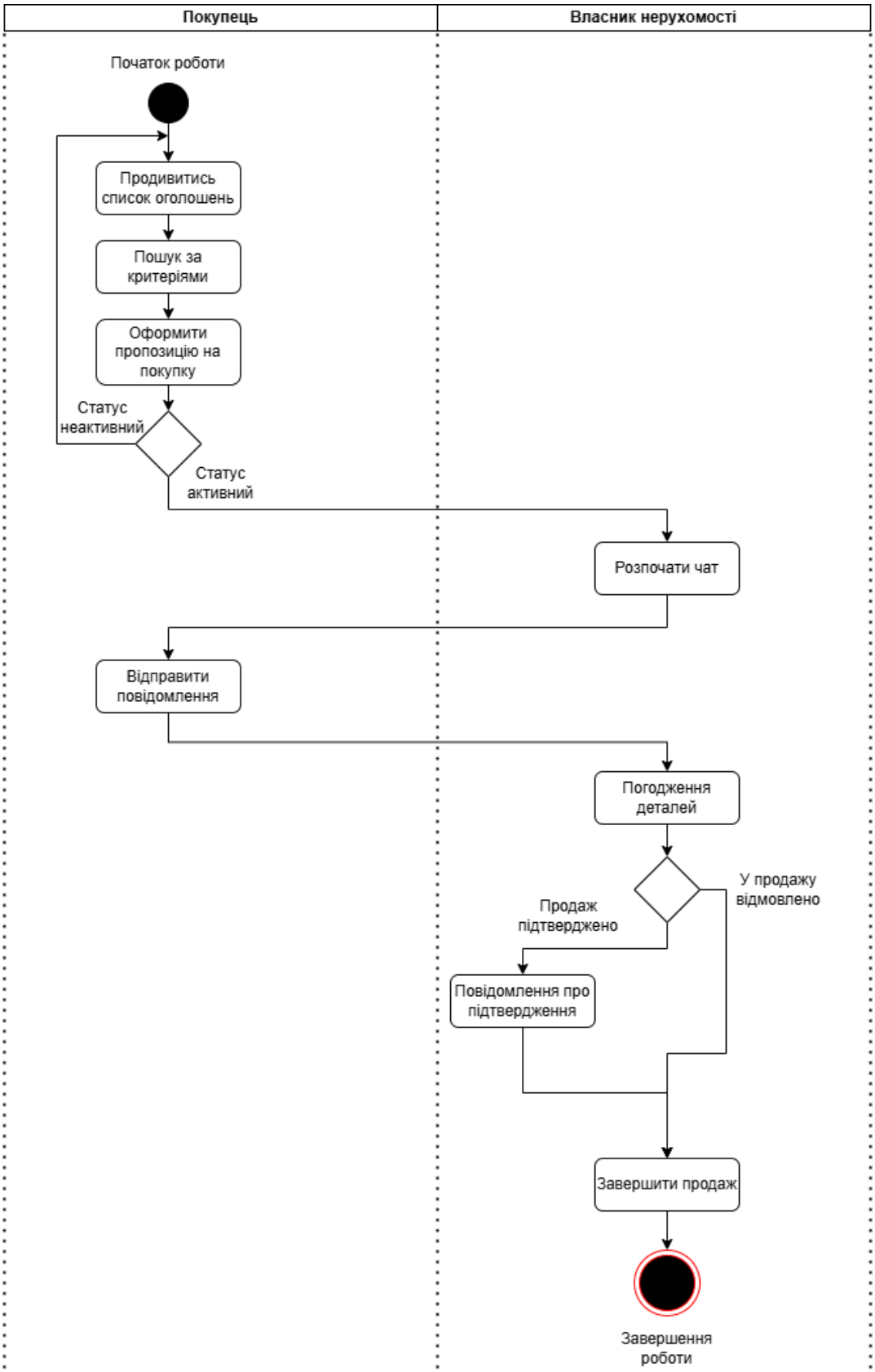

Рис. 2.3 Діаграма активності

Ця діаграма активності ілюструє процес взаємодії покупця нерухомості із системою, охоплюючи основні етапи покупки через онлайн-платформу.

Спочатку покупець здійснює пошук, отримуючи список доступних оголошень. Після вибору потрібного варіанту може виявитися, що оголошення має неактивний статус, що вимагає пошуку альтернативного варіанту. Якщо оголошення підходить, покупець реєструє пропозицію на покупку, а потім обирає співучасника для продовження процесу. На цьому етапі створюється чат, де покупець і власник нерухомості обмінюються повідомленнями, що дозволяє вести переговори та уточнювати деталі. Ця схема наочно демонструє логіку взаємодії між користувачами та платформою, забезпечуючи структурований підхід до купівлі нерухомості онлайн.

2.3 Абстракції предметної області

Абстракції предметної області включають визначення та спрощення ключових концепцій, сутностей та зв'язків, що визначають область, в якій працює програмна система. Цей процес допомагає створити концептуальну модель, яка відображає основні характеристики реального середовища без зайвої складності. Зосереджуючись на абстракціях, розробники та аналітики можуть краще зрозуміти обсяг та межі системи та забезпечити відповідність дизайну фактичним потребам бізнесу [14].

У контексті онлайн-сервісу купівлі-продажу нерухомості абстракції допомагають виділити основні елементи, такі як власність, користувачі (покупці та продавці), пропозиції та канали зв'язку. Замість моделювання кожної дрібної деталі, абстракція зосереджується на узагальнених сутностях, таких як «Оголошення про продаж нерухомості», що представляють будь-який тип власності, або «Користувач», що охоплює різні ролі, що взаємодіють із системою.

Створення цих абстракцій підтримує чіткішу комунікацію між зацікавленими сторонами та забезпечує міцну основу для проєктування архітектури системи та моделей даних. Це дозволяє команді зосередитися на тому, що дійсно важливо для функціональності системи, такому як управління

оголошеннями, обробка пропозицій та сприяння переговорам за допомогою обміну повідомленнями.

Крім того, абстракції забезпечують гнучкість та масштабованість у проєктуванні програмного забезпечення. Працюючи зі спрощеними представленнями, майбутні розширення або модифікації легше інтегрувати, не порушуючи роботу всієї системи. Такий підхід також допомагає зменшити складність під час етапів впровадження та обслуговування.

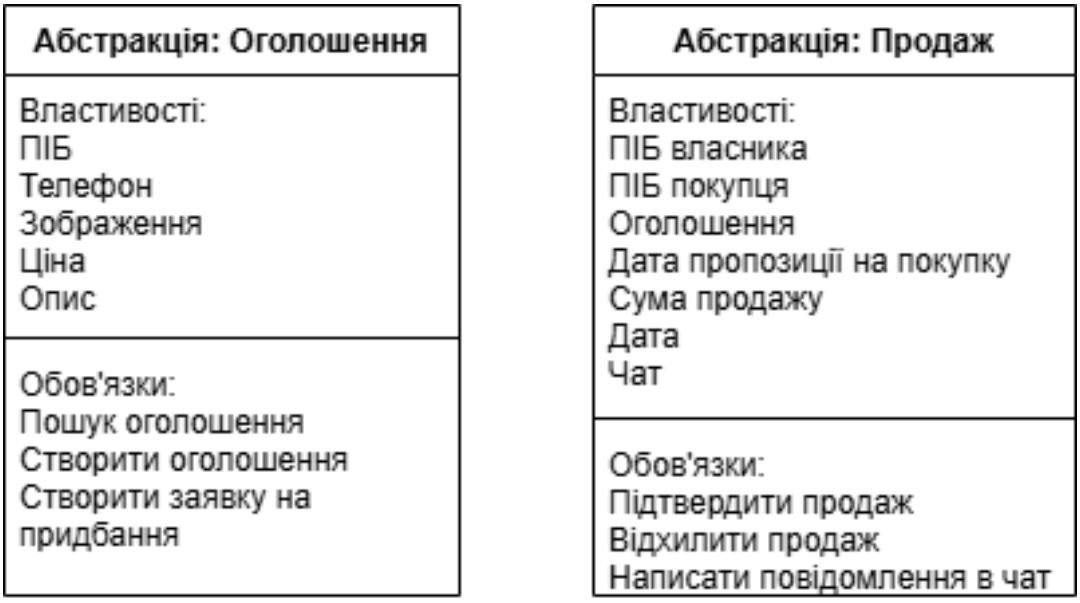

Рис. 2.4 Абстракції предметної області

У предметній області нерухомості основні абстракції включають ключові поняття та об’єкти, що визначають процеси та взаємодії користувачів із системою.

Покупець нерухомості — це суб’єкт, який шукає відповідні пропозиції для придбання житла або комерційної нерухомості. Він взаємодіє із системою оголошень, де представлена інформація про доступні об'єкти. Кожне оголошення містить опис нерухомості, її статус (активний чи неактивний) та умови покупки. Якщо оголошення підходить, покупець оформляє пропозицію, яка фіксує його намір купівлі. У деяких випадках покупець може залучити співучасника, з яким ділить право власності на об'єкт.

На фінальному етапі створюється чат, що дає можливість учасникам угоди обговорити деталі, поставити запитання та домовитися про остаточні

умови. Власник нерухомості також є важливим суб’єктом, адже саме він відповідає на пропозицію та веде переговори. Ця система забезпечує структуровану, логічну взаємодію між усіма учасниками процесу та спрощує угоду.

Підсумовуючи, абстракції предметної області розбивають ринок нерухомості та взаємодію з користувачами на керовані, змістовні компоненти, спрямовуючи розробку надійного, орієнтованого на користувача програмного рішення.

2.4 Діаграма класів

Діаграми класів – це фундаментальний тип UML-діаграм, що використовується для представлення статичної структури програмної системи. Вони ілюструють класи системи, їх атрибути, методи та зв'язки між цими класами. Це візуальне представлення допомагає зрозуміти, як організовані дані та як різні частини системи взаємодіють на структурному рівні [15].

Діаграма класів зазвичай включає класи, зображені у вигляді прямокутників, розділених на секції для назви класу, атрибутів (властивостей) та операцій (методів). Такі зв'язки, як асоціації, успадкування, агрегації та композиції, показують, як класи пов'язані або залежні один від одного.

У контексті онлайн-сервісу купівлі-продажу нерухомості діаграми класів можуть моделювати такі сутності, як Користувач, Оголошення про нерухомість, Пропозиція та Повідомлення. Наприклад, клас Користувач може мати такі атрибути, як ім'я, електронна пошта та роль, тоді як клас Оголошення про нерухомість міститиме такі деталі, як адреса, ціна та фотографії. Асоціації можуть показувати, що Користувач може створювати кілька Оголошень про нерухомість, і кожне Оголошення може мати багато Пропозицій від різних покупців.

Діаграми класів служать кресленнями як для розробників, так і для зацікавлених сторін, надаючи чіткий огляд структури даних системи. Вони допомагають у проєктуванні баз даних, написанні класів в об'єктно-орієнтованому програмуванні та забезпеченні узгодженості протягом усього життєвого циклу системи. Крім того, вони допомагають виявляти потенційні проблеми проєктування на ранній стадії, такі як надлишкові дані або нечіткі зв'язки.

Загалом, діаграми класів є безцінними для візуалізації організації програмної системи, керівництва її розробкою та полегшення комунікації між членами команди.

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

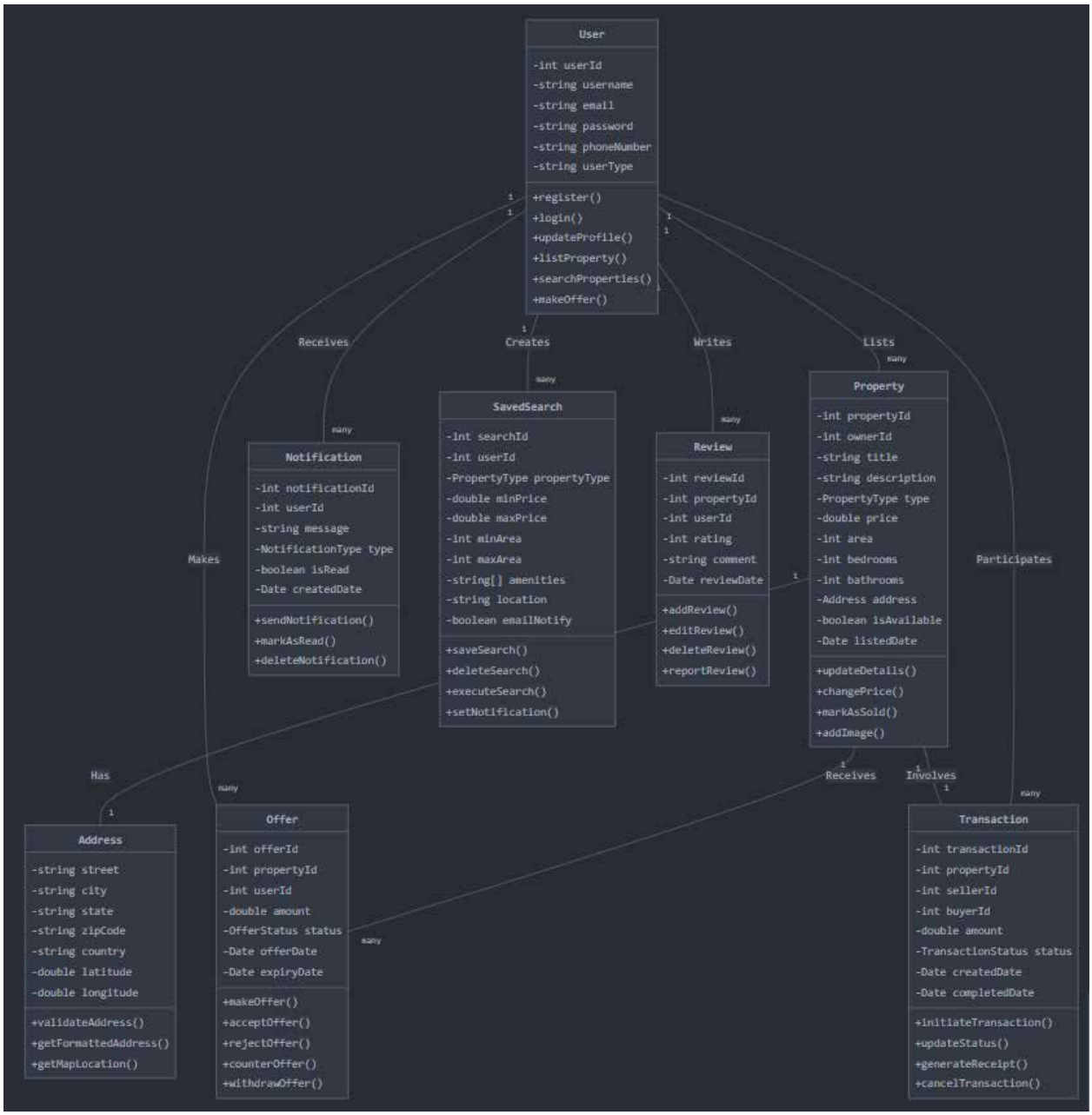

Рис. 2.5 Діаграма класів

Запропонована архітектура онлайн-ринку нерухомості зосереджена навколо восьми взаємопов'язаних класів, які утворюють комплексну екосистему для транзакцій з нерухомістю.

В основі системи лежить клас User (Користувач), який слугує основою для всіх взаємодій платформи. Цей клас керує індивідуальними профілями, процесами автентифікації та розрізняє різні типи користувачів, такі як покупці, продавці та агенти. Користувачі можуть виконувати важливі дії, такі

як реєстрація облікових записів, оновлення особистої інформації та навігація по списках нерухомості.

Клас Property (Нерухомість) охоплює всі аспекти списків нерухомості, доступних на платформі. Кожен екземпляр об'єкта нерухомості містить важливі деталі, включаючи розміри, конфігурацію кімнат, ціни та статус доступності. Об'єкти нерухомості нерозривно пов'язані з їхніми власниками (продавцями) та ведуть облік того, коли вони вперше були виставлені на ринок.

Для точного географічного представлення кожен об'єкт нерухомості пов'язаний з класом Address (Адреса). Цей спеціалізований компонент обробляє дані, що стосуються місцезнаходження, такі як інформація про вулицю, поштові індекси та географічні координати. Клас Address (Адреса) забезпечує життєво важливу функціональність для перевірки місцезнаходження та картографічних послуг, що покращує взаємодію з користувачем.

Ринок спрощує транзакції через два взаємодоповнюючі класи. Клас Offer керує етапом переговорів, дозволяючи потенційним покупцям висловлювати зацікавленість, надсилаючи фінансові пропозиції щодо нерухомості. Пропозиції існують у різних станах протягом свого життєвого циклу, включаючи статус «очікує розгляду», «прийнято», «відхилено» або «відхилено». Тим часом клас Transaction керує фактичною передачею права власності на нерухомість після прийняття пропозиції, відстежуючи процес від початку до завершення.

Щоб залучати та інформувати користувачів, клас Notification своєчасно надсилає сповіщення про відповідні дії на платформі. Ці сповіщення можуть стосуватися оновлень пропозицій, прогресу транзакції або нових об'єктів нерухомості, що відповідають збереженим критеріям. Ця система зв'язку гарантує, що всі сторони залишаються в курсі подій протягом усього процесу придбання нерухомості.

Клас SavedSearch підвищує зручність користувачів, зберігаючи персоналізовані налаштування нерухомості. Ця функція дозволяє користувачам зберігати певні параметри пошуку, такі як цінові діапазони, уподобання щодо місцезнаходження та бажані зручності. Потім система може проактивно сповіщати користувачів, коли нові оголошення відповідають їхнім встановленим критеріям.

Завершуючи екосистему, клас Review надає механізм зворотного зв'язку, де користувачі можуть ділитися досвідом та оцінками нерухомості, яку вони відвідали або придбали. Цей соціальний компонент формує довіру спільноти та пропонує цінну інформацію для потенційних покупців, які орієнтуються на ринку.

Ці класи працюють разом, створюючи єдину платформу, яка підтримує весь процес купівлі-продажу нерухомості — від початкового пошуку об'єкта до остаточного завершення покупки. Взаємозв'язки між класами відображають природний потік операцій з нерухомістю, забезпечуючи інтуїтивно зрозумілу навігацію та комплексний функціонал для всіх користувачів.

3 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Логічна модель даних

Логічна модель даних представляє організацію даних у системі структурованим та абстрактним чином, зосереджуючись на логічних зв'язках та правилах без заглиблення в деталі фізичної реалізації. Вона визначає сутності, їхні атрибути та зв'язки між ними, встановлюючи план для проєктування бази даних, яка підтримує вимоги програми [16].

У контексті онлайн-сервісу для купівлі та продажу нерухомості логічна модель даних визначає основні сутності, такі як Користувачі, Власність, Пропозиції та Повідомлення. Кожна сутність включає відповідні атрибути — наприклад, сутність Власність може містити такі деталі, як місцезнаходження, ціна, опис та фотографії, тоді як сутність Користувач зберігає таку інформацію, як ім'я, контактні дані та роль (покупець чи продавець). Зв'язки між сутностями ілюструють, як користувачі взаємодіють з нерухомістю та пропозиціями, забезпечуючи такі функціональні можливості, як розміщення оголошень, подання пропозицій щодо купівлі та обмін повідомленнями.

Ця модель абстрагується від специфіки фізичного зберігання, такої як індексація або системи управління базами даних, зосереджуючись на тому, як елементи даних логічно пов'язані та забезпечують цілісність даних. Наприклад, вона визначає первинні ключі для унікальної ідентифікації записів та зовнішні ключі для забезпечення зв'язків, таких як зв'язування пропозиції як з Користувачем, так і з Власністю.

Створення логічної моделі даних є вирішальним кроком у розробці системи, оскільки воно забезпечує узгоджену та послідовну структуру даних, узгоджену з бізнес-процесами. Воно полегшує комунікацію між розробниками, дизайнерами баз даних та зацікавленими сторонами, надаючи чітку карту потреб та обмежень системи в даних, перш ніж переходити до проєктування фізичної бази даних.

Підсумовуючи, логічна модель даних закладає основу для ефективного управління даними на платформі нерухомості, допомагаючи організувати та пов'язати інформацію таким чином, щоб забезпечити безперебійну роботу системи та майбутню масштабованість.

Логічна модель системи представлена на рисунку 3.1.

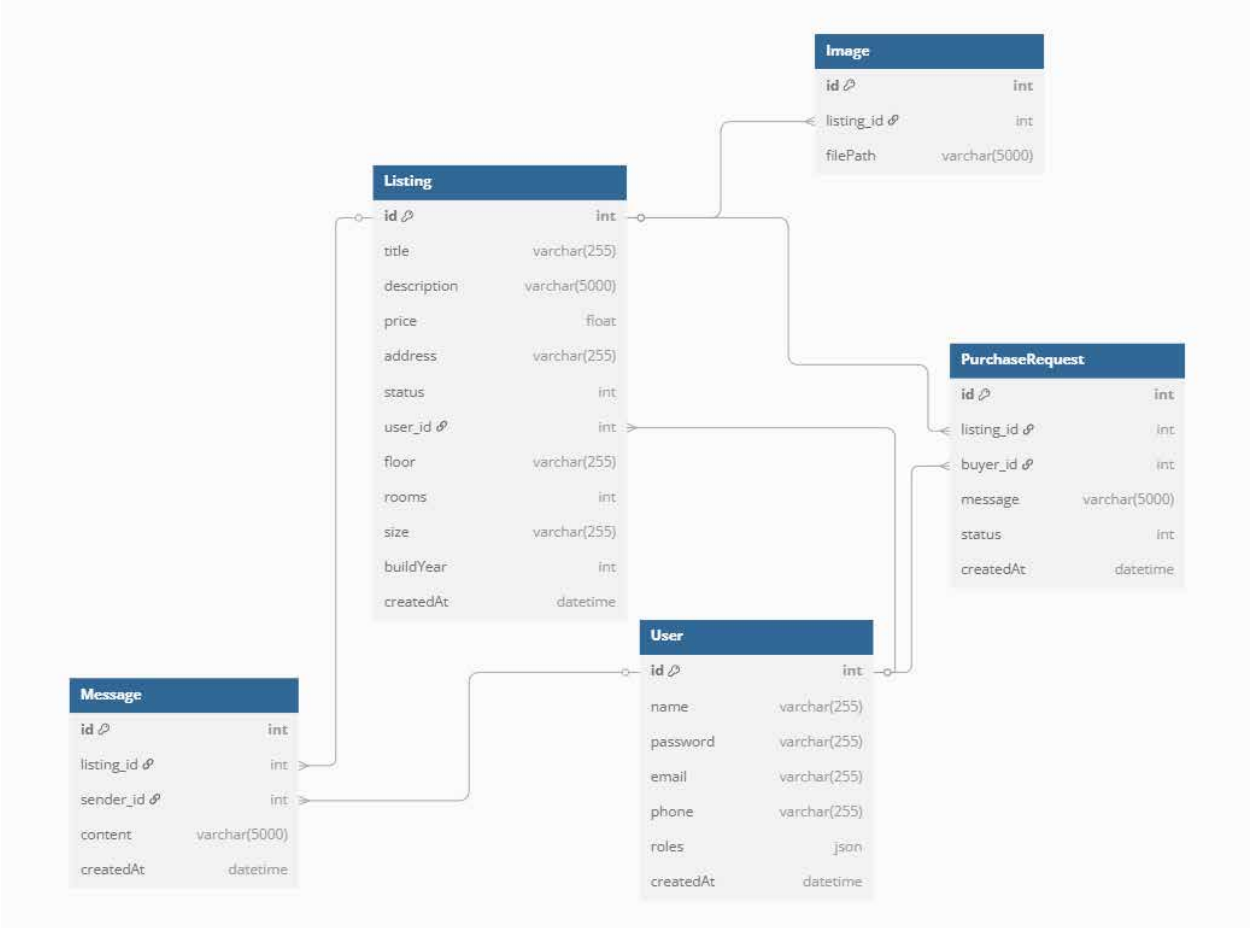

Рис. 3.1 ER-діаграма

Детальний опис діаграми бази даних:

User

Зберігає інформацію про користувачів платформи. Користувачі можуть бути продавцями (власниками оголошень), покупцями або і тими, й іншими.

1. id (int, первинний ключ, автоінкремент) — Унікальний ідентифікатор для кожного користувача;
2. name (string) — Повне ім'я користувача;
3. password (string) — Хешований пароль для автентифікації;
4. email (string) — Адреса електронної пошти;

5. phone (string) — Контактний номер телефону;
6. roles (json або масив) — Ролі або дозволи, які має користувач (наприклад, покупець, продавець, адміністратор);
7. createdAt (дата та час) — Дата та час реєстрації користувача.

Listing

1. id (int, PK, автоінкремент) — Унікальний ідентифікатор оголошенняж;
2. title (string) — Назва або заголовок оголошення (наприклад, "3-кімнатна квартира в центрі міста");
3. description (string, nullable) — Детальний опис нерухомості;
4. price (float) — Запитувана ціна нерухомості;
5. address (string, nullable) — Фізична адреса об'єкта нерухомості;
6. status (int) — Статус об'єкта нерухомості (наприклад, активний, проданий, орендований);
7. user_id (int, FK) — Посилається на продавця (користувача), який створив об'єкт нерухомості;
8. floor (string, nullable) — Номер поверху або опис;
9. rooms (int, nullable) — Кількість кімнат у об'єкті нерухомості;
10. size (string, nullable) — Розмір або площа об'єкта нерухомості (наприклад, у квадратних метрах);
11. buildYear (int, nullable) — Рік будівництва об'єкта нерухомості;
12. createdAt (datetime) — Дата/час створення об'єкта нерухомості.

Image

Містить зображення, пов'язані з об'єктами нерухомості, що дозволяє додавати кілька фотографій до кожного об'єкта.

1. id (int, PK, автоінкремент) — Ідентифікатор зображення;
2. listing_id (int, FK) — Посилається на об'єкт нерухомості, до якого належить зображення;
3. filePath (string, nullable) — Шлях або URL-адреса до файлу зображення.

PurchaseRequest

Запит на купівлю або оренду, подані користувачами (покупцями) для оголошень.

1. id (int, PK, auto-increment) — Унікальний ідентифікатор запиту на купівлю;
2. listing_id (int, FK) — Оголошення, до якого відноситься запит;
3. buyer_id (int, FK) — Користувач, який подав запит на купівлю/оренду;
4. message (string, nullable) — Додаткове повідомлення від покупця (наприклад, питання або пропозиції);
5. status (int) — Статус запиту (очікує на розгляд, схвалено, відхилено);
6. createdAt (дата/час) — Мітка часу, коли було зроблено запит.

Message

Зберігає повідомлення, якими обмінюються користувачі щодо певних оголошень (наприклад, переговори або питання).

1. id (int, PK, auto-increment) — Ідентифікатор повідомлення;
2. listing_id (int, FK) — Оголошення, пов'язане з повідомленням;
3. sender_id (int, FK) — Користувач, який надіслав повідомлення;
4. content (string) — Текстовий вміст повідомлення;
5. createdAt (дата й час) — Коли повідомлення було надіслано.

Структура бази даних розроблена для ефективної підтримки основної функціональності онлайн-платформи нерухомості, забезпечуючи безперебійну взаємодію між користувачами та оголошеннями. В основі системи лежать користувачі, які можуть виступати як продавці або покупці. Продавець може створювати кілька оголошень про нерухомість, причому кожне оголошення пов'язане виключно з одним користувачем. Ці оголошення представляють пропозиції нерухомості та слугують основою для подальшої взаємодії на платформі.

Кожне оголошення може бути збагачене кількома зображеннями, що забезпечують візуальне представлення та допомагають потенційним покупцям краще зрозуміти стан та характеристики нерухомості. Таке з'єднання дозволяє одному оголошенню мати кілька пов'язаних зображень, кожне з яких унікально пов'язане з цією конкретною пропозицією.

Потенційні покупці можуть висловити зацікавленість у нерухомості, подавши запити на купівлю або оренду. Кожен із цих запитів пов'язаний з певним оголошенням та ініціюється користувачем, який діє як покупець. Така схема гарантує, що кілька запитів можуть бути пов'язані з одним оголошенням, і кожен користувач має можливість подати кілька запитів на різні об'єкти нерухомості, залежно від своїх інтересів.

Для полегшення комунікації система включає компонент обміну повідомленнями. Користувачі можуть надсилати кілька повідомлень, кожне з яких пов'язане з певним оголошенням, що дозволяє безпосередньо обговорювати дану нерухомість. Це сприяє переговорам, роз'ясненням та взаємодії між покупцями та продавцями у структурований та організований спосіб.

Крім того, використання полів статусу в оголошеннях та запитах на купівлю дозволяє ефективно відстежувати прогрес, наприклад, чи нерухомість все ще доступна, чи запит очікує на розгляд, схвалений чи відхилений. Поля позначок часу, вбудовані в різні сутності, відіграють важливу роль у сортуванні записів у хронологічному порядку та веденні точних журналів для аудиту та історії активності користувачів.

Загалом, ця реляційна структура гарантує, що всі ключові взаємодії в угоді з нерухомістю — оголошення, висловлення зацікавленості, перегляд зображень та обмін повідомленнями — безперешкодно пов'язані, забезпечуючи міцну та масштабовану основу для онлайн-платформи, яка обробляє купівлю та продаж нерухомості.

3.2 Вибір системи управління базою даних та її реалізація

Під час розробки онлайн-платформи нерухомості вибір правильної системи керування базами даних є фундаментальним рішенням, яке безпосередньо впливає на продуктивність, масштабованість та зручність обслуговування системи. Програмне забезпечення повинно обробляти різні типи даних та підтримувати динамічну взаємодію між користувачами, оголошеннями та пов'язаними з ними об'єктами, одночасно забезпечуючи безпеку, швидкість та цілісність даних. Враховуючи ці потреби, MySQL виділяється як найбільш підходящий вибір. Це перевірена реляційна система баз даних, яка пропонує надійну продуктивність, розширену документацію та високу сумісність з фреймворками на основі PHP, такими як Symfony або Laravel, які зазвичай використовуються в розробці веб-додатків [16].

Структура бази даних продумано розроблена, щоб відображати типовий потік взаємодії користувачів на платформі торгівлі нерухомістю. В її основі лежить таблиця User (Користувач), яка містить важливу інформацію про осіб, які користуються платформою, незалежно від того, чи є вони покупцями, чи продавцями. Ці користувачі можуть створювати оголошення про нерухомість, надсилати запити та обмінюватися повідомленнями. Об'єкт Listing (Оголошення) зберігає дані, пов'язані з нерухомістю, включаючи ціну, опис, місцезнаходження та статус, і кожен оголошення пов'язаний з обліковим записом користувача. Для покращення презентації нерухомості окрема таблиця зображень поєднує кілька фотографій з кожним оголошенням, допомагаючи потенційним покупцям візуалізувати те, що пропонується.

Покупці взаємодіють з оголошеннями, надсилаючи запити на купівлю, які містять повідомлення, позначку часу та статус для відстеження прогресу запиту — від початкового інтересу до потенційного схвалення або відхилення. Ці запити пов'язані як з відповідним оголошенням, так і з користувачем, який їх надіслав. Для підтримки прямого зв'язку платформа також має таблицю повідомлень, що дозволяє користувачам обмінюватися інформацією,

пов'язаною з певним оголошенням. Кожне повідомлення записує його вміст, відправника та точний час його створення.

Схема бази даних використовує зовнішні ключі для збереження цілісності посилань між взаємопов'язаними таблицями, гарантуючи, що всі зв'язки даних залишаються узгодженими та логічними. Такі поля, як індикатори стану та позначки часу, включені для хронологічного сортування, фільтрації та адміністративного аудиту. Крім того, індексація часто шуканих полів допомагає покращити швидкість запитів та швидкість реагування системи, навіть коли платформа масштабується для розміщення більшої кількості користувачів та оголошень [17].

Під час розробки програмного забезпечення для онлайн-сервісу, спеціального купівлі та продажу нерухомості, вибору відповідної системи керування базами даних (СКБД) має вирішальне значення для забезпечення надійності, масштабованості та загального успіху платформи. Після оцінки кількох варіантів MySQL було обрано оптимальне рішення для цього проєкту, що виходить з низки практичних та технічних міркувань.

Перш за все, MySQL — це широко розширена реляційна СКБД з відкритим кодом, яка має давню репутацію стабільності та продуктивності. Її зрілість та велика база користувачів відзначають, що вона протестована в реальних сценаріях, включаючи веб-додатки з високим трафіком, що робить її надійною основою для зберігання та управління даними в системі нерухомості.

Ще одним фактором є безперешкодна інтеграція MySQL з фреймворками на основі PHP, такими як Symfony або Laravel, які фактично використовуються у веб-розробці. Ця сумісність спрощує процес розробки, зменшує проблеми з конфігурацією та дозволяє швидше впроваджувати такі функції, як автентифікація користувачів, списки нерухомості та сторінки, керовані даними.

Зі структурної точки зору MySQL підтримує складні зв'язки між сутностями, які є для цього типу додатків. Система повинна ефективно

керувати користувачами (як покупцями, так і продавцями), оголошеннями про нерухомість, галереями зображень, запитами на купівлю та повідомленнями — все це взаємопов'язано. Підтримка MySQL зовнішніх ключів, транзакцій та індексації дозволяє створити нормалізовану та добре структуровану схему, яка підтримує цілісність даних та ефективне запитування.

Крім того, MySQL надає надійні функції безпеки, такі як керування привілеями користувачів і зашифрованими з'єднаннями, які мають вірішне значення для захисту конфіденційних даних, включаючи особисту інформацію користувачів і деталі транзакцій. Особливо важливий рівень безпеки на платформах, що включає фінансові та юридичні аспекти, такі як операції з нерухомістю [18].

MySQL також перевершує продуктивність та масштабованість, обробляючи великий обсяг одночасних операцій читання та запису з мінімальною затримкою. Зі зростанням кількості користувачів та система оголошень повинна продовжувати працювати ефективно. Можливості індексації MySQL та підтримка оптимізованих запитів допомагають забезпечити чутливість платформи під завантаженням.

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

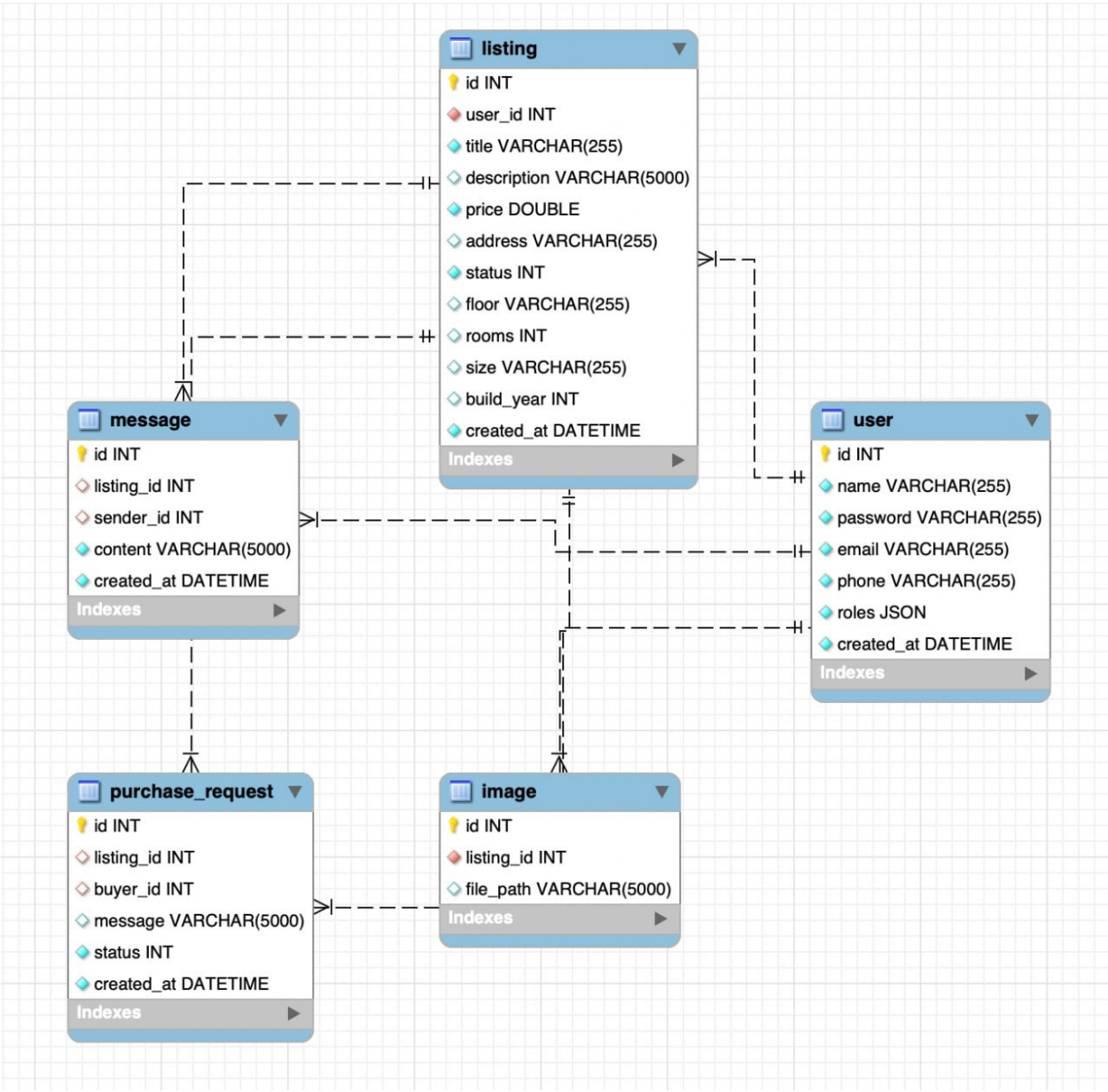

Рис. 3.2 База даних

Ця база даних моделює ключові аспекти системи купівлі нерухомості, забезпечуючи зберігання та управління інформацією про оголошення, пропозиції, користувачів і їхню взаємодію.

Основними таблицями є Користувачі, які містять дані про покупців і власників нерухомості, Оголошення, що включають параметри нерухомості, такі як тип, розташування, ціна та статус активності, та Пропозиції, де фіксуються умови потенційних угод, включаючи ціну та співучасників угоди. Окрема таблиця Чати забезпечує збереження повідомлень між покупцем і власником, що дозволяє вести переговори та уточнювати деталі угоди.

Зв’язки між таблицями реалізовані через унікальні ідентифікатори: оголошення прив’язані до власників, пропозиції—до конкретного покупця та вибраного оголошення, а чати зв’язують учасників угоди. Така структура дозволяє оптимізувати процеси пошуку, вибору, реєстрації пропозицій і комунікації між сторонами.

3.3 Архітектура програмного забезпечення

У контексті розробки програмного забезпечення для онлайн-сервісу купівлі та продажу нерухомості, архітектура програмного забезпечення розроблена для забезпечення масштабованості, зручності обслуговування та надійності. Ця архітектура дотримується багаторівневої структури, яка розділяє завдання та спрощує процеси розробки та розгортання.

В основі лежить рівень презентації, який відповідає за обробку інтерфейсу користувача. Цей рівень безпосередньо взаємодіє з користувачами (покупцями та продавцями), дозволяючи їм переглядати оголошення, реєструватися, входити в систему, завантажувати інформацію про нерухомість, надсилати повідомлення та подавати запити на купівлю. Фронтенд зазвичай створюється з використанням сучасних веб-технологій, таких як HTML5, CSS3, JavaScript, та фреймворків, таких як Vue.js або React, що забезпечує швидку реакцію та зручний користувацький досвід.

Рівень додатків розташований між рівнем презентації та рівнем даних. Він містить бізнес-логіку, яка керує процесом обробки даних та взаємодією користувачів із системою. Цей рівень реалізовано за допомогою PHP з фреймворком Symfony, який сприяє чіткій організації коду завдяки модульному та сервісно-орієнтованому підходу. Тут обробляються такі завдання, як перевірка форм, автентифікація користувачів, управління ролями, управління оголошеннями та обробка повідомлень [18].

Під логікою програми знаходиться рівень доступу до даних, який відповідає за керування зв'язком з базою даних. Цей рівень гарантує, що всі

операції з базою даних, такі як створення, читання, оновлення та видалення записів, виконуються ефективно та безпечно. Symfony Doctrine ORM (об'єктно-реляційне відображення) використовується для абстрагування запитів до бази даних та полегшення взаємодії з базою даних MySQL через класи сутностей.

В основі лежить рівень бази даних, що працює на MySQL. Цей рівень зберігає всі постійні дані, включаючи облікові записи користувачів, списки нерухомості, зображення, запити на покупку, повідомлення та історію статусів. Реляційна структура оптимізована за допомогою зовнішніх ключів та індексації для забезпечення цілісності даних та швидкого доступу.

Архітектура також включає додаткові допоміжні служби, такі як сповіщення електронною поштою (для повідомлень та оновлень схвалення), хмарне сховище для завантажених зображень та системи реєстрації для моніторингу активності та помилок.

Безпека забезпечується на всіх рівнях. Перевірка вхідних даних, контроль доступу на основі ролей, зашифрована передача даних (SSL) та безпечне керування сеансами користувачів інтегровані для захисту даних користувачів та запобігання несанкціонованому доступу.

Підсумовуючи, архітектура програмного забезпечення для цієї платформи нерухомості дотримується багаторівневого та модульного підходу, використовуючи Symfony для логіки серверної частини, MySQL для зберігання даних та сучасний фреймворк JavaScript для фронтенду. Така структура забезпечує чіткий розподіл завдань, покращує зручність обслуговування та підтримує майбутнє зростання та масштабованість.

Під час розробки програмного забезпечення для онлайн-сервісу, присвяченого купівлі та продажу нерухомості, рішення про використання 4-рівневої архітектури було зумовлене необхідністю чіткого розподілу завдань, покращеної масштабованості, зручності обслуговування та довгострокової адаптивності. Кожен рівень у цій архітектурі має чітку відповідальність, що

дозволяє системі функціонувати ефективно та безпечно, а також бути легшою в управлінні та розширенні в майбутньому.

Перший рівень – це рівень презентації, який обробляє всі взаємодії між користувачем та системою. Це включає користувацькі інтерфейси, такі як веб-сторінки та форми, через які покупці та продавці можуть шукати нерухомість, створювати оголошення, спілкуватися один з одним або подавати запити на купівлю. Ізолюючи цей рівень, можна постійно покращувати взаємодію користувача, не впливаючи на бізнес-логіку чи структуру даних [19].

Другий рівень – це рівень додатків, який відповідає за координацію дій між логікою презентації та бізнес-логікою. Цей рівень діє як місток, який направляє запити, застосовує бізнес-правила та обробляє вхідні дані користувача, перш ніж вони взаємодіють з основною системою. Завдяки інкапсуляції логіки керування, систему легше тестувати та модифікувати, коли змінюються вимоги користувача.

Третій рівень, рівень бізнес-логіки, містить основні правила та функції, що визначають принцип роботи платформи, такі як обробка схвалень оголошень, перевірка запитів на покупку, керування ролями користувачів та розрахунок змін статусу. Ізоляція бізнес-логіки гарантує, що зміни в політиці або функціях можна впроваджувати, не впливаючи на інші рівні, що підвищує гнучкість та повторне використання коду.

Рівень доступу до даних керує взаємодією з базою даних. Він відповідає за зберігання та отримання постійних даних, таких як оголошення, користувачі, повідомлення та зображення. Використання рівня абстракції, такого як Doctrine ORM з MySQL, забезпечує безпечний та послідовний доступ до даних, одночасно підтримуючи простішу міграцію та оптимізацію.

Розділивши програму на ці чотири рівні, процес розробки стає більш структурованим та ефективним. Кожна команда або розробник може зосередитися на певній області, мінімізуючи помилки та зменшуючи взаємозалежності. Крім того, ця багаторівнева архітектура підтримує майбутні

вдосконалення, такі як інтеграція мобільних додатків, сторонніх API або розширеної аналітики, без необхідності значних переробок [19].

Загалом, вибір 4-рівневої архітектури гарантує, що програмне забезпечення залишається надійним, безпечним та адаптивним, що є важливим для довгострокового успіху платформи нерухомості, де високий трафік, динамічний контент та надійна продуктивність є критично важливими.

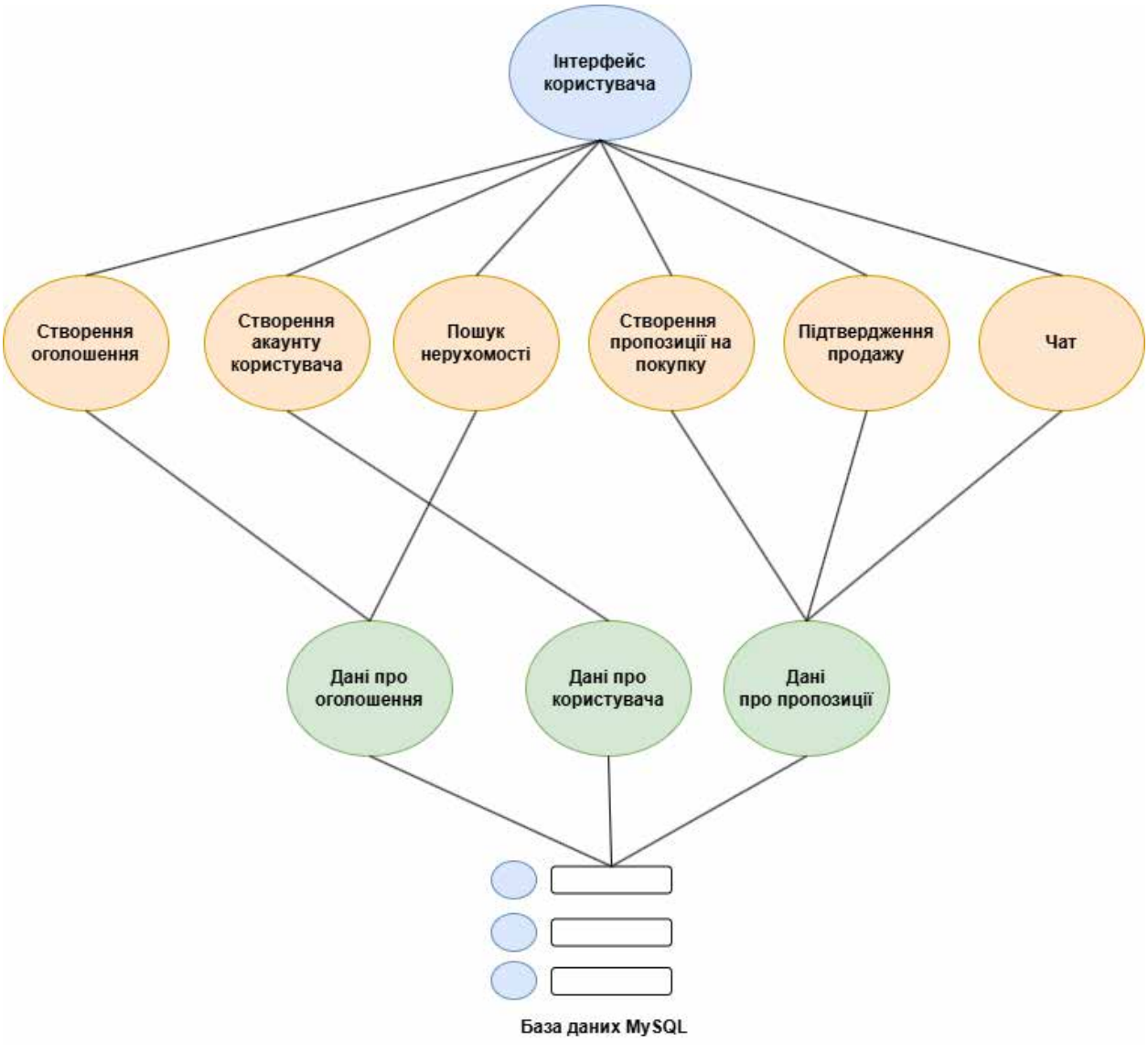

Рис. 3.3 Багатошарова архітектура

Чотиришарова архітектура програмного забезпечення у цій системі нерухомості структурована таким чином, щоб забезпечити ефективну взаємодію між користувачами та платформою. Вона включає такі основні рівні:

Презентаційний рівень — відповідає за взаємодію з користувачем. Тут реалізовано інтерфейс для покупця нерухомості, що дозволяє йому здійснювати пошук, переглядати оголошення та ініціювати пропозиції. Також у цьому шарі забезпечується доступ до чату для комунікації з власником нерухомості.

Логічний рівень — містить бізнес-логіку системи. Тут обробляються запити користувача, перевіряється активність оголошень, реєструються пропозиції на покупку та створюється чат. Взаємодія між об’єктами системи регулюється відповідно до встановлених правил.

Рівень доступу до даних — відповідає за обробку запитів до бази даних. Він керує збереженням оголошень, реєстрацією пропозицій, налаштуванням чату та доступом до повідомлень. У цьому шарі забезпечується оптимізація операцій читання і запису даних.

Рівень зберігання даних — містить саму базу даних, де зберігаються ключові сутності: користувачі, оголошення, пропозиції та історія повідомлень у чаті. Цей рівень гарантує цілісність і захист інформації.

Такий багатошаровий підхід дозволяє модульно організувати роботу системи, спрощуючи її масштабування та підтримку.

3.4 Організаційна структура програмного забезпечення

3.4.1 Діаграма пакетів. У контексті бакалаврської роботи під назвою «Програмне забезпечення для онлайн-сервісу купівлі та продажу нерухомості» діаграма пакета відіграє життєво важливу роль у представленні загальної модульної структури програми. Вона візуально окреслює, як система поділена на окремі логічні компоненти або пакети, кожен з яких інкапсулює пов'язану функціональність. Це архітектурне рішення допомагає забезпечити організованість, масштабованість та легше обслуговування проєкту в міру його зростання [20].

В основі системи лежить пакет керування користувачами, який відповідає за обробку всіх процесів, пов'язаних з обліковими записами користувачів. Це включає реєстрацію, вхід та управління ролями, гарантуючи, що як покупці, так і продавці мають доступ до відповідних функцій. Компоненти безпеки, такі як служби автентифікації та керування сеансами, також є частиною цього пакета, забезпечуючи безпечну та ефективну взаємодію з користувачами.

Модуль керування оголошеннями обробляє створення, зберігання та відображення оголошень про нерухомість. Кожне оголошення може містити детальні описи, ціни, місцезнаходження та інші відповідні атрибути. Підтримуючи це, компоненти обробки зображень керують завантаженням та асоціацією візуального контенту, щоб допомогти ефективно представити нерухомість потенційним покупцям.

Взаємодія між користувачами також здійснюється через систему обміну повідомленнями. Цей пакет підтримує внутрішні комунікації, пов'язані з конкретними оголошеннями, дозволяючи користувачам обмінюватися інформацією, ставити запитання або обговорювати умови безпосередньо на платформі. Ці розмови зазвичай пов'язані з окремими оголошеннями, що сприяє релевантній та організованій комунікації.

Ключовою особливістю платформи є можливість для покупців подавати запити на купівлю або оренду. Ця функціональність керується спеціальним пакетом, який займається створенням, відстеженням та оновленням статусу цих запитів. Він підтримує весь процес транзакції від початкового інтересу до потенційного схвалення або відхилення продавцем.

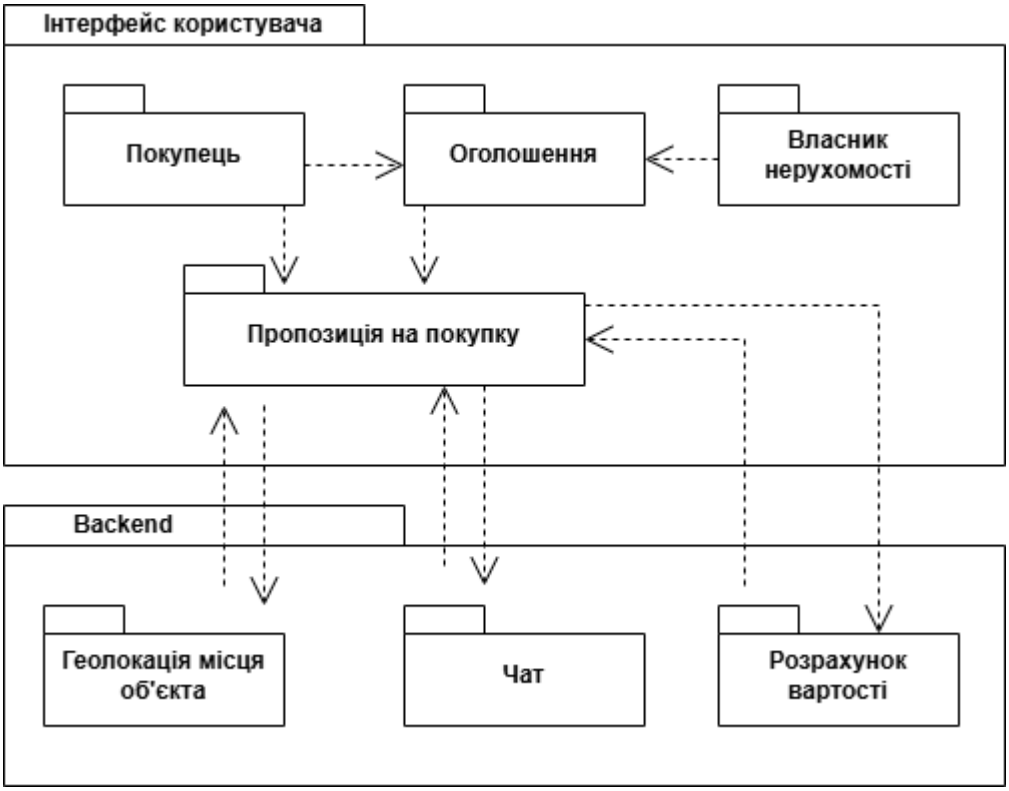

Рис. 3.4 Діаграма пакетів

Ця діаграма пакетів моделює структуру програмного забезпечення для системи купівлі нерухомості, розподіляючи функціональні компоненти на логічні групи для забезпечення гнучкості та організованості.

Пакет “Інтерфейс користувача” містить модулі, відповідальні за взаємодію з покупцем нерухомості, включаючи перегляд оголошень, реєстрацію пропозицій і доступ до чату. Він є точкою входу, що дозволяє користувачу легко здійснювати навігацію в системі.

Пакет “Бізнес-логіка” управляє основними процесами, такими як перевірка активності оголошень, обробка пропозицій на покупку та створення чату між учасниками угоди. Він забезпечує коректність операцій і дотримання умов платформи.

Пакет “Обробка даних” містить механізми роботи з базою даних, включаючи отримання списку оголошень, збереження пропозицій і управління повідомленнями. Він відповідає за ефективне та безпечне збереження інформації.

Пакет “Зберігання даних” складається з фізичної бази даних, де розташовані таблиці користувачів, оголошень, пропозицій та чату. Цей рівень гарантує стабільність платформи, забезпечуючи цілісність і швидкий доступ до інформації.

Діаграма демонструє чіткі залежності між пакетами, дозволяючи системі працювати ефективно та підтримувати масштабованість.

3.5 Вибір інструментарію для створення програмного забезпечення

Під час розробки програмного забезпечення для онлайн-сервісу, присвяченого купівлі-продажу нерухомості, було ретельно розглянуто вибір інструментів та технологій для забезпечення ефективності, масштабованості та простоти обслуговування. Кожен обраний інструмент сприяє безперебійному функціонуванню системи та підтримує конкретні потреби платформи, від обробки на серверній частині до дизайну інтерфейсу користувача.

Основною мовою програмування обрано PHP, надійне та широко використовуване рішення для веб-розробки. Він відомий своєю чудовою сумісністю з веб-серверами та великою екосистемою, яка включає незліченну кількість бібліотек та фреймворків. Для розширення можливостей PHP та забезпечення добре структурованого, модульного серверної частини було обрано фреймворк Symfony. Symfony пропонує потужну основу для створення складних, масштабованих веб-додатків та сприяє таким передовим практикам, як повторно використовуваний код, впровадження залежностей та налаштування за допомогою анотацій.

Для зберігання та пошуку даних було обрано MySQL як систему управління реляційними базами даних. MySQL пропонує надійність, продуктивність та легкість інтеграції з додатками на базі PHP. Щоб подолати розрив між об'єктно-орієнтованим програмуванням та реляційними базами даних, було використано Doctrine ORM. Цей інструмент спрощує операції з базою даних, дозволяючи розробникам працювати з об'єктами PHP, а не писати сирий SQL, зменшуючи кількість шаблонного коду та підвищуючи зручність обслуговування.

Середовище розробки працює на базі PHPStorm, професійного IDE, яке надає інтелектуальну допомогу в кодуванні, розширені інструменти налагодження та безшовну інтеграцію з системами контролю версій. Підтримка Symfony та Doctrine ще більше пришвидшує розробку, пропонуючи аналіз коду в режимі реального часу, можливості автозаповнення та рефакторингу, адаптовані до структури проєкту.

Для розробки фронтенду було використано комбінацію HTML, CSS та JavaScript. HTML структурує вміст веб-сторінок, CSS забезпечує адаптивний та візуально привабливий дизайн, а JavaScript додає інтерактивності, роблячи користувацький досвід динамічним та захопливим. Ці технології дозволяють створити зручний інтерфейс, який підтримує інтуїтивно зрозумілу навігацію та безперебійну взаємодію з системою.

Для проєктування користувацького інтерфейсу та планування макета платформи як інструмент проєктування було обрано Figma. Він дозволяє створювати інтерактивні прототипи та забезпечує спільну роботу з проєктуванням, що полегшує візуалізацію кінцевого продукту перед впровадженням.

Разом ці інструменти забезпечують комплексний та сучасний технологічний стек, придатний для створення повнофункціонального, безпечного та орієнтованого на користувача веб-додатку для нерухомості.

4 ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЯ СИСТЕМИ

4.1 Вимоги до апаратного та програмного забезпечення

Для успішної розробки та розгортання програмного забезпечення для онлайн-сервісу, орієнтованого на купівлю-продаж нерухомості, важливо чітко визначити вимоги як до апаратного, так і до програмного забезпечення. Ці специфікації забезпечують ефективну роботу платформи на етапах розробки, тестування та виробництва, а також її масштабування відповідно до потреб користувачів.

Вимоги до апаратного забезпечення

Щодо розробки, достатньо персонального комп'ютера середньої або високої продуктивності. Рекомендована конфігурація включає багатоядерний процесор (наприклад, Intel Core i5 або AMD Ryzen 5 і вище), щонайменше 8 ГБ оперативної пам'яті (бажано 16 ГБ для плавнішої багатозадачності) та твердотільний накопичувач (SSD) для швидких операцій читання/запису. Для комфортної роботи із середовищами розробки та інструментами проєктування бажано використовувати дисплей Full HD. Стабільне підключення до Інтернету також має вирішальне значення для доступу до віддалених репозиторіїв, залежностей та API.

Для середовища розгортання (тобто сервера, на якому розміщено програму) апаратне забезпечення слід вибирати на основі очікуваного навантаження користувача. Для розгортання малого та середнього масштабу рекомендується віртуальний приватний сервер (VPS) або хмарний екземпляр щонайменше з 2 віртуальними процесорами, 4–8 ГБ оперативної пам'яті та 80–100 ГБ SSD-сховища. Зі зростанням трафіку користувачів конфігурацію сервера можна масштабувати відповідно або перейти на контейнерну чи хмарну архітектуру.

Вимоги до програмного забезпечення

Стек розробки базується на технологіях з відкритим кодом. Бекенд використовує PHP 8.x або вище в поєднанні з фреймворком Symfony, що забезпечує міцну основу для організації коду модульним та зручним у підтримці способом. База даних управляється за допомогою MySQL, що забезпечує надійність та сумісність із програмами на основі PHP. Doctrine ORM використовується для ефективнішої обробки взаємодії з даними за допомогою об'єктно-реляційного відображення.

Середовище розробки підтримується PHPStorm, потужним IDE, яке добре інтегрується з Symfony та Doctrine, забезпечуючи інтелектуальне автодоповнення коду, інструменти рефакторингу та можливості налагодження. Контроль версій здійснюється за допомогою Git, а для спільної роботи над кодом можна використовувати такі хостингові платформи, як GitHub або GitLab.

Фронтенд-розробка вимагає сучасного веб-браузера та використовує стандартні веб-технології: HTML5, CSS3 та JavaScript. За потреби можна інтегрувати додаткові фронтенд-фреймворки або бібліотеки, такі як Bootstrap для адаптивного дизайну або Vue.js/React для динамічних компонентів інтерфейсу користувача.

Проєктування та прототипування виконуються у Figma, спільному веб-інструменті, який спрощує створення вайрфреймів, макетів інтерфейсу користувача та інтерактивних прототипів.

Програмний стек сервера включає операційну систему на базі Linux (наприклад, Ubuntu Server), веб-сервер (Apache або Nginx), середовище виконання PHP, сервер MySQL та Composer для керування залежностями. Також повинні бути впроваджені заходи безпеки, такі як SSL-сертифікати та конфігурації брандмауера, для забезпечення захисту даних.

Таке поєднання апаратного та програмного забезпечення створює надійне середовище як для розробки, так і для розгортання, забезпечуючи високу продуктивність, безпеку та масштабованість платформи нерухомості.

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

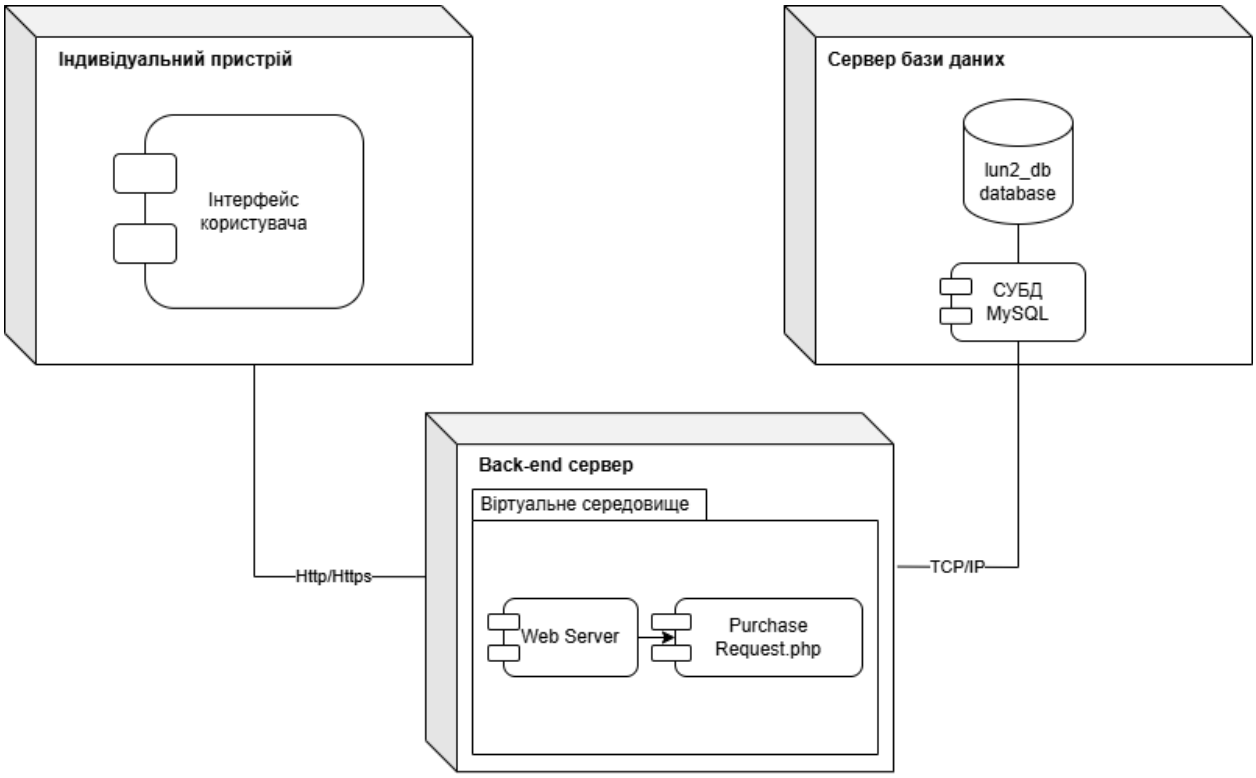

Рис. 4.1 Діаграма розгортання

4.2 Тестування системи

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

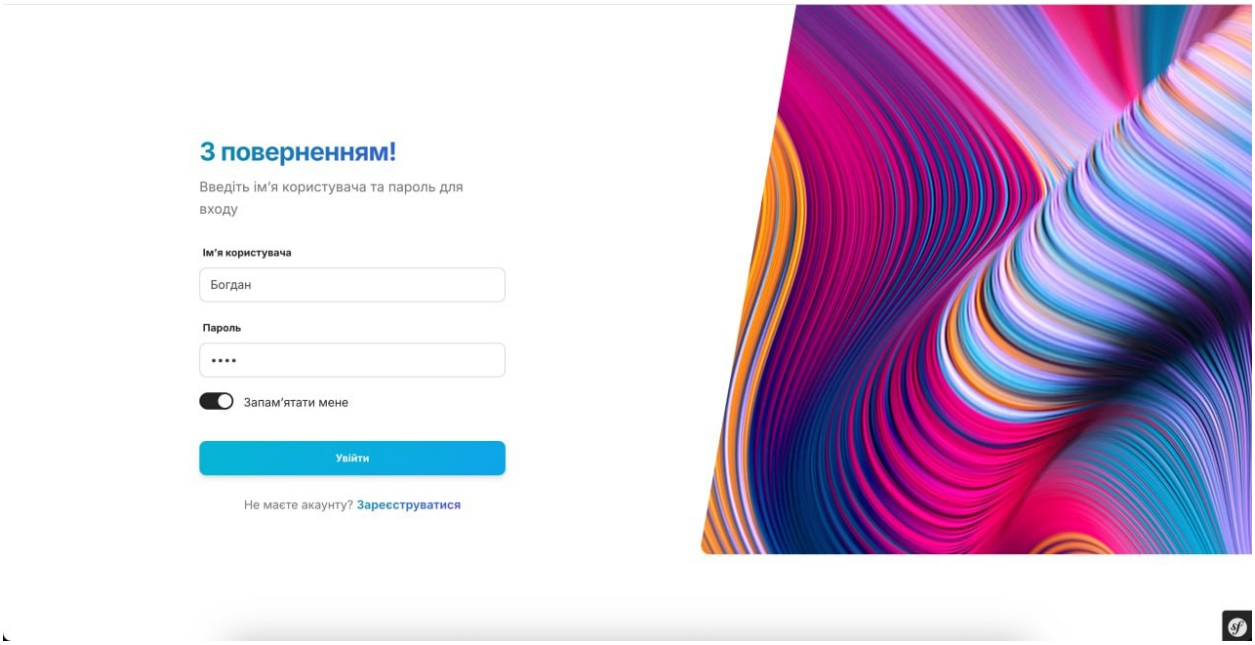

Рис. 4.2 Форма авторизації

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

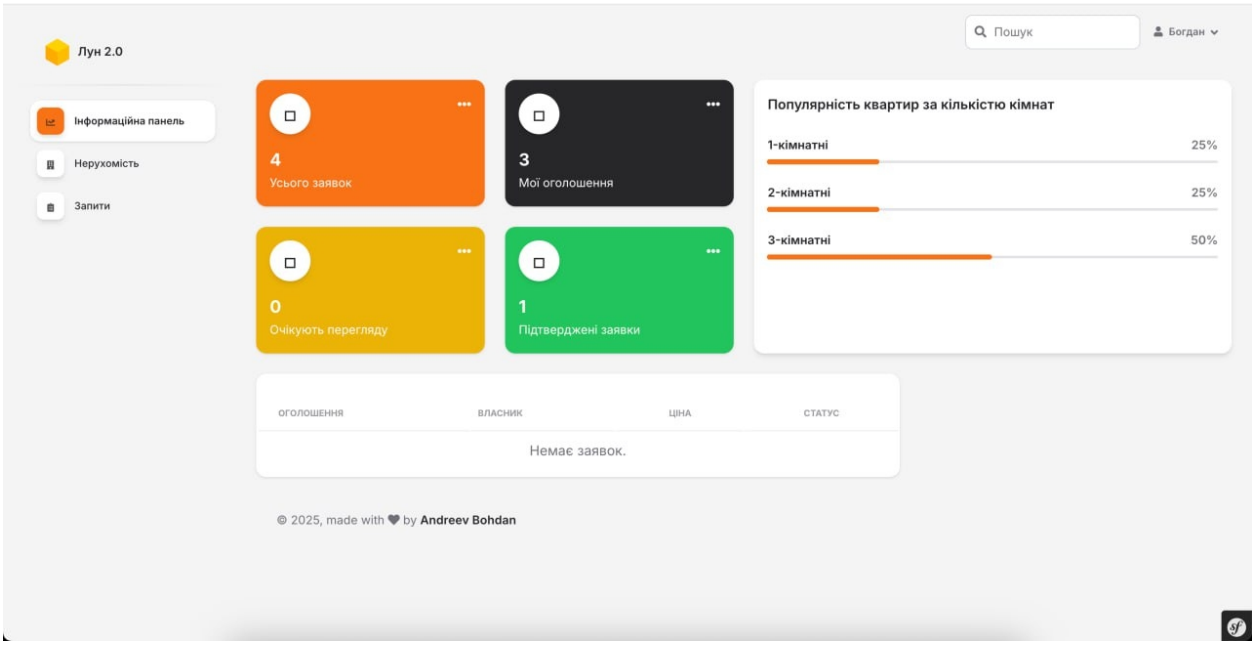

Рис. 4.3 Головна сторінка

Переходимо на сторінку "Нерухомість" і у нас виводиться список всіх створених оголошень (Рис. 4.4), ми можемо шукати цікаві для нас варіанти за ключовими словами, а також відкривати самостійно.

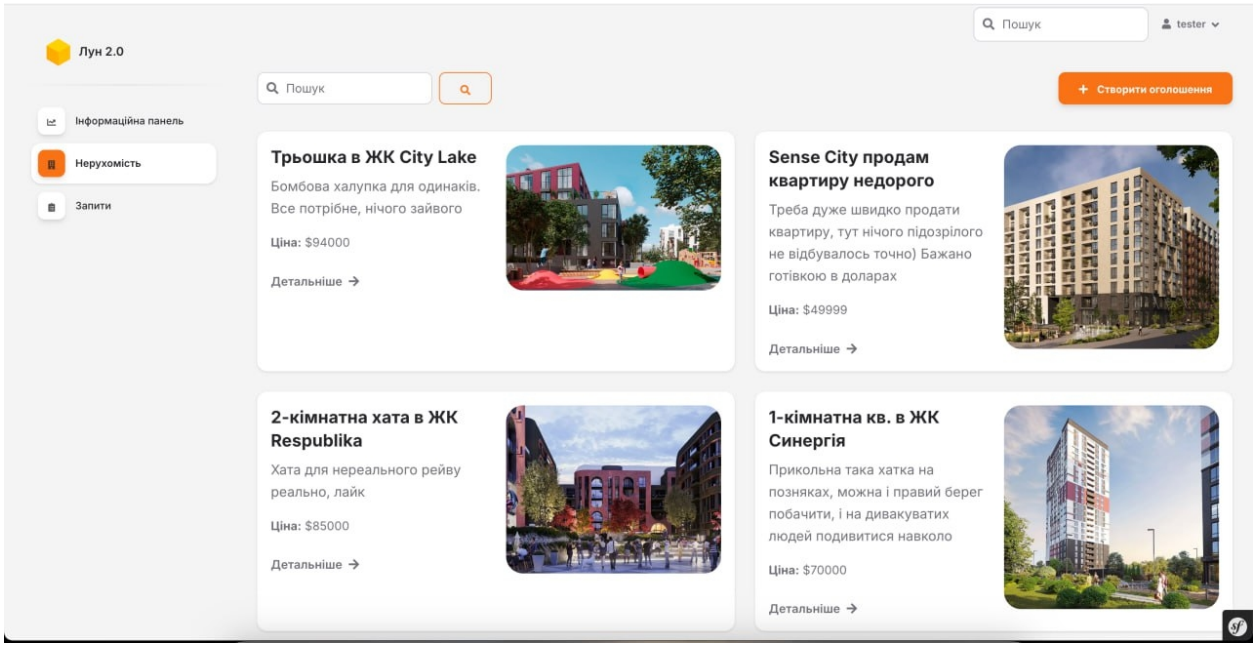

Рис. 4.4 Сторінка “Нерухомість”

Спробуємо створити оголошення, для цього нам потрібно натиснути кнопку створення оголошення на сторінці з пошуком нерухомості, яка нам відкриє форму заповнення де нам необхідно ввести всі дані про житло: назву, опис, фотографії, ціну і т.д. Після підтвердження, наше оголошення буде успішно додано до бази даних (Рис. 4.5).

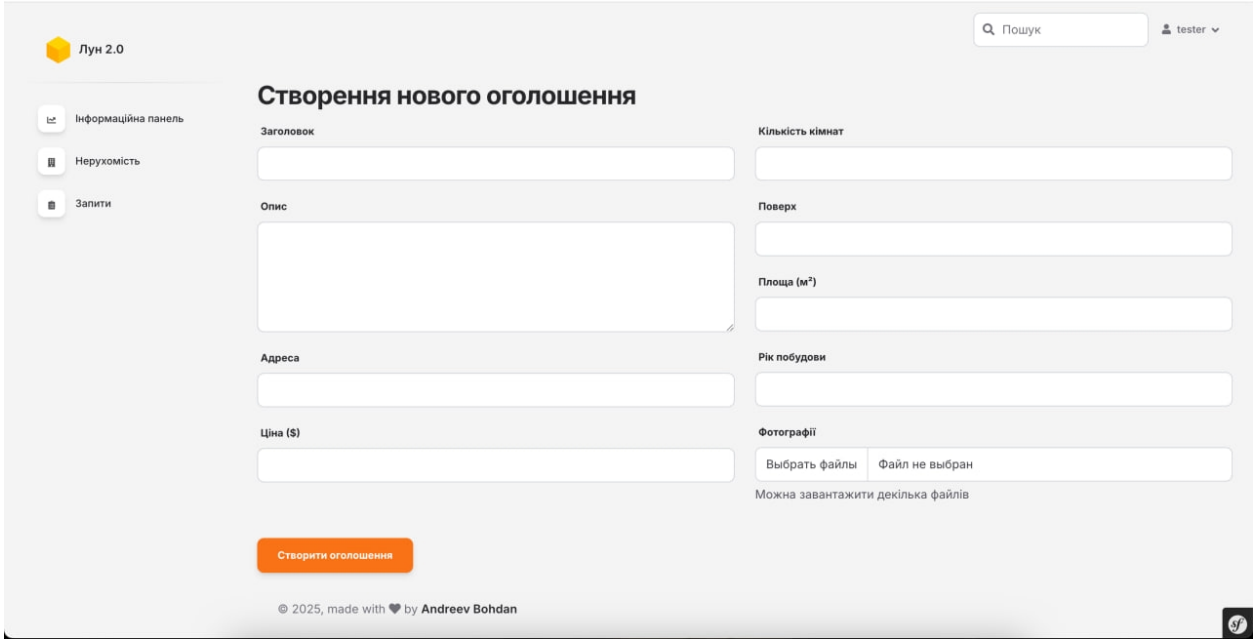

Рис. 4.5 Сторінка створення оголошення

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 придбання (Рис. 4.6).

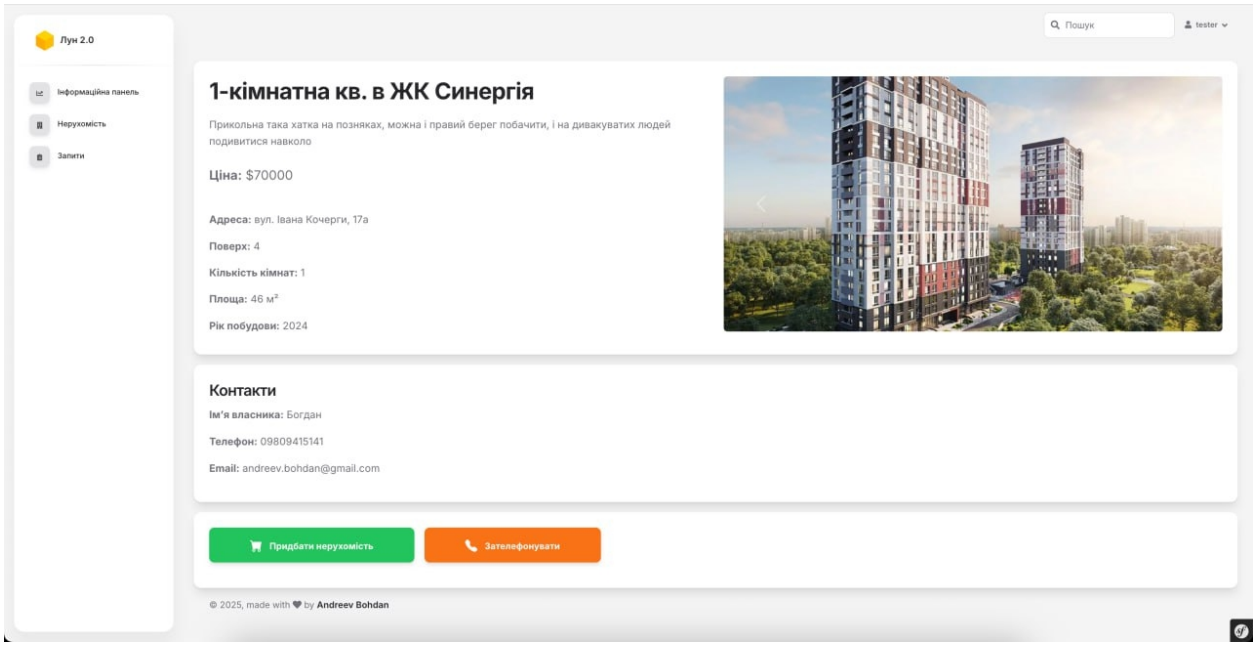

Рис. 4.6 Сторінка новоствореного оголошення

Якщо ми захочемо придбати житло, переходимо на сторінку оформлення заявки, то нам треба вказати повідомлення при потребі та підтвердити заявку (Рис. 4.7).

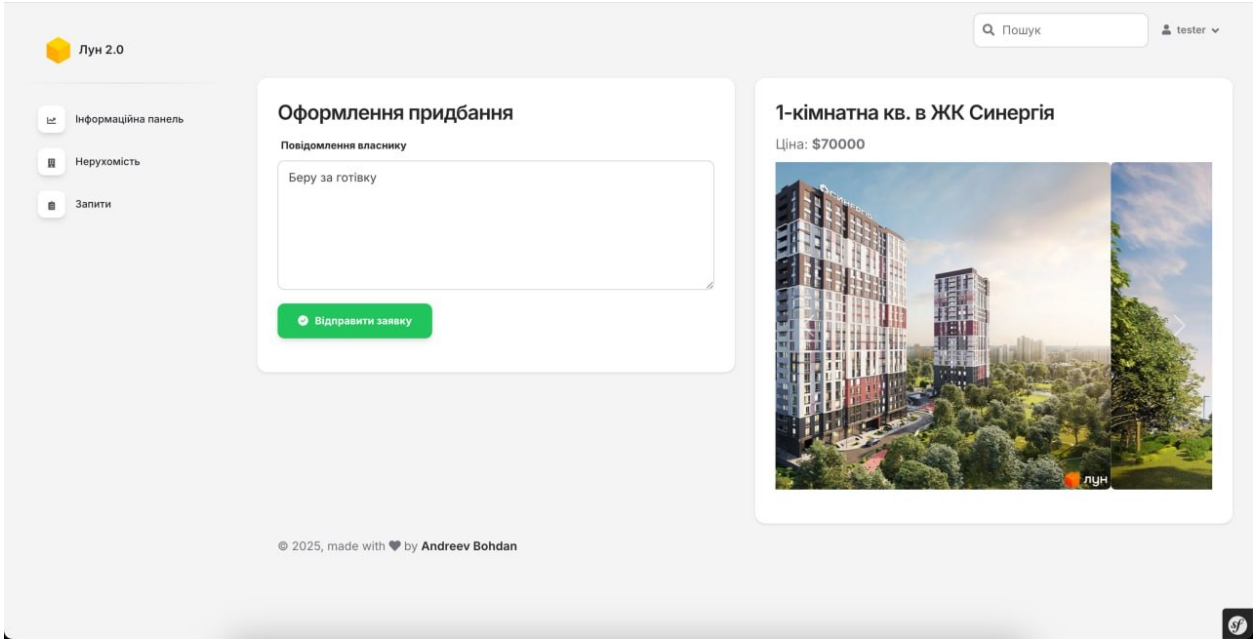

Рис. 4.7 Оформлення заявки на придбання

Рис. 4.8 Повідомлення про успішно створену заявку

Переходимо на сторінку пропозицій, щоб побачити список усіх заявок на придбання наших оголошень від покупців (Рис. 4.9).

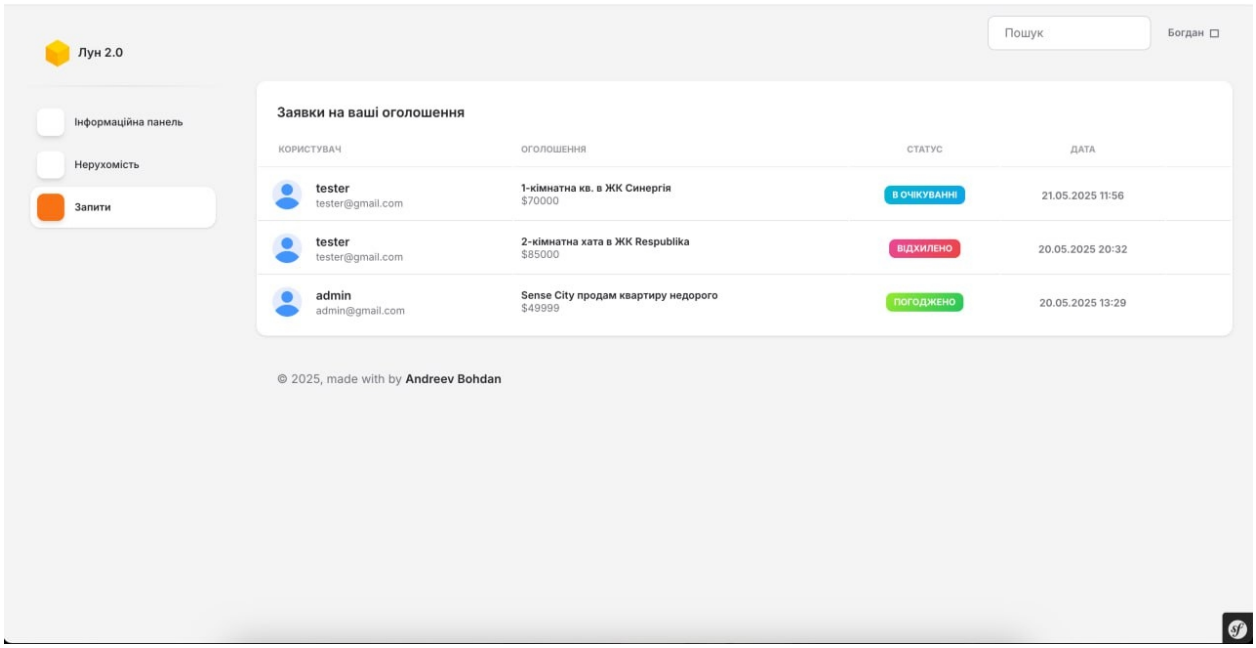

Рис. 4.9 Сторінка “Запити”

Вибираємо якусь заявку та переходимо на сторінку обговорення. Тут у нас представлена вся інформація про придбання, а також чат для спілкування та переговорів між власником та покупцем. Власник оголошення може підтвердити або відмовити у придбанні (Рис. 4.10 – 4.11).

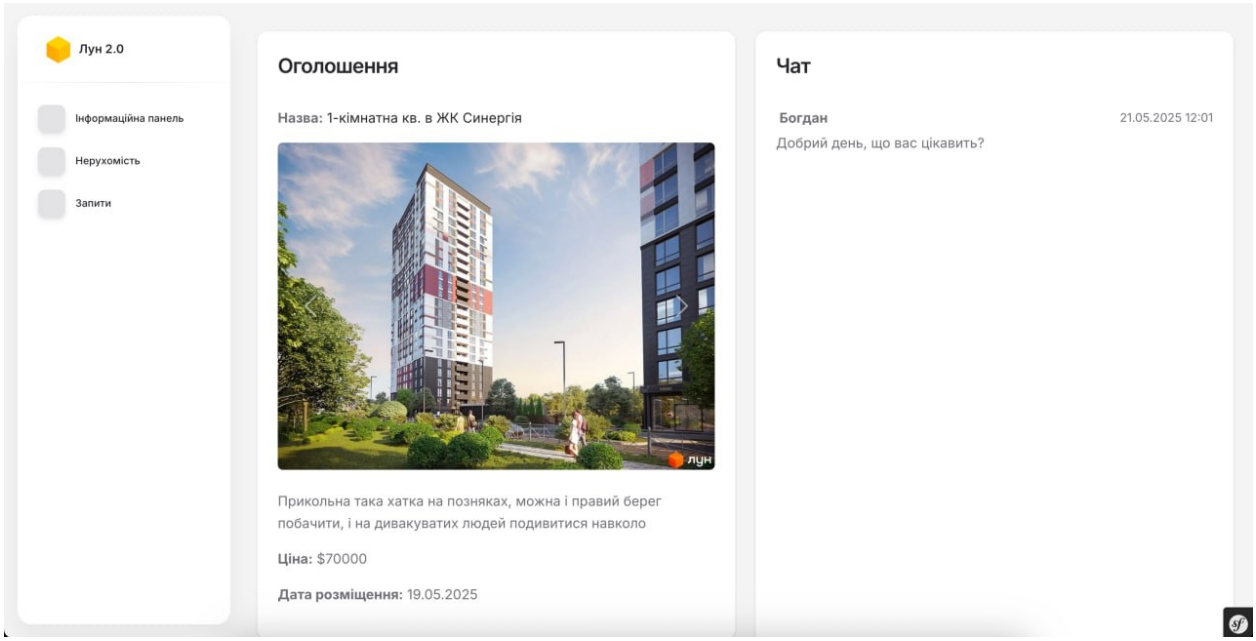

Рис. 4.10 Деталі заявки на бронювання

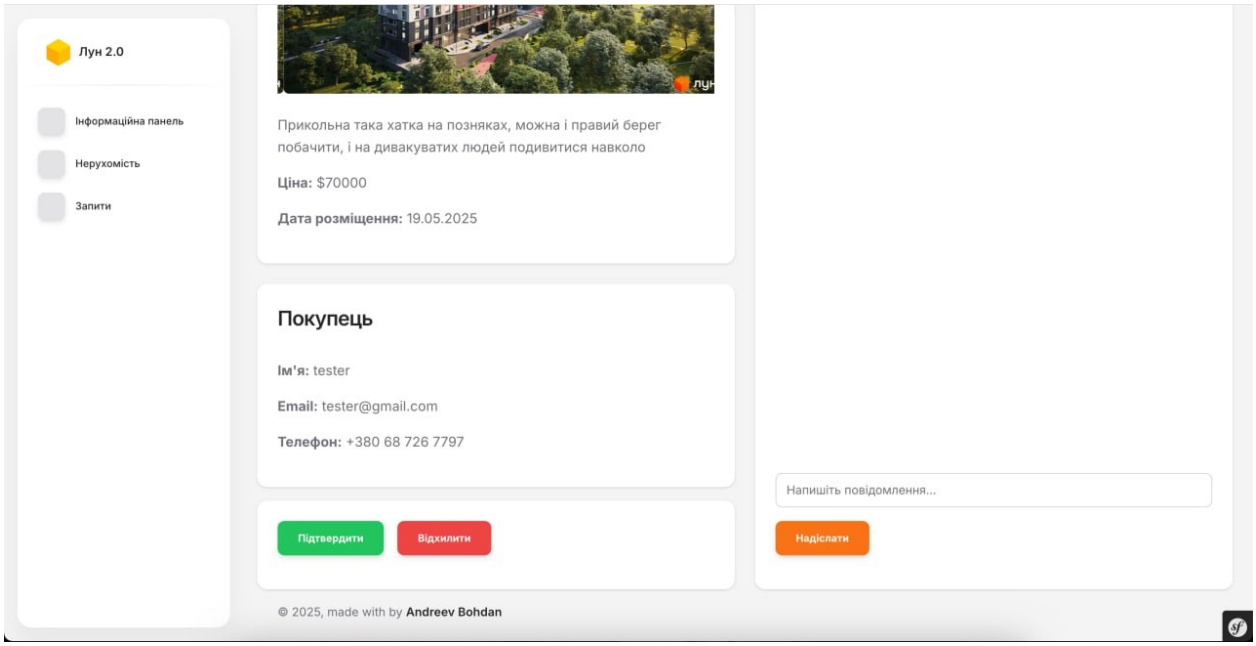

Рис. 4.11 Деталі заявки на бронювання

Після того як власник нерухомості підтвердить пропозицію, у цій пропозиції з'явиться плашка з інформацією про те, що пропозиція була схвалена (Рис. 4.12).

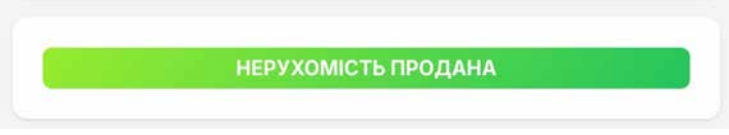

Рис. 4.12 Повідомлення про погодження продажу нерухомості

ВИСНОВКИ

На завершення, розроблене програмне забезпечення для онлайн-сервісу купівлі-продажу нерухомості пропонує комплексне цифрове рішення, адаптоване до потреб сучасних користувачів. Воно спрощує взаємодію між продавцями нерухомості та потенційними покупцями, дозволяючи створювати, переглядати та керувати оголошеннями про нерухомість у безпечному та інтуїтивно зрозумілому середовищі. Ця система значно зменшує залежність від фізичних агентств, роблячи процес угоди з нерухомістю більш доступним та ефективним.

Рішення використовувати PHP у поєднанні з фреймворком Symfony виявилося міцною основою для створення надійного та масштабованого веб-додатку. Модульність Symfony та розгалужена екосистема дозволили швидко розробляти та інтегрувати найкращі практики. MySQL було обрано як систему управління базами даних завдяки її надійності, високій продуктивності та безшовній інтеграції з Doctrine ORM, що спростило маніпулювання даними та покращило підтримку кодової бази.

Впровадження 4-рівневої архітектури забезпечило чіткий розподіл обов'язків, що полегшило керування, тестування та масштабування додатка. Кожен рівень – презентація, додаток, домен та інфраструктура – відіграє окрему роль, сприяючи гнучкості та повторному використанню компонентів. Цей архітектурний вибір гарантує, що система може розвиватися в майбутньому без необхідності суттєвої реструктуризації.

Інструменти розробки, такі як PHPStorm та Figma, відіграли вирішальну роль у вдосконаленні процесів розробки та дизайну. PHPStorm забезпечив потужне середовище кодування з можливостями налагодження, тоді як Figma дозволив спільний та ефективний дизайн UI/UX. Використання HTML, CSS та JavaScript забезпечило адаптивний та зручний інтерфейс, покращуючи загальний користувацький досвід.

З функціональної точки зору, система підтримує такі критично важливі функції, як управління оголошеннями, реєстрація та автентифікація користувачів, обробка зображень, обмін повідомленнями та подання запитів на купівлю. Ці функції працюють разом, щоб створити безперебійний робочий процес для користувачів, які займаються операціями з нерухомістю, підвищуючи прозорість та зручність.

Загалом, проєкт демонструє успішне застосування теоретичних знань для вирішення практичної проблеми. Він закладає основу для подальших удосконалень та реального впровадження, демонструючи значний потенціал для використання в галузі нерухомості. Ця робота не лише відповідає академічним вимогам, але й служить цінним прототипом комерційно життєздатного програмного рішення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ling, D. C., Archer, W. R. Принципи нерухомості: ціннісний підхід. — McGraw-Hill Education, 2022.
2. Geltner, D., Miller, N. G., Clayton, J., Eichholtz, P. Аналіз комерційної нерухомості та інвестиції. — Cengage Learning, 2014.
3. Ratcliffe, J., Stubbs, M., Keeping, M. Міське планування та розвиток нерухомості. — Routledge, 2021.
4. Лун – [Електронний ресурс] – Режим доступу: https://lun.ua/?srsltid=AfmBOooMxo7vNjbNsFygQgLrfYHQ5042L4Rj8vgzDCkU413RbFD6H3qI
5. Zillow – [Електронний ресурс] – Режим доступу: https://www.zillow.com/buy/app-download/
6. Фрідман, Дж. П., Гарріс, Дж. К., Ліндеман, Н. Словник термінів нерухомості. — Barron's Educational Series, 2017.
7. Галлохер, Дж. Інформаційні системи: Посібник менеджера з використання технологій. — FlatWorld, 2020.
8. Фаулер, М. Шаблони архітектури корпоративних додатків. — Addison-Wesley, 2002.
9. Поттс, К. Ф. Управління витратами на будівництво: навчання на прикладах. — Routledge, 2013.
10. Курняван, Б. Початки PHP та MySQL: від новачка до професіонала. — Apress, 2021.
11. Потенсьє, Ф. Symfony: The Fast Track. — Symfony SAS, 2023.
12. Веллінг, Л., Томсон, Л. Веб-розробка на PHP та MySQL. — Pearson Education, 2017.
13. Ніксон, Р. Вивчення PHP, MySQL та JavaScript: за допомогою jQuery, CSS та HTML5. — O’Reilly Media, 2018.

14. Thill, J. V., Bovee, C. L. Excellence in Business Communication. — Pearson, 2019. (Корисно для розуміння взаємодії користувачів на платформах нерухомості.)
15. Walker, R. The Digital Deal: Real Estate Transactions in the Information Age. — Real Estate Press, 2020.
16. Національна асоціація ріелторів. 2023 Profile of Home Buyers and Sellers. — NAR Research Report, 2023 – [Електронний ресурс] – Режим доступу: www.nar.realtor
17. Doctrine ORM – [Електронний ресурс] – Режим доступу: https://www.doctrine-project.org/projects/orm.html
18. Документація Symfony – [Електронний ресурс] – Режим доступу: https://symfony.com/doc/current/index.html
19. Документація MySQL – [Електронний ресурс] – Режим доступу: https://dev.mysql.com/doc/
20. Посібник з PHP – [Електронний ресурс] – Режим доступу: https://www.php.net/manual/en/
21. Документація Git – [Електронний ресурс] – Режим доступу: https://git-scm.com/doc
22. Посібник з HTML W3Schools – [Електронний ресурс] – Режим доступу: https://www.w3schools.com/html/
23. Посібник з CSS W3Schools – [Електронний ресурс] – Режим доступу: https://www.w3schools.com/css/
24. Посібник з JavaScript W3Schools – [Електронний ресурс] – Режим доступу: https://www.w3schools.com/js/
25. Інструмент для створення діаграм Draw.io – [Електронний ресурс] – Режим доступу: https://app.diagrams.net/
26. ПРОЄКТУВАННЯ ER-ДІАГРАМИ – [Електронний ресурс] – Режим доступу: https://nationalteam.worldskills.ua/skills/proektirovanie-er-diagrammy/

27. Діаграма варіантів використання (UseCase diagram) – [Електронний ресурс] – Режим доступу: https://flexberry.github.io/ua/fd_use-case-diagram.html
28. 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/
29. What is Sequence Diagram? – [Електронний ресурс] – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/
30. UML - Activity Diagrams – Tutorialspoint – [Електронний ресурс] – Режим доступу: https://www.tutorialspoint.com/uml/uml_activity_diagram.htm
31. Побудова діаграми класів – [Електронний ресурс] – Режим доступу: https://flexberry.github.io/ua/gpg_class-diagram.html
32. UML-діаграми класів – [Електронний ресурс] – Режим доступу: https://prog-cpp.ua/uml-classes/
33. Entity Relationship Diagram - Data Modeling – [Електронний ресурс] – Режим доступу: https://www.visualparadigm.com/VPGallery/datamodeling/EntityRelationshipDiagram.html
34. UML 2 Tutorial - Package Diagram - Sparx Systems – [Електронний ресурс] – Режим доступу: https://sparxsystems.com/resources/tutorials/uml2/package-diagram.html
35. Документація Bootstrap (офіційний сайт) – [Електронний ресурс] – Режим доступу: www.getbootstrap.com/documentation.

ДОДАТОК А

Фрагменти програмного коду. Функція виставлення нерухомості на продаж

```
class ListingController extends AbstractController
{
    #[Route('/listings', name: 'app_listing')]
    public function index(Request $request, EntityManagerInterface $entityManager): Response
    {
        $searchTerm = $request->query->get('q');

        $repository = $entityManager->getRepository(Listing::class);

        if ($searchTerm) {
            $qb = $repository->createQueryBuilder('l');
            $qb->where(
                $qb->expr()->orX(
                    $qb->expr()->like('l.title', ':search'),
                    $qb->expr()->like('l.description', ':search')
                )
            )
                ->setParameter('search', '%' . $searchTerm . '%')
                ->orderBy('l.createdAt', 'DESC');

            $listings = $qb->getQuery()->getResult();
        } else {
            $listings = $repository->findBy([], ['createdAt' => 'DESC']);
        }

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

    #[Route('/listings/create', name: 'app_listing_create', methods: ['GET', 'POST'])]
    #[IsGranted('ROLE_USER')]
    public function create(Request $request, EntityManagerInterface $em, SluggerInterface
$slugger): Response
    {
        if ($request->isMethod('POST')) {
            $listing = new Listing();
            $listing->setTitle($request->request->get('title'));
            $listing->setDescription($request->request->get('description'));
            $listing->setPrice((float) $request->request->get('price'));
            $listing->setAddress($request->request->get('address'));
            $listing->setFloor($request->request->get('floor'));
            $listing->setRooms((int) $request->request->get('rooms'));
            $listing->setSize($request->request->get('size'));
            $listing->setBuildYear((int) $request->request->get('buildYear'));
            $listing->setStatus(Listing::LISTING_ACTIVE);
```

```
        $listing->setCreatedAt(new DateTimeImmutable());
        $listing->setUser($this->getUser());

        /** @var UploadedFile[] $uploadedFiles */
        $uploadedFiles = $request->files->get('images');
        if ($uploadedFiles) {
            foreach ($uploadedFiles as $uploadedFile) {
                if ($uploadedFile instanceof UploadedFile) {
                    $originalFilename = pathinfo($uploadedFile->getClientOriginalName(), PATHINFO_FILENAME);
                    $safeFilename = $slugger->slug($originalFilename);
                    $newFilename = $safeFilename . '-' . uniqid('', true) . '.' . $uploadedFile->guessExtension();

                    try {
                        $uploadedFile->move(
                            $this->getParameter('images_directory'),
                            $newFilename
                        );
                    } catch (FileException $e) {
                        flash()->error('Не вдалося завантажити зображення');
                        continue;
                    }

                    $image = new Image();
                    $image->setListing($listing);
                    $image->setFilePath('/uploads/' . $newFilename);
                    $em->persist($image);
                }
            }
        }

        $em->persist($listing);
        $em->flush();

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

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

#[Route('/listings/{id}', name: 'app_listing_show', requirements: ['id' => '\d+'])]
public function show(Listing $listing): Response
{
    return $this->render('listings/show.html.twig', [
        'listing' => $listing,
        'images' => $listing->getImages(),
    ]);
}

#[Route('/listings/{id}/purchase', name: 'app_purchase_request', methods: ['GET', 'POST'])]
```

```
    #[IsGranted('ROLE_USER')]
    public function purchase(Request $request, Listing $listing, EntityManagerInterface $em): Response
    {
        $purchase = new PurchaseRequest();

        if ($request->isMethod('POST')) {
            $purchase->setListing($listing);
            $purchase->setBuyer($this->getUser());
            $purchase->setMessage($request->request->get('message'));
            $purchase->setStatus(PurchaseRequest::RENTAL_PENDING);
            $purchase->setCreatedAt(new \DateTimeImmutable());

            $em->persist($purchase);
            $em->flush();

            flash()->success('Заявка успішно оформлена');
            return $this->redirectToRoute('app_listing_show', ['id' => $listing->getId()]);
        }

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

ДОДАТОК Б

Фрагменти програмного коду. Функціонал придбання нерухомості

```
class RequestController extends AbstractController
{
    #[Route('/requests', name: 'app_requests')]
    #[IsGranted('ROLE_USER')]
    public function index(PurchaseRequestRepository $requestRepository): Response
    {
        $user = $this->getUser();

        $requests = $requestRepository->findByUserListings($user);

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

    #[Route('/requests/{id}', name: 'app_request_show')]
    #[IsGranted('ROLE_USER')]
    public function show(
        PurchaseRequest $requestEntity,
        Request $httpRequest,
        EntityManagerInterface $em,
        MessageRepository $messageRepository
    ): Response {
        $user = $this->getUser();

        if ($httpRequest->isMethod('POST')) {
            $message = new Message();
            $message->setSender($user);
            $message->setContent($httpRequest->request->get('message'));
            $message->setListing($requestEntity->getListing());
            $message->setCreatedAt(new \DateTimeImmutable());

            $em->persist($message);
            $em->flush();

            return $this->redirectToRoute('app_request_show', ['id' => $requestEntity->getId()]);
        }

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

        return $this->render('requests/show.html.twig', [
            'request' => $requestEntity,
            'messages' => $messages,
        ]);
```

```
    }

    #[Route('/requests/{id}/approve', name: 'app_request_approve', methods: ['POST'])]
    #[IsGranted('ROLE_USER')]
    public function approve(PurchaseRequest $requestEntity, EntityManagerInterface $em, Listing
$listing): Response
    {
        $this->denyAccessUnlessGranted('IS_OWNER', $requestEntity->getListing());

        $requestEntity->setStatus(PurchaseRequest::RENTAL_APPROVED);
        $listing->setStatus(Listing::LISTING_RENTED);

        $em->flush();

        flash()->success('Заявку на продаж успішно підтверджено');
        return $this->redirectToRoute('app_request_show', ['id' => $requestEntity->getId()]);
    }

    #[Route('/requests/{id}/reject', name: 'app_request_reject', methods: ['POST'])]
    #[IsGranted('ROLE_USER')]
    public function reject(PurchaseRequest $requestEntity, EntityManagerInterface $em): Response
    {
        $this->denyAccessUnlessGranted('IS_OWNER', $requestEntity->getListing());

        $requestEntity->setStatus(PurchaseRequest::RENTAL_REJECTED);
        $em->flush();

        flash()->warning('Заявку на продаж відхилено');
        return $this->redirectToRoute('app_request_show', ['id' => $requestEntity->getId()]);
    }
}
```