Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «МАГІСТР»

РОЗРОБКА ТА ДОСЛІДЖЕННЯ ЗАСТОСУНКУ ДЛЯ УПРАВЛІННЯ В РІЕЛТОРСЬКІЙ СФЕРІ

DEVELOPMENT AND RESEARCH OF AN APPLICATION FOR MANAGEMENT IN THE REAL ESTATE INDUSTRY

спеціальність 122 Комп'ютерні науки

освітня програма «Комп'ютерні науки»

Виконав: здобувач вищої освіти
групи КНм-21
Харченко Артем Сергійович
_____
(підпис)
Керівник: к.т.н., доцент
Ліщина Валерій Олександрович
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«___» _____ 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Валерій Олександрович
_____
(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерних наук*
Ступінь вищої освіти: *магістр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *122 Комп'ютерні науки*
Освітня програма: *«Комп'ютерні науки»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Валерій ЛІЩИНА
«14» травня 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА
ДРУГОГО (МАГІСТЕРСЬКОГО) РІВНЯ ВИЩОЇ ОСВІТИ

Харченко Артем Сергійович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «Розробка та дослідження застосунку для управління в ріелторській сфері»
Керівник к.т.н., доцент Ліщина Валерій Олександрович
затверджені наказом закладу вищої освіти від «*14*» *травня* *2025* р. № 255/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи *«05» грудня 2025 р.*
3. Вихідні дані до роботи: _____
_____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):
Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз і вибір засобів проектування, опис функціонального наповнення об'єкта проектування, розробка й обґрунтування системного наповнення, експерементальне дослідження результативності предмету дослідження.
_____
5. Перелік графічного матеріалу:

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз проблематики за темою роботи та постановка завдань дослідження	*Ліщина В. О.*		
Теоретичне дослідження та практична реалізація предмету дослідження	*Ліщина В. О.*		
Експеременталье дослідження результативності предмету дослідження	*Ліщина В. О.*		
Показник запозичень тексту	_____ %		
Інструментальна перевірка	*Кошелюк В. А.*		
Нормоконтроль	*Сачук В. О.*		
Гарант ОПП	*Ліщина В. О.*		

7. Дата видачі завдання *«14» травня 2025 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	*Провести огляд літературних джерел по темі кваліфікаційної роботи*	*до 30.06.2025 р*	
2	*Провести аналіз загальної проблеми і вибір напрямків дослідження*	*до 01.09.2025 р.*	
3	*Розробити функціональну схему роботи програмного продукту*	*до 01.10.2025 р*	
4	*Описати засоби розробки об'єкта проектування*	*до 15.10.2025 р.*	
5	*Практична реалізація об'єкта проектування*	*до 10.11.2025 р.*	
6	*Провести експеременталье дослідження результативності предмету дослідження*	*до 25.11.2025 р.*	
7	*Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру*	*до 05.12.2025 р.*	

Здобувач вищої освіти _____ Артем ХАРЧЕНКО

Керівник роботи _____ Валерій ЛІЩИНА

АНОТАЦІЯ

Харченко А. С. Розробка та дослідження застосунку для управління в ріелторській сфері. Рукопис.

Кваліфікаційна робота магістра ОП «Комп'ютерні науки». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота магістра складається з вступу, 3 розділів, висновків, списку використаних джерел та додатків.

Кваліфікаційна робота магістра присвячена розробці та дослідженню застосунку для управління процесами в ріелторській сфері.

У роботі розроблено концептуальну модель застосунку для підтримки повного життєвого циклу ріелторської угоди – від реєстрації об'єкта нерухомості та формування бази клієнтів до супроводу переговорів, фіксації домовленостей, планування показів і аналітики результативності агентів. Застосунок реалізовано як веборієнтовану систему з адаптивним інтерфейсом, що забезпечує роботу на настільних і мобільних пристроях.

Проведено експериментальне дослідження результативності запропонованого рішення на основі сценарного тестування та порівняльного аналізу з традиційним підходом до роботи ріелтора (електронні таблиці, месенджери, паперові нотатки). Показано, що використання застосунку дозволяє скоротити середній час реєстрації нового об'єкта, зменшити кількість помилок у контактних даних, підвищити прозорість воронки продажів і забезпечити доступ керівництва до актуальної звітності в режимі реального часу.

Ключові слова: інформаційна система, вебзастосунок, ріелторська діяльність, управління нерухомістю, CRM, REST-API, база даних.

ABSTRACT

Artem Kharchenko. Development and research of an application for management in the real estate industry. Manuscript.

Master's thesis in Computer Science. Lutsk National Technical University. Lutsk, 2025.

The master's thesis consists of an introduction, 3 chapters, conclusions, a list of references, and appendices.

The master’s thesis is devoted to the development and research of an application for managing business processes in the real estate domain.

The thesis proposes a conceptual model of an application that supports the full life cycle of a real-estate deal: from property registration and customer relationship management to negotiation support, appointment scheduling, and performance analytics. The application is implemented as a web-based system with a responsive user interface, providing convenient access from both desktop and mobile devices.

An experimental study of the proposed solution is carried out using scenario-based testing and comparative analysis with the traditional workflow of real-estate agents, which relies on spreadsheets, messengers, and paper notes. The results demonstrate that the application reduces the average time required to register a new property, decreases the number of errors in contact data, increases the transparency of the sales funnel, and provides management with up-to-date reports in near real time.

Keywords: information system, web application, real-estate management, real-estate agency, CRM, REST API, database.

ЗМІСТ

ВСТУП

Ринок нерухомості є однією з найбільш динамічних і конкурентних сфер економічної діяльності. Для успішної роботи ріелторських компаній недостатньо лише мати актуальні пропозиції об’єктів – критичного значення набуває оперативне опрацювання запитів клієнтів, ефективна комунікація між агентами, керівництвом та партнерами, прозорий облік етапів кожної угоди та можливість швидко отримати аналітичну інформацію для прийняття управлінських рішень.

На практиці значна частина ріелторів продовжує використовувати фрагментований набір інструментів: електронні таблиці, месенджери, окремі особисті нотатки, файлові сховища. Такий підхід ускладнює контролювання стану воронки продажів, призводить до дублювання даних, втрати історії взаємодії з клієнтами та помилок у документуванні угод. В умовах підвищеної конкуренції на ринку нерухомості та розвитку «proptech»-рішень це знижує ефективність бізнесу й ускладнює масштабування компаній.

Сучасні інформаційні технології дозволяють створювати спеціалізовані застосунки, що об’єднують у єдиному середовищі управління об’єктами нерухомості, клієнтською базою, завданнями ріелторів, календарем показів, документами та аналітикою. Інтеграція таких рішень із вебпорталами оголошень, хмарними сервісами та мобільними пристроями створює умови для побудови наскрізної цифрової екосистеми ріелторської компанії.

Актуальність теми полягає в необхідності розроблення адаптивного застосунку для управління ріелторською діяльністю, який би враховував специфіку локального ринку, підтримував гнучкі бізнес-процеси агенцій різного розміру та забезпечував можливість подальшого розширення функціональності за рахунок підключення аналітичних і мобільних модулів.

Мета роботи – розробити та дослідити вебзастосунок для управління процесами в ріелторській сфері, який забезпечує централізоване зберігання даних про об’єкти й клієнтів, підтримку повного життєвого циклу ріелторської угоди та надання аналітичної інформації для прийняття управлінських рішень.

Для досягнення поставленої мети необхідно вирішити такі завдання:

− проаналізувати сучасний стан цифровізації ріелторського ринку та існуючі інформаційні системи управління нерухомістю;

− дослідити методи та програмні засоби розробки вебзастосунків і CRM-систем, релевантних для ріелторської сфери;

− сформувати вимоги до застосунку, розробити концептуальну та інформаційну моделі предметної області;

− обґрунтувати вибір архітектурного підходу, технологій і інструментів реалізації серверної та клієнтської частин системи;

− реалізувати прототип вебзастосунку для управління об’єктами, клієнтами, угодами та задачами ріелторів;

− розробити методику експериментального дослідження результативності застосунку та провести сценарне тестування;

− виконати обробку та аналіз результатів експерименту, оцінити ефективність запропонованого рішення та визначити напрями подальшого розвитку системи.

Об’єкт дослідження – інформаційні процеси управління діяльністю ріелторської компанії, пов’язані з обліком об’єктів нерухомості, клієнтів та угод.

Предмет дослідження – методи та програмні засоби розробки вебзастосунку для управління ріелторською діяльністю, включно з моделями даних, архітектурними рішеннями та алгоритмами підтримки бізнес-процесів.

Наукова новизна роботи полягає в удосконаленні підходу до моделювання та автоматизованого супроводу життєвого циклу ріелторських угод на основі поєднання CRM-функціоналу, управління задачами та аналітичних інструментів у єдиному вебзастосунку з відкритим REST-інтерфейсом. Запропоновано структуру моделі даних і набір метрик, що дозволяють комплексно оцінювати результативність роботи агентів та ефективність каналів залучення клієнтів.

РОЗДІЛ 1
АНАЛІЗ ПРОБЛЕМАТИКИ ЗА ТЕМОЮ РОБОТИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Огляд і аналіз предметної області проблеми

Ринок нерухомості належить до базових секторів економіки, оскільки оперує об’єктами житлової, комерційної та інфраструктурної забудови, впливає на мобільність населення, інвестиційну активність та міський розвиток. За останнє десятиріччя цей ринок активно трансформується під впливом цифровізації та появи так званих PropTech-рішень (property technologies), які зміщують акцент з «паперового» та телефонного брокерського посередництва до роботи в єдиному цифровому середовищі [1-2].

У сучасних дослідженнях PropTech розглядають як складову загальної цифрової трансформації індустрії нерухомості, що спрямована на підвищення ефективності операцій, прозорості угод, якості клієнтського досвіду та гнучкості бізнес-моделей ріелторських компаній [1-3]. Йдеться не лише про онлайн-портали оголошень, а про комплексні екосистеми: CRM-системи, платформи управління об’єктами та орендою, рішення на основі штучного інтелекту для аналізу ринку, VR/AR-тури, блокчейн-платформи для реєстрації прав власності тощо.

Європейський та український контекст цифровізації ринку нерухомості. У країнах ЄС цифрова трансформація ринку нерухомості вже розглядається як один з ключових напрямів економічної політики. Дослідження [3] підкреслює активне впровадження геоінформаційних систем, технологій електронної ідентифікації, відкритих даних про об’єкти нерухомості, а також інструментів штучного інтелекту й блокчейну для підвищення прозорості та ефективності ринку.

Для України цифровізація нерухомості стала особливо актуальною на тлі воєнних та економічних викликів: потрібна прозора інформація про стан житлового фонду, вартість і доступність об’єктів у різних регіонах; активно

розвивається ринок оренди та інвестицій у житло в безпечніших регіонах; зростає роль відкритих даних і геоінформаційних систем для прийняття рішень щодо відновлення і забудови територій.

Ряд сучасних українських досліджень показує, що аналіз ринку нерухомості вже спирається на великі масиви даних (десятки мільйонів об’єктів із відкритих джерел), геоінформаційний аналіз вторинного ринку та ретроспективу цін і орендних ставок [4-5]. Це підтверджує тезу, що без систематизованих цифрових інструментів ріелтору або невеликій агенції складно орієнтуватися в динамічному ринку.

Основні учасники та бізнес-процеси ріелторської діяльності. У предметній області ріелторської діяльності доцільно виділити кілька груп учасників:

Кінцеві клієнти – покупці та продавці, орендарі та орендодавці.

Ріелтори (агенти) – працюють із клієнтами, формують пропозиції, супроводжують покази та переговори.

Керівництво агенції – контролює воронку продажів, розподіляє ліди, аналізує ефективність агентів.

Партнери – нотаріуси, банки, страхові компанії, девелопери, керуючі компанії.

Життєвий цикл типової угоди складається з послідовності етапів:

- залучення ліда (запиту) від клієнта;
- уточнення потреб (бюджет, локація, тип об’єкта);
- підбір та демонстрація релевантних варіантів;
- організація показів об’єктів;
- переговори щодо умов угоди;
- резервування об’єкта;
- підготовка та погодження пакету документів;
- підписання договору та завершення угоди.

Цю послідовність доцільно відобразити у вигляді лінійної схеми (рис. 1.1), яка слугуватиме основою для подальших діаграм послідовності та сценаріїв use case.

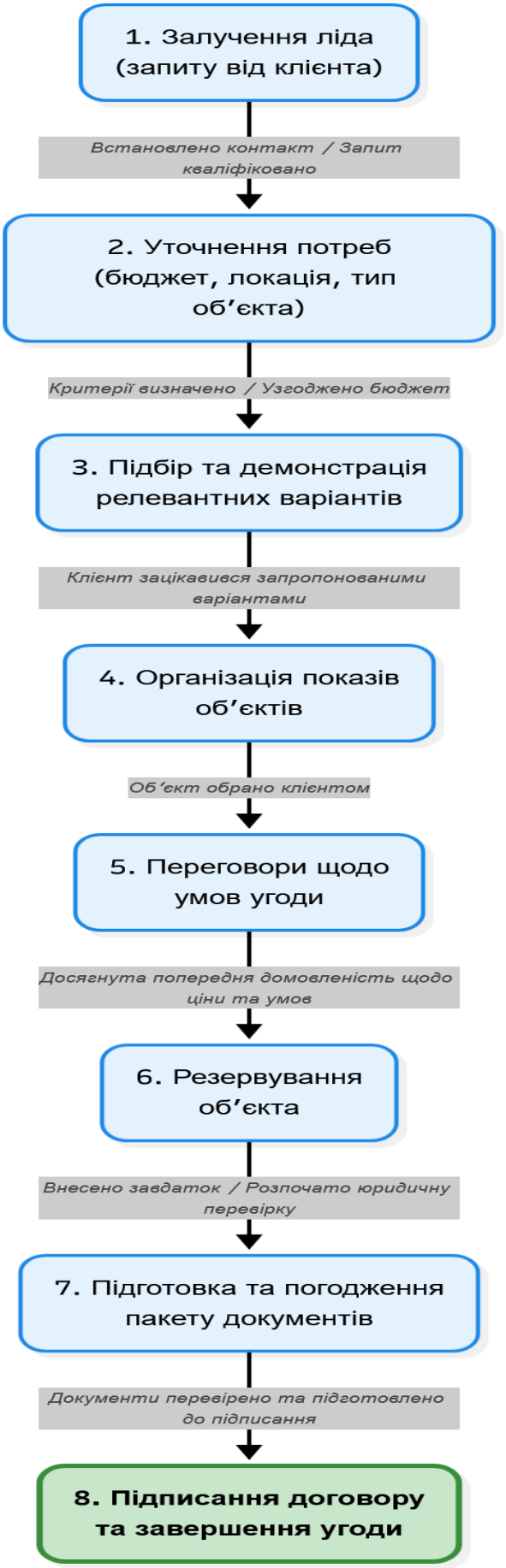

Рисунок 1.1 – Типовий життєвий цикл ріелторської угоди

Проблеми традиційної організації інформаційних процесів. Більшість проблем, на які вказують як практики, так і дослідники, пов’язані з тим, що інформаційні потоки між учасниками ринку залишаються фрагментованими й малостандартизованими. Об’єкти нерухомості фігурують одночасно в кількох базах (Excel-файли, локальні CRM, публічні портали) з різними версіями описів. Історія контактів із клієнтом розірвана між месенджерами, електронною поштою та телефонами. Статус угоди часто фіксується неформально (в особистих нотатках агента) і не доступний керівництву в реальному часі Звіти про активність агентів, ефективність каналів залучення лідів, середню тривалість угод формуються вручну, що є трудомістким і помилковитим.

Дослідження з розробки веборієнтованих систем управління нерухомістю одностайно наголошують, що ручні процеси та неінтегровані інформаційні системи призводять до затримок, високої частки помилок і слабкої прозорості для клієнтів та керівництва [6-8].

Для відображення ситуації у традиційній організації побудуємо діаграму потоків даних (DFD) або спрощену блок-схему (рис. 1.2).

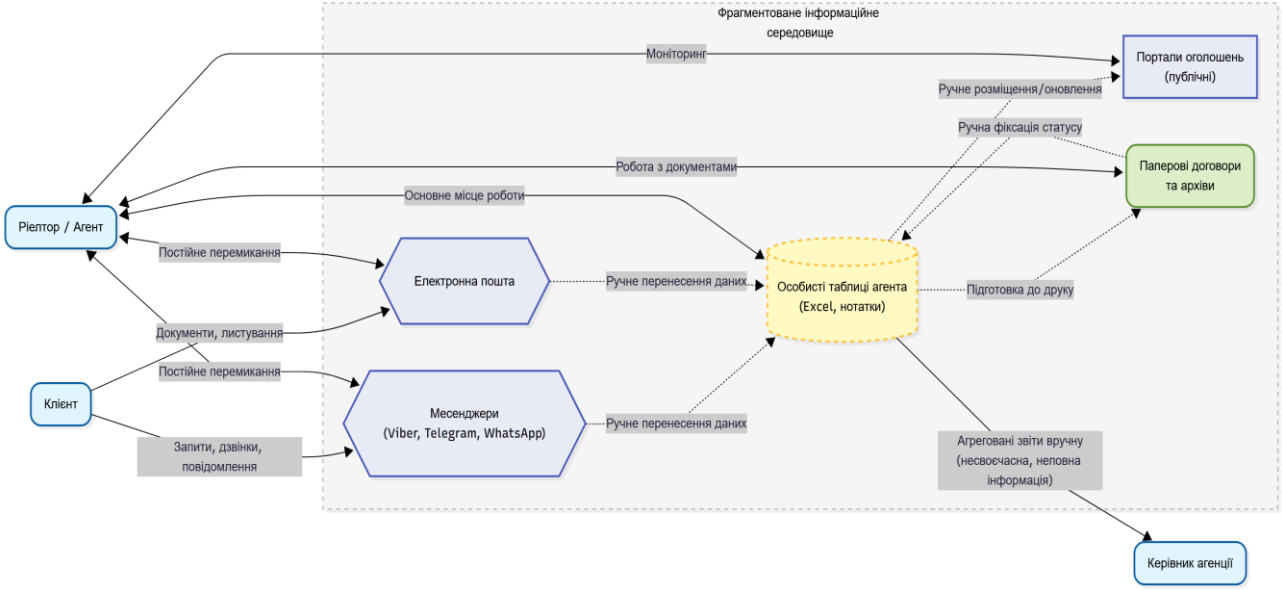

Рисунок 1.2 – Фрагментована схема інформаційних потоків у традиційній роботі ріелтора

Потреба у комплексних цифрових рішеннях для ріелторських компаній. У відповідь на зазначені проблеми на ринку оформилися кілька класів цифрових рішень.

CRM-платформи для ріелторів – фокус на лідах, контактах, воронці продажів.

Системи управління об’єктами та орендою (Property Management Systems, PMS) – фокус на обліку об’єктів, орендних відносинах, платежах.

Аналітичні платформи та BI-рішення – фокус на аналізі ринку, прогнозуванні цін, оцінці портфеля об’єктів [9-10].

За даними галузевих оглядів, CRM-системи стали основним типом програмного забезпечення, що використовується брокерськими компаніями: частка CRM у структурі спеціалізованого ПЗ для ринку нерухомості перевищує 60 %, а їх упровадження дозволяє підвищити конверсію лідів та знизити відтік клієнтів на десятки відсотків [11].

Систематичний огляд використання CRM-платформ у ріелторських агенціях показує, що такі системи покращують персоналізацію сервісу, пришвидшують комунікацію та підвищують задоволеність клієнтів [9].

Водночас у наукових роботах і практичних кейсах звертають увагу на те, що «коробкові» CRM або PMS не завжди добре відображають специфіку локальних ринків, особливо там, де поєднуються: подвійні бази «попит–пропозиція»; складні схеми спільного фінансування та оренди; неоднорідна нормативно-правова база [3-4].

Для невеликих ріелторських компаній це створює нішу для власних вебзастосунків, які точно відображають їх бізнес-процеси, легко інтегруються з обраними порталами оголошень, підтримують адаптивний інтерфейс для роботи «в полі» (з телефона/планшета), можуть розвиватися еволюційно – через додавання нових модулів (аналітика, мобільний застосунок, чат-боти тощо).

1.2 Огляд і аналіз методів та засобів розробки застосунків для ріелторської сфери

Ключова проблема ріелторських компаній полягає не лише у зберіганні даних про об’єкти, а в комплексній підтримці бізнес-процесів: веденні контактів, воронці угод, плануванні показів, інтеграції з каналами залучення лідів, аналітиці. Тому при виборі методів і засобів розробки важливо орієнтуватися на вже накопичений досвід створення CRM-систем для ринку нерухомості та веборієнтованих систем управління нерухомістю.

Класи програмних рішень та їх функціональність. На основі аналізу літератури та галузевих оглядів можна виділити такі класи програмних рішень для ріелторської сфери [11].

Горизонтальні CRM-системи загального призначення орієнтовані на різні галузі; забезпечують базову роботу з лідами, контактами, завданнями; потребують глибокої адаптації під специфіку об’єктів нерухомості (додаткові поля, кастомні воронки, інтеграція з порталами оголошень).

Спеціалізовані CRM для ріелторів (Real Estate CRM) вже містять модулі обліку об’єктів, воронку продажів, інтеграції з порталами; часто мають готові звіти за агентами, каналами, типами об’єктів; орієнтовані на масовий ринок і не завжди дозволяють гнучко змінювати бізнес-логіку.

Property Management Systems (PMS)/Residential Property Management фокусуються на управлінні об’єктами й орендними відносинами, оплатою послуг, комунікацією з орендарями; використовуються або керуючими компаніями, або великими власниками портфеля об’єктів; у наукових роботах їх розглядають як веборієнтовані системи з модулем звітності, календарем робіт, фінансовими підсистемами [6-8].

Аналітичні та прогнозні платформи застосовують Big Data та методи машинного навчання для прогнозування цін, орендних ставок, заповнюваності; інтегруються з CRM та PMS, але рідко замінюють їх повністю [5-10].

У таблиці 1.1 узагальнено ключові типи рішень та їх типову функціональність з погляду ріелторської агенції.

Таблиця 1.1 – Типи програмних рішень для ріелторської діяльності

Тип рішення	Основні функції	Типові користувачі	Обмеження для невеликих агенцій
Горизонтальна CRM	Ліди, контакти, задачі, базові звіти	Продажі в різних галузях	Немає моделі об’єкта, складна адаптація
Реєлторська CRM	Об’єкти, воронка угод, інтеграція з порталами, e-mail/SMS-комунікація	Ріелторські агенції	Закритий код, фіксований набір бізнес-процесів
PMS / пропертi-менеджмент	Облік об’єктів, оренда, платежі, заявки орендарів	Керуючі компанії, девелопери	Складність, надлишковість функцій
Аналітичні платформи	Збір ринкових даних, прогнозування цін, BI-звіти	Аналітики, керівництво	Потребують інтеграції з CRM/PMS

Архітектурні підходи та веб-технології в наукових розробках. Окрему групу становлять наукові роботи, в яких описано створення прототипів або повноцінних систем управління нерухомістю. Їх спільні риси: використання веборієнтованої архітектури «браузер-сервер» (B/S); застосування реляційних СУБД для зберігання даних про об’єкти, орендарів, транзакції; реалізація модулів звітності й оцінювання системи за показниками якості (часто – у відповідності до ISO/IEC 25010) [6-8].

Наприклад, у сучасних працях пропонуються хмарні PMS для управління об’єктами й досвіду орендарів із вебінтерфейсом і модулем оцінки якості за ISO 25010 [6-8]; web-based Real Estate Management System (REMS), орієнтована на взаємодію продавців і покупців, де особлива увага приділяється прозорості й безпеці угод [6-7]; системи управління житловими комплексами на основі J2EE, ASP.NET та інших платформ, що використовують модель B/S, модульну структуру й сучасні вебфреймворки [8].

Ці роботи показують де-факто стандарт: вебзастосунок із чітко виділеними рівнями – база даних, логіка та інтерфейс, з можливістю розгортання в хмарній інфраструктурі та доступом через браузер із різних пристроїв.

Цифрова трансформація та вимоги до архітектури ріелторських систем. Сучасні огляди цифрової трансформації ринку нерухомості відзначають, що найуспішніші компанії використовують єдине джерело правди щодо об’єктів і клієнтів. Будують API-орієнтовану архітектуру, яка дозволяє інтегрувати CRM/PMS із порталами оголошень, фінансовими та маркетинговими сервісами. Застосовують аналітику в реальному часі та інструменти автоматизації (нагадування, тригери, робочі процеси) [12].

Дослідження також підкреслюють зростання ролі штучного інтелекту (аналіз ринку, рекомендації об’єктів, прогнозування цін), блокчейну (безпечна реєстрація угод, токенізація нерухомості), інтернету речей (IoT) і цифрових двійників будівель для моніторингу стану об’єктів [3].

Для невеликих ріелторських агенцій повноцінне впровадження усіх перелічених технологій може бути завеликим за бюджетом, але архітектура нових систем уже повинна враховувати можливість їх подальшої інтеграції. Це означає чіткий поділ на backend API та frontend-клієнти (веб, мобільний застосунок, інтеграції). Використання стандартизованих форматів обміну даними (JSON/REST, потенційно – GraphQL). Застосування масштабованих СУБД (PostgreSQL, іноді – поєднання з NoSQL для логів та телеметрії).

Саме такі архітектурні підходи варто використати при розробці застосунку для управління в ріелторській сфері.

Узагальнення вимог до програмних засобів та обґрунтування вибору стеку. Узагальнюючи результати огляду, можна сформулювати ключові вимоги до програмних засобів, що реалізують функції управління ріелторською діяльністю. Централізоване зберігання даних про об’єкти, клієнтів, угоди, покази, документи. Підтримка бізнес-процесів агенції через налаштовану воронку продажів, задачі та календар подій. Зручний вебінтерфейс із адаптацією під мобільні пристрої. Інтеграція з зовнішніми системами (портали оголошень, пошта, календар, BI-інструменти). Модуль звітності та аналітики, який дозволяє оцінювати ефективність роботи агентів і каналів залучення клієнтів. Можливість

масштабування та подальшого розвитку, включно з додаванням аналітичних модулів на основі AI/ML.

Беручи до уваги практику, описану в сучасних дослідженнях і галузевих гайдах [6-8; 10-12], доцільним є вибір веборієнтованого підходу з клієнт–серверною архітектурою, REST-API на базі Python/Django та реляційною СУБД PostgreSQL, а також фронтендом на React/TypeScript.

1.3 Постановка завдання на кваліфікаційну роботу

З урахуванням проведеного аналізу предметної області та існуючих програмних рішень, завдання кваліфікаційної роботи магістра формулюються таким чином:

Провести систематизований огляд літературних джерел, стандартів та програмних рішень, присвячених цифровим платформам для управління нерухомістю та CRM-системам у сфері послуг.

Проаналізувати бізнес-процеси ріелторської компанії, виділити основні сутності та їх взаємозв’язки (об’єкти, клієнти, агенти, угоди, покази, задачі, документи).

Розробити концептуальну модель застосунку та інформаційну модель бази даних для підтримки основних процесів ріелторської діяльності.

Обґрунтувати вибір архітектури та стеку технологій серверної й клієнтської частин, розробити загальну структурну схему програмного продукту.

Реалізувати прототип вебзастосунку, що забезпечує:

– ведення бази об’єктів нерухомості;

– ведення бази клієнтів та історії взаємодії з ними;

– управління угодами та етапами воронки продажів;

– планування та відображення календаря показів;

– формування основних звітів для керівництва.

Розробити методику експериментального дослідження, що включає сценарне тестування застосунку та порівняння з традиційною організацією роботи.

Провести обробку й аналіз результатів експерименту, сформулювати висновки щодо ефективності розробленого застосунку та можливих напрямів його розвитку.

РОЗДІЛ 2
ТЕОРЕТИЧНЕ ОБҐРУНТУВАННЯ ТА ПРАКТИЧНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ ДЛЯ УПРАВЛІННЯ В РІЕЛТОРСЬКІЙ СФЕРІ

2.1 Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання

Проєктований застосунок повинен підтримувати повний цикл роботи ріелторської компанії: від введення і супроводу об’єктів нерухомості та клієнтів до управління угодами, планування показів і формування звітності. Усі ці процеси вимагають одночасного доступу кількох користувачів, роботи через вебінтерфейс та можливості подальшого розширення функціональності. Тому базовим рішенням є використання веборієнтованої клієнт-серверної архітектури з чітким розподілом на рівні представлення, бізнес-логіки та даних.

З погляду стилю архітектури аналізуються три варіанти. Класичний моноліт передбачає єдине застосування з тісно пов’язаними модулями, що спрощує розроблення та розгортання на початкових етапах, але ускладнює підтримку й еволюцію системи при зростанні вимог. Мікросервісна архітектура, навпаки, розбиває систему на набір незалежних сервісів з власними базами даних, але потребує розвиненої інфраструктури, окремих команд супроводу й підвищених компетенцій у галузі DevOps. Проміжний підхід, модульний моноліт, дозволяє будувати систему як один задеплойований застосунок, але з чітко виділеними внутрішніми модулями, які мають власні зони відповідальності, інтерфейси й можуть в майбутньому бути винесені в окремі сервіси.

Для умов магістерського проєкту та типової ріелторської агенції, де навантаження помірне, але важлива зрозумілість і простота супроводу, доцільно обрати модульний моноліт. Це забезпечує баланс між технологічною складністю та можливістю розвитку системи. У рамках такого підходу застосунок будується як трирівнева система, де верхній рівень відповідає за користувацький інтерфейс

у браузері, середній – за бізнес-логіку та REST-інтерфейс, а нижній – за надійне зберігання даних у реляційній базі [13].

Узагальнену архітектуру системи можна подати у вигляді трирівневої схеми. На клієнтському рівні працює вебзастосунок, що виконується у браузері користувача і реалізований засобами бібліотеки React. На рівні застосунку функціонує сервер, розроблений на базі Django та Django REST Framework, який приймає HTTP-запити, виконує бізнес-логіку, звертається до бази даних і формує відповіді у форматі JSON. Рівень даних представлений СУБД PostgreSQL, де зберігаються всі сутності предметної області (рис. 2.1).

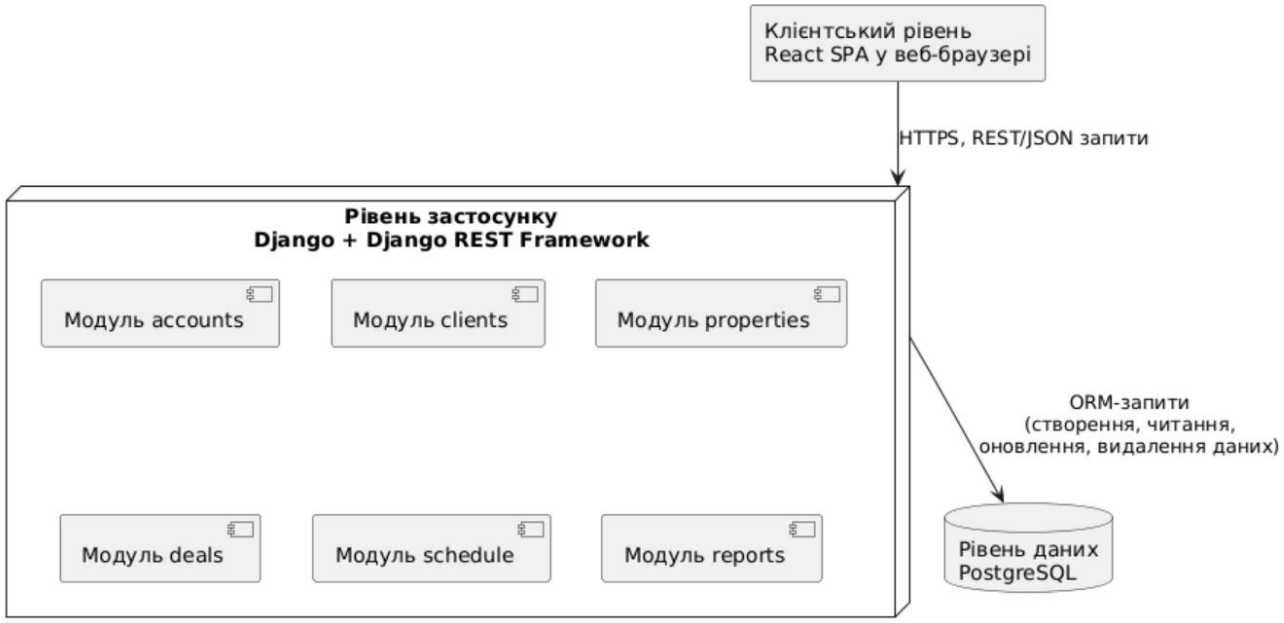

Рисунок 2.1 – Узагальнена трирівнева архітектура застосунку

Така архітектура дозволяє розмежувати відповідальність між рівнями, спростити тестування й налагодження, а також забезпечити можливість розширення системи шляхом додавання нових модулів без порушення цілісності.

Серверна частина системи реалізує всі ключові бізнес-процеси ріелторської діяльності: створення й редагування об’єктів, реєстрацію клієнтів, відкриття та супровід угод, планування показів, формування звітності. Для реалізації цього шару обрано стек Python, Django та Django REST Framework.

Фреймворк Django забезпечує високорівневий підхід до розробки вебзастосунків, надаючи розвинуту систему моделей даних, механізм міграцій схем, централізоване налаштування, засоби аутентифікації користувачів та гнучку маршрутизацію. Завдяки вбудованій ORM бізнес-логіка оперує об’єктами, а не сирими SQL-запитами, що зменшує ризик помилок та спрощує підтримку.

Розширення Django у вигляді Django REST Framework дозволяє організувати REST-інтерфейси за рахунок використання серіалізаторів, наборів представлень (ViewSet) та механізмів фільтрації, пагінації і авторизації. Для кожної доменної сутності створюється модель, серіалізатор і набір REST-ендпоінтів, які відповідають на запити клієнтів. Наприклад, сутність Property має модель Property, серіалізатор PropertySerializer і набір REST-представлень PropertyViewSet, що дозволяє здійснювати повний набір CRUD-операцій.

Послідовність обробки типової операції читання даних, наприклад перегляду списку об’єктів, показано на рисунку 2.2.

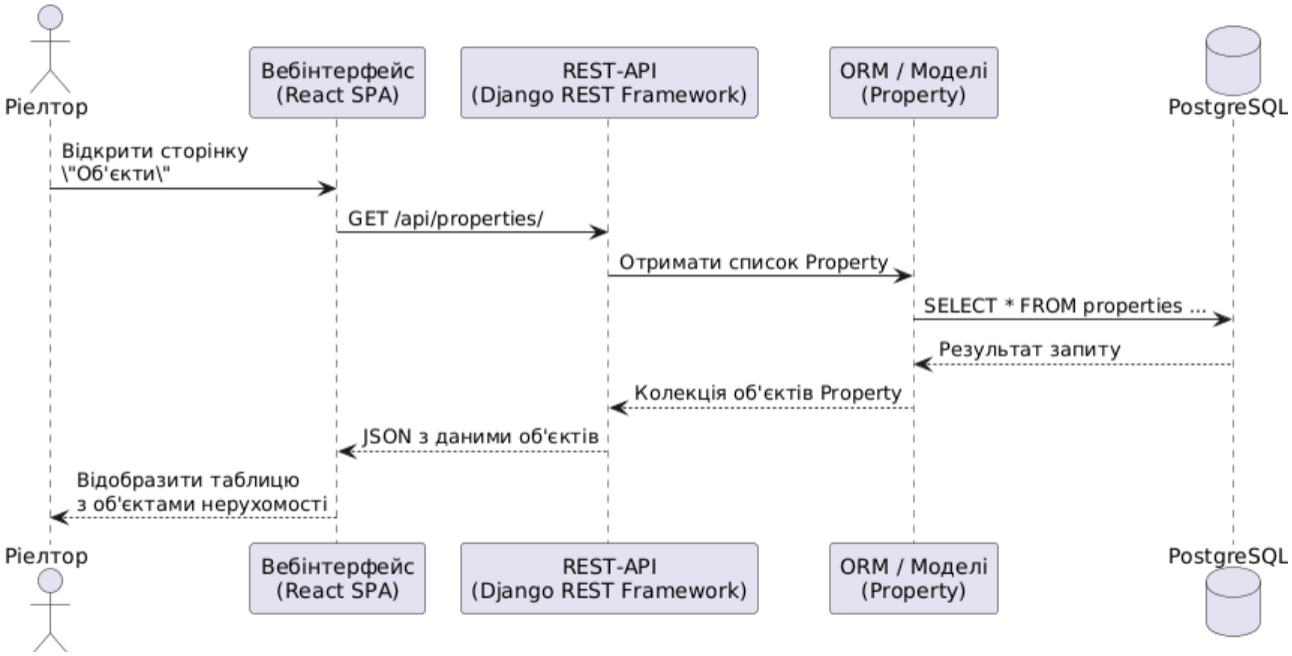

Рисунок 2.2 – Послідовність взаємодії при отриманні списку об’єктів

Ця схема демонструє типовий цикл запит–відповідь, що повторюється для різних сутностей системи.

Проєктування рівня даних і вибір СУБД. Рівень даних є основою системи, оскільки саме тут зберігається структурована інформація про клієнтів, об’єкти, угоди, покази, завдання та документи. Вимоги до цього рівня включають підтримку транзакцій, високий рівень цілісності, можливість складного фільтрування, індексування за різними атрибутами й перспективу використання геопросторових розширень [14].

Для реалізації цих вимог обрано СУБД PostgreSQL. Вона забезпечує повну ACID-сумісність, підтримує різні типи індексів і надає можливість розширення за допомогою додаткових модулів, зокрема PostGIS для роботи з геоданими. Це створює основу для подальшого впровадження функцій пошуку об’єктів у заданому радіусі, аналізу районів тощо (рис. 2.3).

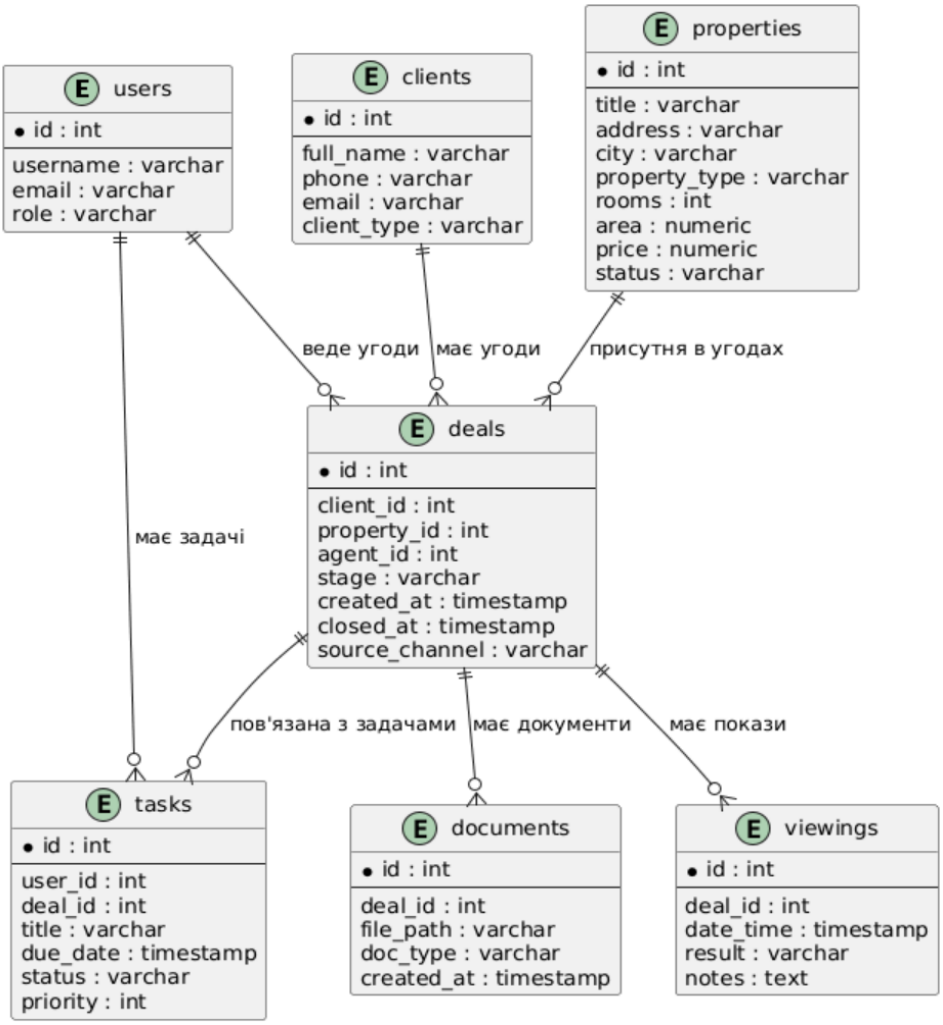

Рисунок 2.3 – ER-діаграма основних сутностей бази даних

Інформаційна модель системи включає кілька базових сутностей, які відповідають реальним об’єктам предметної області. Сутність Client зберігає контактні дані клієнта, його роль (покупець, продавець, орендар, орендодавець) та додаткові характеристики. Сутність Property описує об’єкт нерухомості, включаючи назву, адресу, місто, тип, площу, кількість кімнат, вартість та статус. Сутність Deal представляє окрему угоду між клієнтом і об’єктом, пов’язану з відповідальним ріелтором і поточним етапом воронки продажів. Додаткові сутності Viewing, Task і Document описують відповідно покази об’єктів, задачі для агентів та пов’язані документи.

Взаємозв’язки між цими сутностями відображено на ER-діаграмі бази даних. Запропонована схема дозволяє забезпечити цілісність даних і логічну відповідність між базовими об’єктами предметної області.

Вибір технологій клієнтської частини. Користувачами системи є ріелтори, менеджери й адміністратори, які працюють переважно через веббраузер. Для забезпечення швидкої та інтерактивної роботи інтерфейсу доцільно використати односторінковий застосунок (SPA) на базі бібліотеки React і мови TypeScript.

React дозволяє будувати інтерфейс як композицію незалежних компонентів, кожен з яких відповідає за окрему частину екрану: навігаційну панель, таблицю об’єктів, форму редагування, канбан-дошку угод, календар тощо. Такий підхід полегшує повторне використання елементів інтерфейсу, тестування окремих компонентів, а також поетапне розширення системи. TypeScript забезпечує статичну типізацію даних, які передаються між клієнтом і сервером, що зменшує кількість помилок під час розроблення та спрощує рефакторинг.

Структуру клієнтської частини можна показати у вигляді діаграми компонентів.

На цій діаграмі відображено логічну побудову інтерфейсу: центральний компонент App керує навігацією і макетом, тоді як кожна сторінка взаємодіє з сервером через модуль apiClient (рис. 2.4).

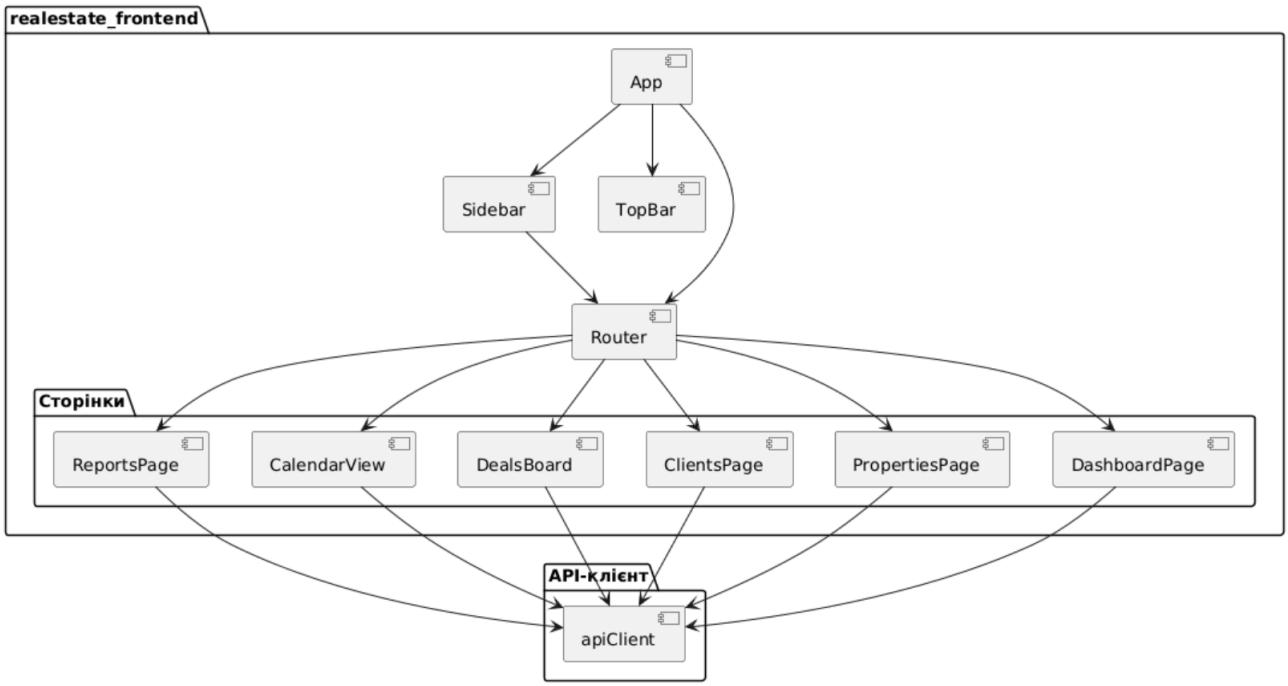

Рисунок 2.4 – Структура клієнтського застосунку

Нефункціональні вимоги та схема розгортання. Система повинна забезпечувати задовільну швидкодію при одночасній роботі кількох користувачів, бути надійною з погляду збереження даних, захищеною від несанкціонованого доступу та зручною для супроводу. Ці вимоги враховано в архітектурних рішеннях [15].

Модульний моноліт на Django і PostgreSQL забезпечує прийнятну продуктивність за рахунок єдиного розгортання й мінімальних накладних витрат на міжпроцесну взаємодію. Дотримання принципів ACID на рівні СУБД гарантує цілісність транзакцій при роботі з угодами та пов’язаними сутностями. Вбудовані механізми захисту Django дозволяють протидіяти основним вебзагрозам, таким як SQL-ін’єкції, XSS та CSRF, а використання HTTPS додає захист каналу передачі даних.

Логічна схема розгортання системи включає вебсервер, сервер застосунку та сервер бази даних.

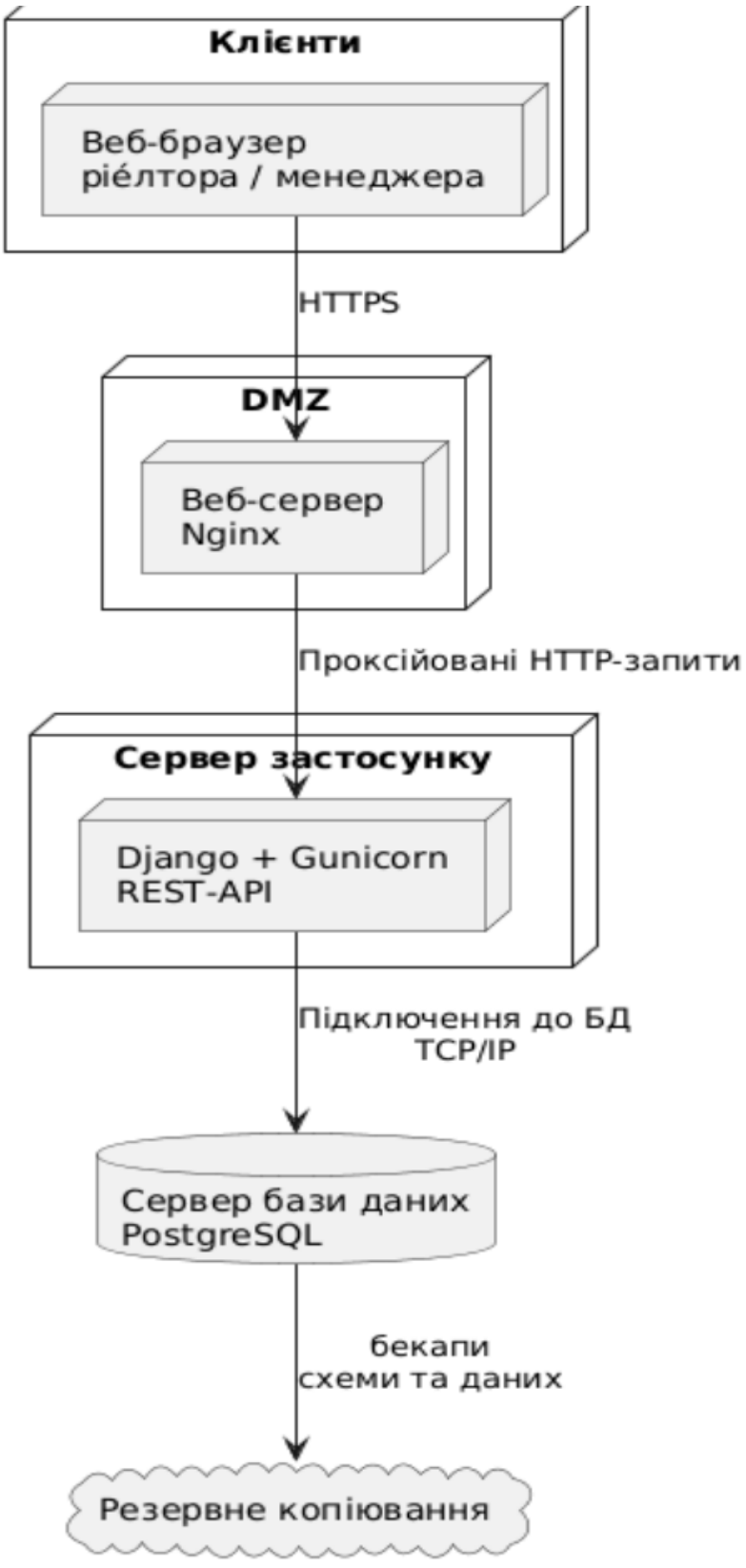

Рисунок 2.5 – Логічна схема розгортання системи

На діаграмі показано, що всі клієнтські запити проходять через вебсервер, який виконує роль реверс-проксі й термінатора TLS, після чого передаються до

серверу застосунку. Останній взаємодіє з базою даних у внутрішній мережі, а сама база даних регулярно резервується.

2.2 Практична реалізація об’єкта проєктування

Практична реалізація застосунку включає конкретне моделювання предметної області на рівні програмних класів, побудову серверної частини з REST-інтерфейсами, створення клієнтського інтерфейсу та реалізацію механізмів звітності й безпеки. У цьому підрозділі розглянуто структуру програмного забезпечення та ключові аспекти реалізації [16].

Серверна частина реалізована як Django-проєкт з низкою додатків, кожен з яких відповідає за певну підсистему. Центральною є конфігурація проєкту, де визначаються підключені модулі, параметри бази даних, налаштування безпеки та загальні middleware. Окремі додатки відповідають за облікові записи користувачів, клієнтів, об’єкти нерухомості, угоди, розклад показів і звітність.

Структуру серверної частини демонструє діаграма пакетів (рис. 2.6).

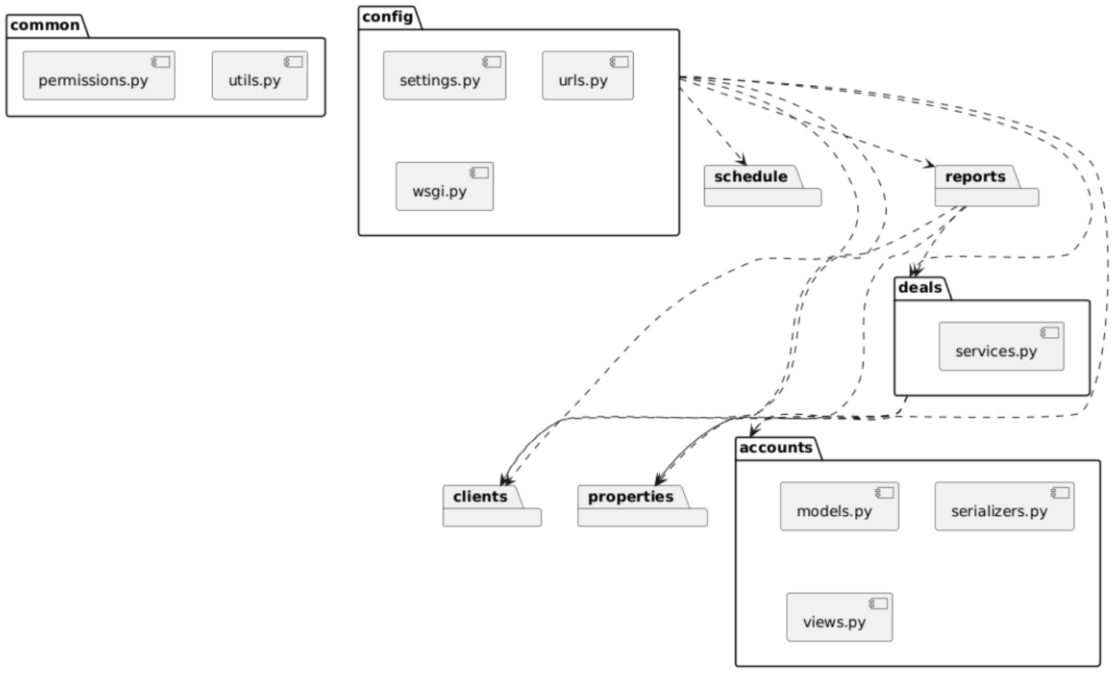

Рисунок 2.6 – Структура серверної частини Django-проєкту

На діаграмі видно, що модуль deals залежить від даних клієнтів, об’єктів і користувачів, а модуль reports використовує інформацію з усіх доменних підсистем.

Реалізація доменної моделі. Доменною моделлю системи є набір класів Django, які відповідають сутностям предметної області. Вони відображують структуру бази даних, але також можуть містити додаткові методи, пов’язані з бізнес-логікою, наприклад розрахунок тривалості угоди або перевірку допустимості переходу між етапами.

UML-діаграма класів відображає структуру основних доменних сутностей.

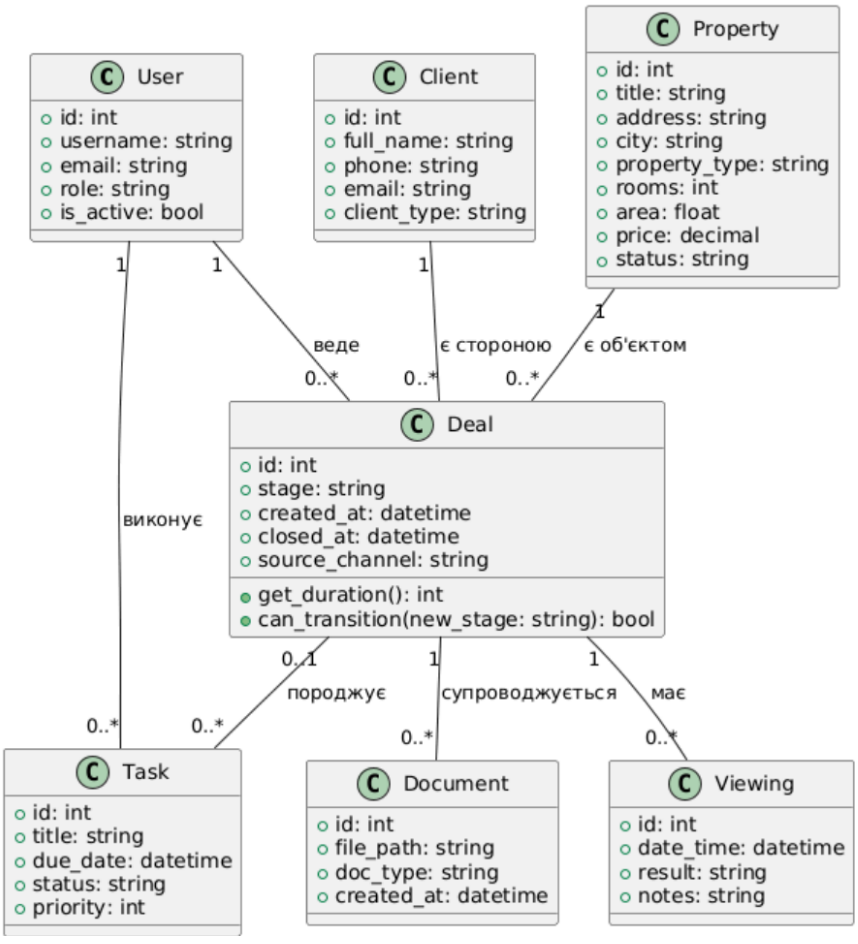

Рисунок 2.7 – UML-діаграма класів доменної моделі

На основі цієї діаграми будуються конкретні файли models.py у відповідних додатках Django. Наприклад, модель Property реалізується у вигляді

класу з полями для назви, адреси, міста, типу, площі, кількості кімнат, вартості та статусу, а також автоматично керованими полями дати створення й оновлення.

Бізнес-логіка системи розподілена між методами моделей, сервісними класами та REST-представленнями. Типовий сценарій роботи з даними передбачає, що клієнтський застосунок формує HTTP-запит до певного ресурсу, серверна частина перевіряє права доступу, виконує необхідні дії з базою даних і повертає результат у форматі JSON.

На прикладі процесу створення угоди та планування показу можна розглянути увесь ланцюг викликів. Ріелтор заповнює форму угоди у вебінтерфейсі, обираючи клієнта, об’єкт і вказуючи джерело звернення. Після відправлення форми клієнт надсилає запит POST до ендпоінта створення угоди. ViewSet на стороні сервера викликає серіалізатор, який валідує вхідні дані, а потім створює об’єкт Deal. Додатково сервісний клас може автоматично встановити початковий етап воронки, додати першу задачу для агента або сформувати нагадування. Далі, при плануванні показу, клієнтський застосунок надсилає запит POST до ендпоінта створення Viewing, де фіксуються дата, час і пов’язана угода [17].

Цей сценарій показано на діаграмі послідовності.

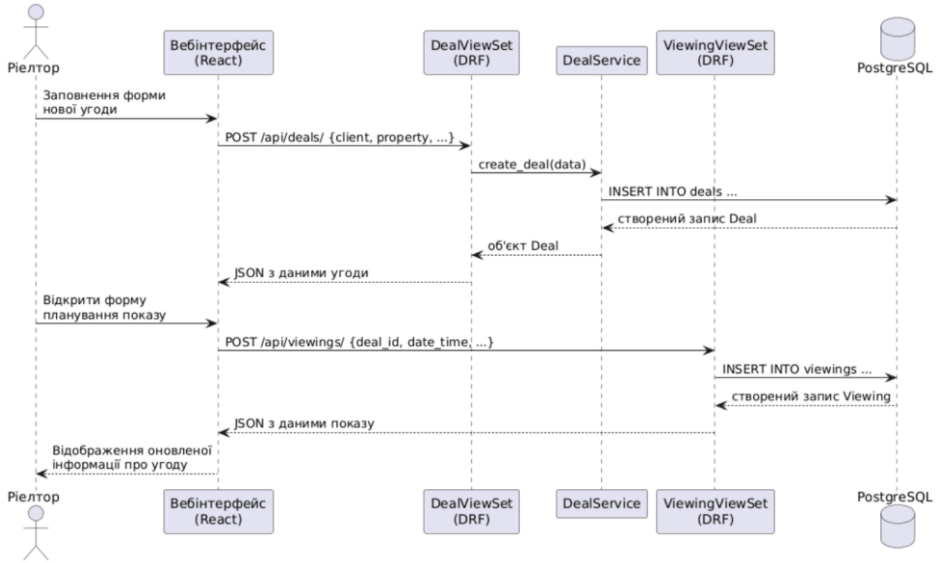

Рисунок 2.8 – Діаграма послідовності «Створення угоди та планування показу»

У коді такі сценарії реалізуються через класи ViewSet у Django REST Framework, серіалізатори й допоміжні сервіси, які інкапсулюють складніші операції.

Реалізація клієнтського інтерфейсу. Клієнтський інтерфейс має забезпечити зручну роботу ріелтора з об’єктами, клієнтами, угодами та розкладом. Він організований у вигляді набору сторінок, між якими користувач перемикається за допомогою бокового меню. Кожен екран поєднує таблиці даних, форми вводу та інтерактивні елементи, які звертаються до REST-API.

Сторінка «Об’єкти» (рис. 2.9) є центральним інструментом для роботи з базою нерухомості. У верхній частині розташовано панель фільтрів, де користувач може уточнювати місто, тип об’єкта, діапазон цін та статус. Основну частину займає таблиця з переліком об’єктів, де в кожному рядку відображуються назва, адреса, тип, ціна та поточний статус. Справа або у верхньому куті розташовуються дії для додавання нового об’єкта, редагування й видалення.

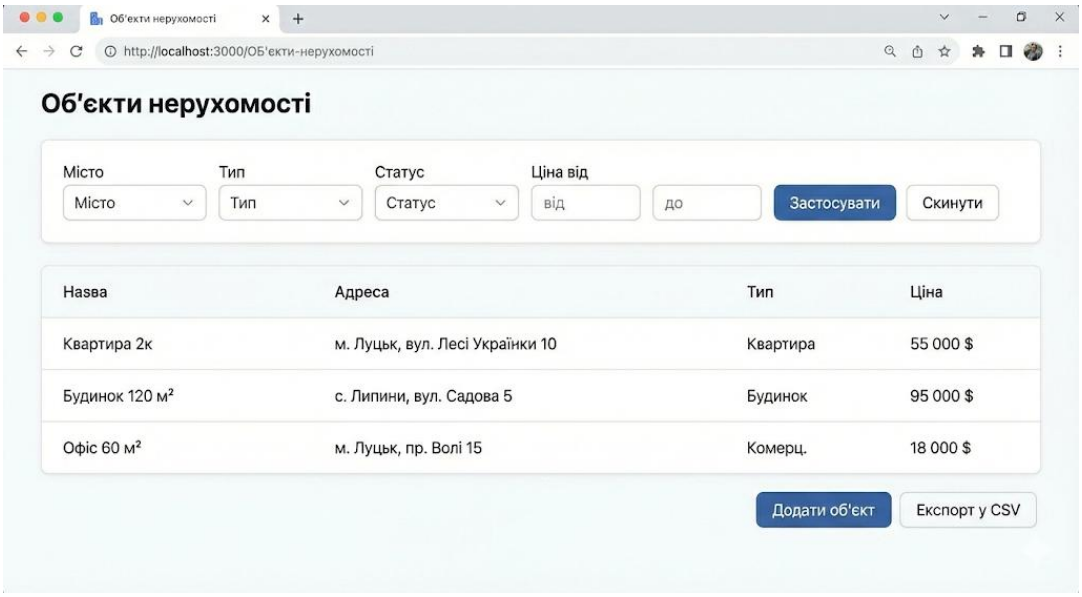

Рисунок 2.9 – Макет сторінки «Об’єкти нерухомості»

У React такий екран реалізується як компонент PropertiesPage, який при монтуванні звертається до apiClient, отримує список об’єктів, зберігає його у

стані та відображає у вигляді таблиці. Зміна фільтрів призводить до повторного запиту з передачею параметрів.

Сторінка «Угоди» (рис. 2.10) відображає стан воронки продажів у вигляді канбан-дошки. Кожна колонка відповідає певному етапу (наприклад лід, покази, переговори, резерв, завершені), а всередині знаходяться картки угод. Переміщення картки між колонками змінює етап угоди й викликає відповідний запит до API.

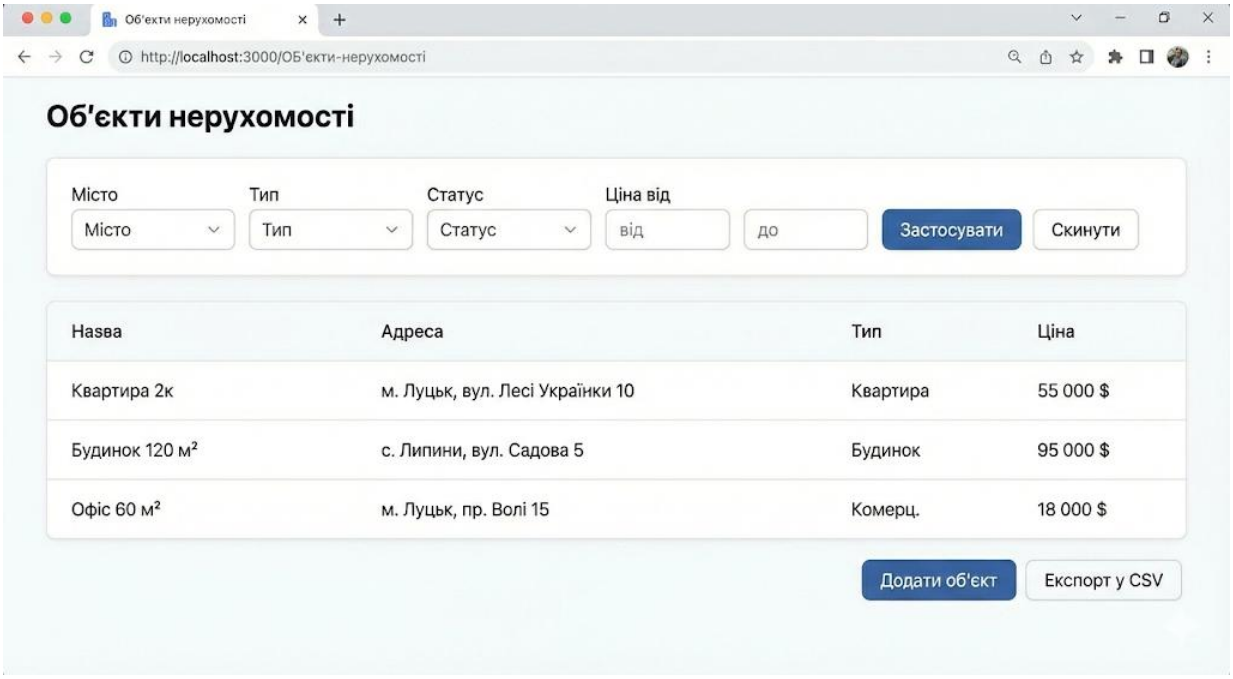

Рисунок 2.10 – Канбан-дошка для управління угодами

Цей макет відображає логіку управління угодами й дозволяє швидко оцінити, на якому етапі знаходиться кожен клієнт.

Сторінка «Календар» (рис. 2.11) показів відтворює події у форматі календаря. Користувач може перемикатися між переглядом місяця, тижня або дня, фільтрувати події за агентами чи типами об’єктів.

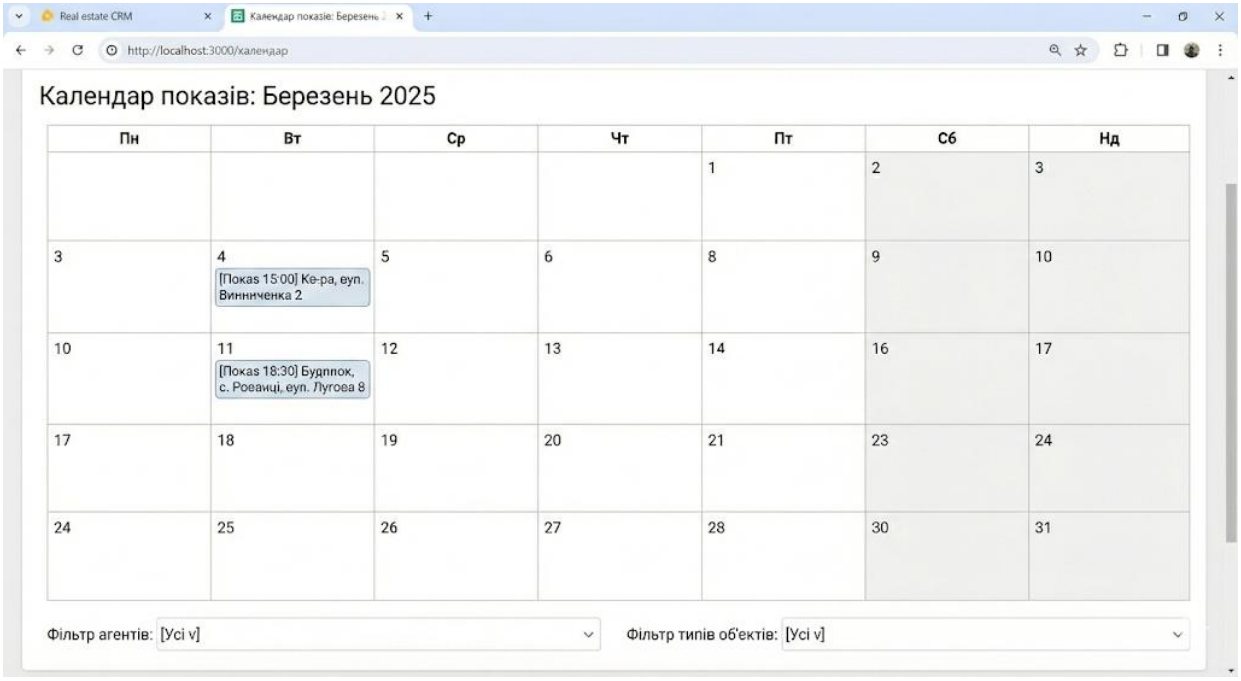

Рисунок 2.11 – Екран календаря показів об’єктів

Таким чином клієнтський інтерфейс реалізує зрозумілу й візуально наочну модель роботи ріелтора в системі.

Модуль звітності та аналітики. Окрему роль у застосунку відіграє модуль звітності. Саме він дозволяє керівництву ріелторської компанії оцінювати ефективність роботи агентів, становище активних угод і виявляти вузькі місця у воронці продажів. На сервері реалізовано набір сервісів, які формують агреговані показники: кількість угод за етапами, кількість угод, закритих кожним агентом, середня тривалість угоди, кількість нових об’єктів і клієнтів за період.

На стороні клієнта ці дані візуалізуються у вигляді діаграм. Однією з ключових є діаграма конверсії угод між етапами воронки, яка показує, скільки угод знаходиться на кожному етапі за вибраний інтервал часу (рис. 2.12).

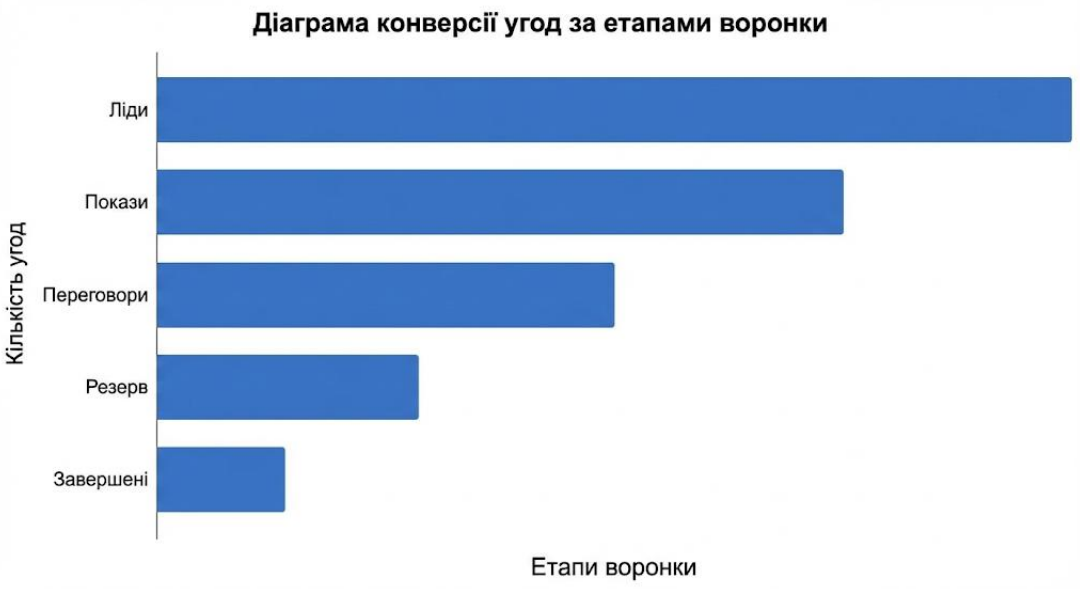

Рисунок 2.12 – Діаграма конверсії угод за етапами воронки

У даному прикладі товщина стовпчиків умовно показує зменшення кількості угод у міру просування по воронці, що дозволяє виявити, де саме відбувається найбільша втрата клієнтів.

Модель безпеки та контроль доступу. Оскільки система оперує конфіденційною інформацією, важливою частиною реалізації є організація аутентифікації, авторизації та контролю доступу. У застосунку передбачено три основні ролі: адміністратор, менеджер і ріелтор. Адміністратор має повний доступ до конфігурації, користувачів і довідників, може переглядати всі угоди та звіти. Менеджер має повний доступ до об’єктів, клієнтів і угод, але обмежений у зміні налаштувань системи. Ріелтор працює переважно зі своїми угодами і завданнями, має можливість переглядати, створювати й змінювати об’єкти та клієнтів у межах наданих прав.

Цю модель доступу можна подати у вигляді матриці ролей і ресурсів.

Модель ролей і прав доступу в системі

Роль	Користувачі	Об'єкти	Клієнти	Угоди	Звіти
Адмін	чит/запис	чит/запис	чит/запис	чит/запис	чит/запис
Менеджер	читання	чит/запис	чит/запис	чит/запис	читання
Ріелтор	власний профіль	чит/частк.	чит/частк.	чит/запис своїх	обмежений

Рисунок 2.13 – Модель ролей і прав доступу в системі

У програмній реалізації ця модель відображається через систему дозволів Django, власні класи пермисій у Django REST Framework і фільтрацію запитів відповідно до ролі поточного користувача.

РОЗДІЛ 3
ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ЗАСТОСУНКУ ДЛЯ УПРАВЛІННЯ В РІЕЛТОРСЬКІЙ СФЕРІ

3.1 Постановка експерименту та методика проведення дослідження

Метою експериментального дослідження є кількісне та якісне оцінювання ефективності розробленого застосунку для управління в ріелторській сфері порівняно з традиційним підходом, що ґрунтується на використанні розрізнених інструментів (електронних таблиць, месенджерів, телефонних дзвінків, неструктурованих файлів). Предметом дослідження виступає процес виконання типових операцій ріелтора, а саме реєстрації об’єктів, роботи з клієнтами, супроводу угод, планування показів та формування звітності. Об’єктом є інформаційна технологія комплексного управління діяльністю ріелторської компанії, реалізована у вигляді веб-застосунку.

У рамках дослідження формулюється припущення, що використання розробленого застосунку дозволяє зменшити час виконання рутинних операцій, скоротити кількість дій, які повинен виконати ріелтор для досягнення того самого результату, а також знизити кількість помилок, пов’язаних з некоректним введенням або втратою даних. Додатково перевіряється гіпотеза про підвищення суб’єктивної зручності роботи за рахунок структурованого інтерфейсу, наявності воронки угод, календаря показів і модуля звітності.

Для коректної постановки експерименту визначаються чіткі критерії оцінювання. В якості основних обрано час виконання операції, кількість елементарних дій користувача (кліків, введень полів) до досягнення заданого результату, кількість допущених помилок за фіксовану кількість повторень сценарію, а також інтегральну оцінку зручності інтерфейсу, яку дають користувачі за п’ятибальною шкалою. Вимірювання проводяться окремо для традиційного підходу і для роботи з розробленим застосунком, після чого обчислюються відносні зміни показників.

Методика дослідження передбачає проведення серії експериментів на групі користувачів, до якої входять ріелтори та співробітники, що мають базовий досвід роботи з електронними таблицями та веб-інтерфейсами. Кожному учаснику пропонується виконати однаковий набір сценаріїв спочатку у традиційний спосіб, а потім за допомогою розробленого застосунку. Щоб мінімізувати вплив порядку виконання, етапи можуть змінюватися місцями між різними учасниками, однак у дипломній роботі результати подаються вже в усередненому вигляді.

Структуру експериментального дослідження подано у вигляді схеми, що показує взаємозв’язок між підготовчим етапом, виконанням сценаріїв і аналізом результатів (рис. 3.1).

Рисунок 3.1 – Структура експериментального дослідження

Для фіксації часу використовується вбудований секундомір або спеціальне програмне забезпечення екранного моніторингу. Кількість дій рахується за спостереженнями дослідника та журналами подій у системі. Помилки визначаються як ситуації, коли операція не була завершена успішно (наприклад, неправильне збереження даних, втрата запису, некоректно заповнені поля, відмова користувача від продовження через складність процесу). Оцінка зручності збирається у вигляді анкет із короткими запитаннями щодо прозорості логіки інтерфейсу, швидкості доступу до потрібних функцій, візуальної зрозумілості воронки угод та календаря.

Усі експерименти проводяться в однаковому програмно-апаратному середовищі, що дає змогу виключити вплив технічних факторів. Характеристики середовища наведено у таблиці 3.1.

Таблиця 3.1 – Програмно-апаратне середовище експериментального дослідження

Параметр	**Значення**
Операційна система	Windows 10 Pro / Windows 11 Pro
Центральний процесор	Intel Core i5 (4 ядра, 8 потоків) або аналогічний
Оперативна пам’ять	8–16 ГБ
Браузер	Google Chrome (актуальна версія)
СУБД	PostgreSQL, розгорнута на локальному сервері
Сервер застосунку	Django + Gunicorn, розміщені на тому ж вузлі
Локальна мережа	Підключення через Ethernet або стабільний Wi-Fi

Експеримент охоплює набір сценаріїв, що відтворюють типові дії ріелтора протягом робочого дня. Кожен сценарій має чітко заданий початковий стан, очікуваний результат та критерії успішного виконання. Ідеться про такі типи операцій: занесення в систему нового об’єкта з повним заповненням атрибутів, реєстрація нового клієнта з контактними даними, відкриття угоди і доведення її до етапу «переговори», планування показу об’єкта на конкретну дату й час, формування місячного звіту за активністю агента.

Стисло ці сценарії узагальнені в таблиці 3.2.

Таблиця 3.2 – Сценарії експериментального дослідження

№ сценарію	Назва сценарію	Короткий зміст	Тип користувача
1	Реєстрація нового об’єкта	Внесення в систему квартири або будинку	Ріелтор
2	Створення картки клієнта	Запис контактних даних та параметрів запиту	Ріелтор
3	Відкриття та супровід угоди	Зв’язування клієнта й об’єкта, зміна етапів	Ріелтор, менеджер
4	Планування показу	Фіксація показу в календарі, прив’язка до угоди	Ріелтор
5	Формування місячного звіту агента	Отримання статистики угод і показів	Менеджер, керівництво

Для кожного сценарію визначається набір вимірюваних параметрів. Час виконання відлічується від моменту, коли користувач отримує завдання, і до моменту, коли очікуваний результат чітко досягнутий (об’єкт збережено, угоду створено, звіт сформовано тощо). Кількість дій рахується як сумарна кількість кліків мишкою, переходів між формами та заповнених полів. Помилки фіксуються як випадки невдалого завершення сценарію: неправильне заповнення обов’язкових полів, повторна спроба введення тієї самої інформації через її втрату, плутанина між варіантами об’єктів, необхідність повністю перезапустити операцію.

Окремим етапом є суб’єктивне оцінювання зручності. Після завершення обох серій (традиційної і з використанням застосунку) кожен учасник заповнює коротку анкету, де оцінює зрозумілість структури інтерфейсу, логічність навігації, швидкість доступу до частовживаних функцій та загальне враження від роботи з системою. Оцінювання проводиться за шкалою від одного до п’яти балів.

У підсумку формується масив числових даних, який потім агрегується, усереднюється і візуалізується у вигляді таблиць та графіків. Це дозволяє перейти до етапу обробки та аналізу результатів.

3.2 Обробка та аналіз результатів експерименту

Після завершення експерименту всі дані було зведено до узагальнених таблиць, які відображають середній час виконання сценаріїв, середню кількість дій і кількість помилок при традиційному підході та при використанні розробленого застосунку. Крім того, окремо узагальнено суб’єктивні оцінки зручності системи.

Середні значення часу виконання сценаріїв і кількості дій наведено у таблиці 3.3. Для кожного сценарію представлено два порівнювані набори: традиційна організація роботи та робота із застосунком.

Таблиця 3.3 – Порівняння часу і кількості дій для типових сценаріїв

№ сценарію	Операція	Час, хв (традиційний підхід)	Час, хв (застосунок)	Кількість дій (традиційний підхід)	Кількість дій (застосунок)
1	Реєстрація нового об’єкта	10	5	35	18
2	Створення картки клієнта	7	3	22	12
3	Відкриття та супровід угоди	20	12	60	30
4	Планування показу	8	4	25	14
5	Формування місячного звіту	30	10	80	20

З аналізу таблиці видно, що час виконання кожної операції при використанні розробленого застосунку скорочується приблизно удвічі, а в окремих сценаріях (формування місячного звіту) скорочення є ще більш суттєвим. Зменшується й кількість елементарних дій, що необхідні для завершення завдання. Наприклад, у сценарії формування звіту ріелтор у традиційному режимі вимушений працювати з кількома таблицями та фільтрами, механічно переносити дані, що породжує до вісімдесяти дій. У застосунку звіт формують за допомогою налаштованої форми вибору періоду, що зменшує кількість кроків до двадцяти.

Динаміку зміни часу виконання сценаріїв зручно візуалізувати на стовпчиковій діаграмі, де для кожного сценарію у вигляді пари стовпчиків показано час без системи та час із системою (рис. 3.2).

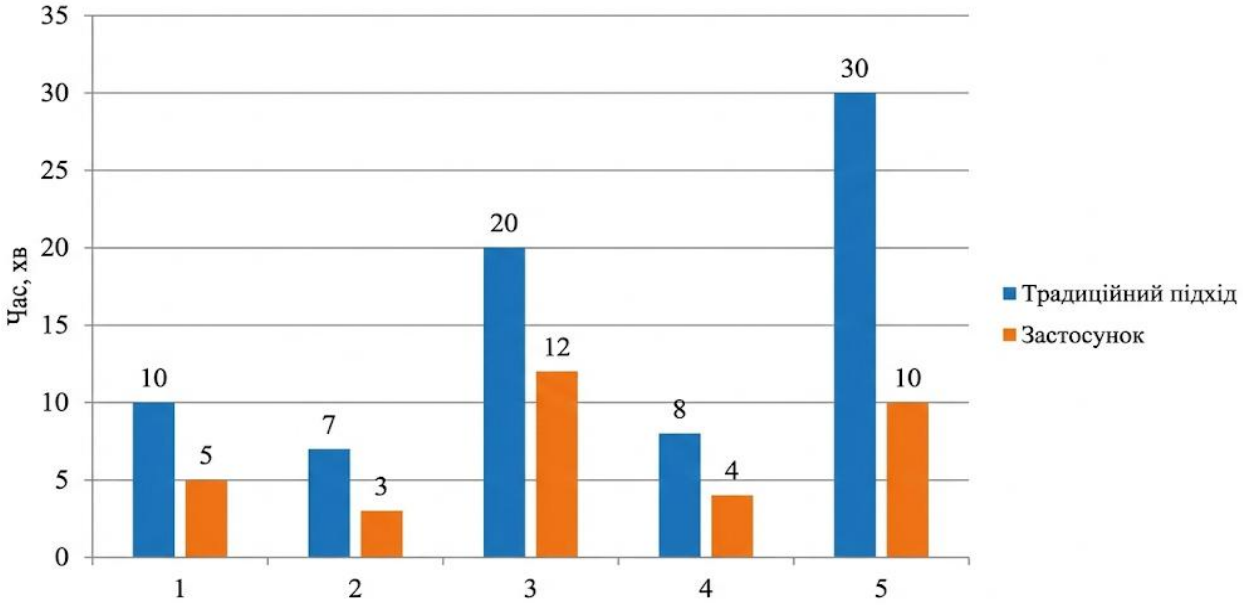

Рисунок 3.2 – Порівняння часу виконання типових операцій

Умовне зображення демонструє, що для кожного сценарію пара стовпчиків, яка відповідає роботі з системою, має помітно меншу висоту, ніж аналогічна пара для традиційного підходу.

Кількість помилок, зафіксованих при виконанні сценаріїв, наведено у таблиці 3.4. У кожному випадку розглядається десять повторень сценарію різними користувачами.

Таблиця 3.4 – Кількість помилок при виконанні сценаріїв

№ сценарію	Операція	Помилки за 10 спроб (традиційний підхід)	Помилки за 10 спроб (застосунок)
1	Реєстрація нового об’єкта	3	1
2	Створення картки клієнта	4	1
3	Відкриття та супровід угоди	5	2
4	Планування показу	2	0
5	Формування місячного звіту	4	0

Для планування показів і формування звітів розроблений застосунок фактично усуває помилки, оскільки більша частина дій автоматизована, а

структура форм і фільтрів не дозволяє залишити критично важливі поля незаповненими. В операціях реєстрації об’єктів і створення карток клієнтів помилки зводяться до поодиноких випадків, які, як правило, пов’язані з неправильною інтерпретацією вихідних даних користувачем, а не з інтерфейсними обмеженнями системи.

Графічне представлення зміни кількості помилок наведено на рисунку 3.3.

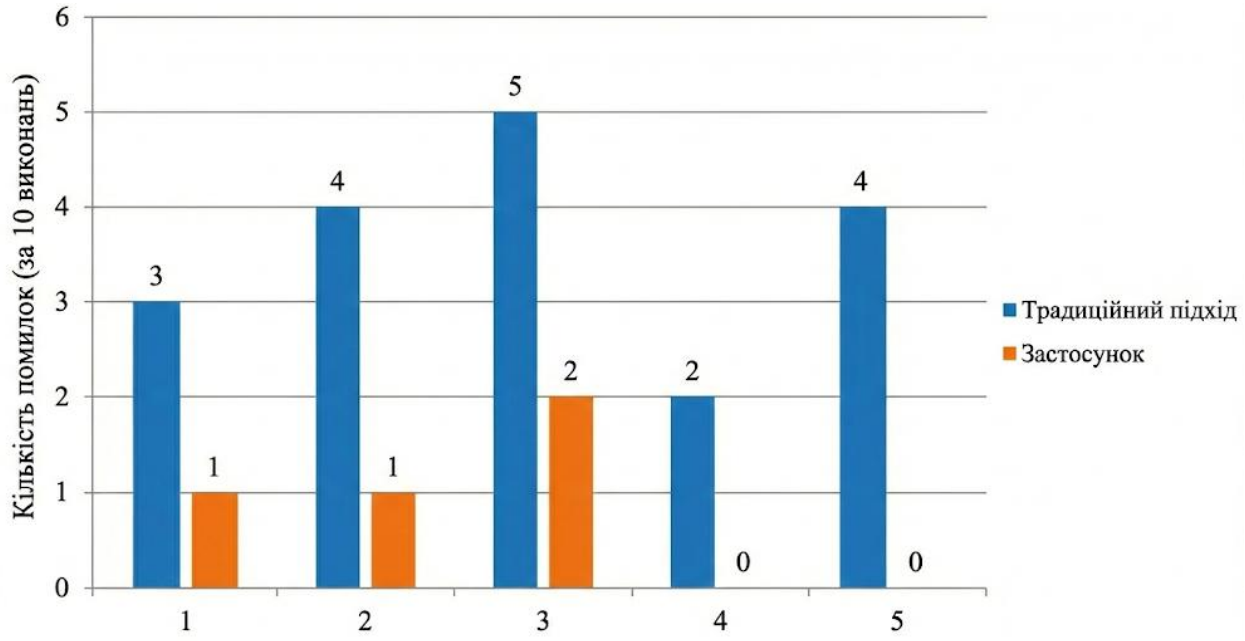

Рисунок 3.3 – Зміна кількості помилок при використанні системи

На діаграмі видно істотне зниження кількості помилок у всіх сценаріях, особливо у сценаріях, пов’язаних із повторюваними рутинними діями та підрахунками.

Для узагальнення результатів доцільно розрахувати інтегральні показники. Середній час виконання операцій у традиційному підході становить п’ятнадцять хвилин, тоді як при роботі з використанням системи цей показник зменшується до приблизно шести–семи хвилин. Середня кількість дій знижується з приблизно сорока чотирьох до дев’ятнадцяти, а середня кількість помилок із розрахунку на десять повторень сценарію зменшується із 3,6 до 0,8. На основі цих даних формується узагальнювальна таблиця 3.5.

Таблиця 3.5 – Узагальнені показники ефективності

Показник	Традиційний підхід	Застосунок	Відносна зміна, %
Середній час операції, хв	15,0	6,8	−54,7
Середня кількість дій на операцію	44,4	18,8	−57,7
Середня кількість помилок на 10 спроб	3,6	0,8	−77,8

Зменшення усіх трьох показників підтверджує гіпотезу про підвищення ефективності роботи агента при використанні розробленого застосунку. Особливо важливим є різке скорочення помилок, що безпосередньо впливає на якість обслуговування клієнтів і довіру до компанії.

Додатково проаналізовано суб’єктивні оцінки зручності інтерфейсу. Після завершення експерименту користувачі оцінювали зрозумілість інтерфейсу, логічність навігації, швидкість виконання типових дій, зручність роботи з воронкою угод і корисність звітів. Усі оцінки виставлялися за шкалою від одного до п’яти балів. Узагальнені результати наведено в таблиці 3.6.

Таблиця 3.6 – Оцінка зручності роботи із застосунком

Аспект оцінювання	Середній бал (з 5)
Зрозумілість структури інтерфейсу	4,4
Логічність навігації між екранами	4,2
Швидкість доступу до основних функцій	4,6
Зручність роботи з воронкою угод	4,3
Корисність звітів та аналітики	4,5
Загальне задоволення роботою системи	4,5

Отримані значення свідчать про загалом позитивне сприйняття застосунку кінцевими користувачами. Особливо високо оцінено швидкість доступу до частовживаних функцій і можливість оперативно формувати звіти, що раніше потребувало значних витрат часу на ручну обробку даних.

Для наочності інтегральний ефект застосування системи можна зобразити у вигляді графіка (рис. 3.4), де в одній площині відображено відносне зменшення часу, кількості дій і помилок.

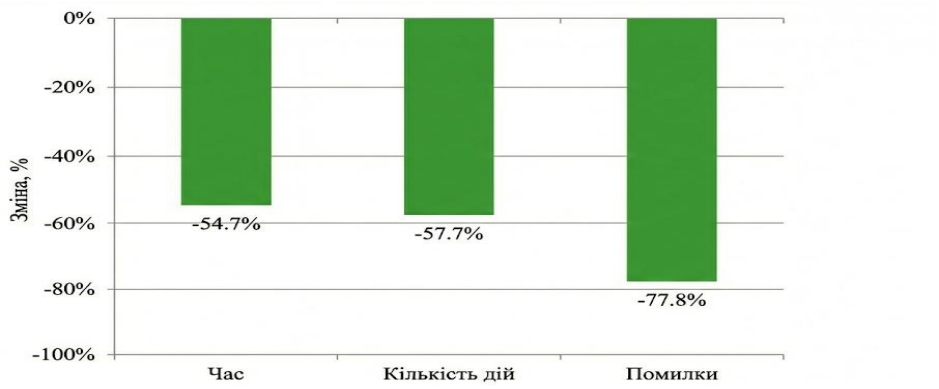

Рисунок 3.4 – Відносна зміна показників при переході до використання системи

Наведена умовна діаграма відображає, що всі обрані показники суттєво покращуються у бік зменшення ресурсоємності та помилковості процесів.

У процесі аналізу також виявлено низку обмежень і напрямів можливого удосконалення. По-перше, вибірка користувачів є порівняно невеликою, тому доцільно в подальшому розширити експеримент за рахунок залучення додаткових агентів з різним рівнем досвіду. По-друге, експеримент проводився в контрольованому середовищі, тоді як у реальних умовах на результати можуть впливати фактори зовнішньої інфраструктури, наприклад якість інтернет-з’єднання або паралельна робота з іншими системами. По-третє, у поточній версії не досліджувалася продуктивність системи при значних обсягах даних і при великій кількості одночасних користувачів, що відкриває поле для окремих навантажувальних тестів.

Попри ці обмеження, отримані результати демонструють, що розроблений застосунок ефективно підтримує ключові бізнес-процеси ріелторської компанії, зменшує витрати часу та людських ресурсів, підвищує точність даних і рівень задоволеності користувачів.

На основі отриманих результатів показано, що використання застосунку дозволяє більш ніж удвічі скоротити середній час виконання типових операцій, істотно зменшити кількість кроків, які виконує користувач, та майже в чотири рази знизити кількість помилок при роботі з даними.

ВИСНОВКИ

Кваліфікаційна робота присвячена розробці та дослідженню застосунку для управління в ріелторській сфері. Актуальність теми зумовлена цифровою трансформацією ринку нерухомості, зростанням обсягів даних про об’єкти та клієнтів, а також необхідністю підтримки повного циклу ріелторських операцій у єдиному інформаційному середовищі. У роботі розв’язано комплекс взаємопов’язаних теоретичних і практичних завдань, спрямованих на створення та оцінювання програмного продукту, орієнтованого на потреби невеликих і середніх ріелторських компаній.

У процесі виконання дослідження проведено всебічний аналіз предметної області ріелторської діяльності. Виокремлено основних учасників (клієнтів, ріелторів, керівників агенцій, партнерів) та описано ключові бізнес-процеси: ведення бази об’єктів, роботу з клієнтами, формування й супровід угод, організацію показів, підготовку документів і звітність. Показано, що при традиційному підході інформаційні потоки залишаються фрагментованими між різними інструментами (електронні таблиці, месенджери, паперові носії), що призводить до втрат часу, помилок, дублювання даних та відсутності цілісної картини стану угод.

Здійснено огляд сучасних програмних рішень для ринку нерухомості, включаючи горизонтальні CRM-системи загального призначення, спеціалізовані CRM для ріелторів, системи управління об’єктами та орендою (Property Management Systems), а також аналітичні платформи на основі методів аналізу даних і машинного навчання. Встановлено, що існуючі «коробкові» рішення не завжди враховують специфіку локальних ринків, можуть бути надмірно складними або закритими до розширення, що формує запит на гнучкі, спеціалізовані вебзастосунки, адаптовані під конкретні процеси агенції.

На основі аналізу предметної області та наявних технологічних підходів обґрунтовано вибір трирівневої клієнт–серверної архітектури з реалізацією модульного моноліту. Такий підхід дозволив поєднати простоту розгортання та

супроводу з чітким розподілом відповідальності між внутрішніми підсистемами (облікові записи, клієнти, об’єкти, угоди, розклад, звіти) і зберегти можливість еволюції системи до мікросервісної архітектури в майбутньому. Визначено доцільність використання стеку Python, Django та Django REST Framework для серверної частини, PostgreSQL як надійної реляційної СУБД з підтримкою ACID-транзакцій та можливими геопросторовими розширеннями, а також React і TypeScript для побудови односторінкового вебінтерфейсу.

Розроблено формальну інформаційну модель системи у вигляді ER-діаграми, яка включає базові сутності предметної області: користувачів, клієнтів, об’єкти нерухомості, угоди, покази, задачі та документи. Встановлено зв’язки між ними, які відображають реальні взаємодії у ріелторській практиці. На основі ER-моделі побудовано UML-діаграми класів, що лягли в основу реалізації доменної моделі в середовищі Django. Це забезпечило узгодженість між концептуальним описом предметної області й конкретною структурою бази даних та програмних класів.

Спроєктовано й реалізовано серверну частину застосунку у вигляді набору взаємопов’язаних модулів. Для кожної доменної сутності створено відповідні моделі, серіалізатори та REST-інтерфейси. Реалізовано бізнес-логіку підтримки життєвого циклу угоди, включаючи зміну етапів воронки продажів, планування показів, прив’язку задач і документів. Забезпечено контроль доступу до даних на основі ролей користувачів (адміністратор, менеджер, ріелтор), що дозволяє гнучко налаштовувати права на читання й модифікацію інформації залежно від посадових обов’язків.

Розроблено клієнтський інтерфейс у вигляді односторінкового вебзастосунку, який охоплює основні сценарії роботи ріелтора. Створено екран для роботи з каталогом об’єктів із фільтрами та таблицею результатів, інтерфейс керування клієнтами, канбан-дошку для візуалізації стану угод у воронці продажів, календар показів, а також модуль звітів. Компонентна структура інтерфейсу забезпечує повторне використання елементів, спрощує розширення функціоналу та підтримку коду.

Окрему увагу приділено модулю звітності та аналітики, який здійснює агрегацію даних щодо кількості угод на різних етапах воронки, активності агентів, тривалості угод і динаміки роботи за період. Реалізовано ендпоінти серверної частини для отримання агрегованих показників, а на клієнтській стороні – графічні представлення (стовпчикові діаграми, таблиці), які дозволяють керівництву агенції швидко оцінювати поточний стан справ та приймати обґрунтовані управлінські рішення.

На базі реалізованого застосунку сформовано та проведено експериментальне дослідження. Розроблено методику оцінювання ефективності, яка охоплює час виконання типових операцій, кількість дій, що виконує користувач, кількість помилок при роботі з даними, а також суб’єктивну оцінку зручності інтерфейсу. Проведено серію сценаріїв, що відтворюють ключові етапи роботи ріелтора: реєстрацію нового об’єкта, створення картки клієнта, відкриття й супровід угоди, планування показів та формування звітів.

За результатами експерименту встановлено, що впровадження застосунку дозволяє більш ніж удвічі скоротити середній час виконання типових операцій порівняно з традиційним підходом, що базується на розрізнених інструментах і ручній обробці даних. Середня кількість елементарних дій користувача зменшується майже на 60 %, а середня кількість помилок при виконанні сценаріїв – майже у чотири рази. Користувачі високо оцінили зручність інтерфейсу, логічність навігації, швидкість доступу до основних функцій та корисність візуалізації воронки угод і календаря показів, що підтверджується середніми оцінками в діапазоні від 4,2 до 4,6 бала за п’ятибальною шкалою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шипулін В. Д. Інтегрована інформаційна система нерухомості. Концепція для України: монографія. Харків: Харківський нац. ун-т міського господарства ім. О. М. Бекетова. 2021.

2. Мазаракі А. А., Волосович С. В. Фінансові послуги в екосистемі PropTech. Фінансово-кредитна діяльність: проблеми теорії та практики. 2023. Вип. 52.

3. Марченко О. В., Коляденко С. В. Цифрова трансформація будівельного бізнесу: тенденції та перспективи. Ефективна економіка. 2023. № 11.

4. Tan Z., Miller J. PropTech in the real estate operations and management: a systematic review. Journal of Property Investment & Finance. 2023.

5. Lozinska A. Development of the information system for solving the problem of renting and buying housing in the Netherlands with the introduction of a contextual search algorithm personalized and accelerated selection. 2024.

6. Sun J., Wang Z., Dang X., Zhang Y. Eye-Tracking Technology in Online Real Estate Rental. Scientific Programming. 2021. Vol. 2021.

7. CRM Platforms Use in Real Estate Agencies and Improved Customer Satisfaction: A Systematic Literature Review. 2025.

8. Ferreira M. S. та ін. Improving real estate CRM user experience and satisfaction. 2023.

9. Шипулін В. Д., Кривошеєва О. С. Інформаційні технології в управлінні об’єктами нерухомості: навч. посіб. Харків: ХНУМГ ім. О. М. Бекетова. 2020.

10. Bass L., Clements P., Kazman R. Software Architecture in Practice. 4th ed. Boston: Addison-Wesley. 2021.

11. Newman S. Building Microservices: Designing Fine-Grained Systems. 2nd ed. Sebastopol: O’Reilly Media. 2021.

12. Blinowski G., Ojdowska A., Przybylek A. Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation. IEEE. 2022.

13. Django Software Foundation. Django documentation. Version 5.x. 2023-2025. Офіційна документація фреймворку Django.

14. Django REST framework. Documentation. 2023-2025. Офіційна документація бібліотеки Django REST framework для побудови Web API.

15. Django REST framework для начинающих: создаём API блог-платформы. Хабр. 2021. Стаття.

16. React Team. React documentation. Version 18. 2022-2025. Офіційна документація бібліотеки React для побудови користувацьких інтерфейсів.

17. PostgreSQL Global Development Group. PostgreSQL 16 Documentation. 2023-2025. Офіційна документація СКБД PostgreSQL.

18. PostgreSQL Tutorial. Practical examples to understand PostgreSQL fast. 2024. Практичний посібник з розробки застосунків з PostgreSQL.

ДОДАТКИ

ДОДАТОК А

Модель клієнта (clients/models.py)

```
from django.db import models


class Client(models.Model):
    """
    Модель клієнта ріелторської агенції.
    Описує контактні дані та тип клієнта (покупець, продавець тощо).
    """

    class ClientType(models.TextChoices):
        BUYER = "buyer", "Покупець"
        SELLER = "seller", "Продавець"
        TENANT = "tenant", "Орендар"
        LANDLORD = "landlord", "Орендодавець"

    full_name = models.CharField(
        max_length=255,
        verbose_name="ПІБ клієнта",
    )
    phone = models.CharField(
        max_length=32,
        verbose_name="Телефон",
    )
    email = models.EmailField(
        blank=True,
        verbose_name="Електронна пошта",
    )
    client_type = models.CharField(
        max_length=20,
        choices=ClientType.choices,
        verbose_name="Тип клієнта",
    )
    notes = models.TextField(
        blank=True,
        verbose_name="Примітки",
    )
    created_at = models.DateTimeField(
        auto_now_add=True,
        verbose_name="Дата створення запису",
    )
    updated_at = models.DateTimeField(
        auto_now=True,
        verbose_name="Дата оновлення запису",
    )

    class Meta:
        verbose_name = "Клієнт"
        verbose_name_plural = "Клієнти"
        ordering = ["full_name"]

    def __str__(self) -> str:
        return f"{self.full_name} ({self.get_client_type_display()})"
```

ДОДАТОК Б

Модель об’єкта нерухомості (properties/models.py)

```
from django.db import models


class Property(models.Model):
    """
    Модель об'єкта нерухомості.
    Використовується для опису квартир, будинків, комерційних приміщень
тощо.
    """

    class PropertyType(models.TextChoices):
        APARTMENT = "apartment", "Квартира"
        HOUSE = "house", "Будинок"
        COMMERCIAL = "commercial", "Комерційна нерухомість"

    class PropertyStatus(models.TextChoices):
        ACTIVE = "active", "Активний"
        RESERVED = "reserved", "Зарезервований"
        SOLD = "sold", "Проданий"
        ARCHIVED = "archived", "В архіві"

    title = models.CharField(
        max_length=255,
        verbose_name="Назва об'єкта",
    )
    city = models.CharField(
        max_length=100,
        verbose_name="Місто",
    )
    address = models.CharField(
        max_length=255,
        verbose_name="Адреса",
    )
    property_type = models.CharField(
        max_length=20,
        choices=PropertyType.choices,
        verbose_name="Тип об'єкта",
    )
    rooms = models.PositiveIntegerField(
        verbose_name="Кількість кімнат",
    )
    area = models.DecimalField(
        max_digits=8,
        decimal_places=2,
        verbose_name="Площа, м²",
    )
    price = models.DecimalField(
        max_digits=12,
        decimal_places=2,
        verbose_name="Ціна",
    )
    description = models.TextField(
        blank=True,
```

```
    verbose_name="Опис об'єкта",
)
status = models.CharField(
    max_length=20,
    choices=PropertyStatus.choices,
    default=PropertyStatus.ACTIVE,
    verbose_name="Статус",
)
created_at = models.DateTimeField(
    auto_now_add=True,
    verbose_name="Дата створення запису",
)
updated_at = models.DateTimeField(
    auto_now=True,
    verbose_name="Дата оновлення запису",
)

class Meta:
    verbose_name = "Об'єкт нерухомості"
    verbose_name_plural = "Об'єкти нерухомості"
    ordering = ["-created_at"]

def __str__(self) -> str:
    return f"{self.title}, {self.city}"
```

ДОДАТОК В

Модель угоди та допоміжні моделі показу й задачі (deals/models.py, schedule/models.py)

```
from django.conf import settings
from django.db import models

from clients.models import Client
from properties.models import Property


class Deal(models.Model):
    """
    Модель угоди між клієнтом і власником об'єкта.
    Визначає етап воронки, відповідального агента та ключові дати.
    """

    class Stage(models.TextChoices):
        LEAD = "lead", "Лід"
        VIEWING = "viewing", "Показ"
        NEGOTIATION = "negotiation", "Переговори"
        RESERVATION = "reservation", "Резерв"
        CLOSED = "closed", "Завершено"

    client = models.ForeignKey(
        Client,
        on_delete=models.PROTECT,
        related_name="deals",
        verbose_name="Клієнт",
    )
    property = models.ForeignKey(
        Property,
        on_delete=models.PROTECT,
        related_name="deals",
        verbose_name="Об'єкт",
    )
    agent = models.ForeignKey(
        settings.AUTH_USER_MODEL,
        on_delete=models.PROTECT,
        related_name="deals",
        verbose_name="Відповідальний агент",
    )
    stage = models.CharField(
        max_length=20,
        choices=Stage.choices,
        default=Stage.LEAD,
        verbose_name="Етап угоди",
    )
    source_channel = models.CharField(
        max_length=50,
        blank=True,
        verbose_name="Канал надходження ліда",
    )
    created_at = models.DateTimeField(
        auto_now_add=True,
```

```
        verbose_name="Дата створення угоди",
    )
    closed_at = models.DateTimeField(
        null=True,
        blank=True,
        verbose_name="Дата завершення угоди",
    )
    notes = models.TextField(
        blank=True,
        verbose_name="Примітки по угоді",
    )

    class Meta:
        verbose_name = "Угода"
        verbose_name_plural = "Угоди"
        ordering = ["-created_at"]

    def __str__(self) -> str:
        return f"Угода #{self.pk}: {self.client} – {self.property}"

    def get_duration_days(self) -> int:
        """
        Обчислює тривалість угоди у днях.
        Якщо угода ще не завершена, використовується поточна дата.
        """
        from django.utils import timezone

        end_date = self.closed_at or timezone.now()
        return (end_date.date() - self.created_at.date()).days

    def can_transition(self, new_stage: str) -> bool:
        """
        Перевіряє допустимість переходу між етапами воронки.
        Наприклад, не дозволяє перескочити з 'lead' одразу в 'closed'.
        """
        allowed_order = [
            self.Stage.LEAD,
            self.Stage.VIEWING,
            self.Stage.NEGOTIATION,
            self.Stage.RESERVATION,
            self.Stage.CLOSED,
        ]
        try:
            current_index = allowed_order.index(self.stage)
            target_index = allowed_order.index(new_stage)
        except ValueError:
            return False
        return target_index >= current_index


class Viewing(models.Model):
    """
    Модель показу об'єкта в рамках конкретної угоди.
    """

    deal = models.ForeignKey(
        Deal,
        on_delete=models.CASCADE,
```

```
            related_name="viewings",
            verbose_name="Угода",
        )
        scheduled_at = models.DateTimeField(
            verbose_name="Запланована дата й час показу",
        )
        result = models.CharField(
            max_length=100,
            blank=True,
            verbose_name="Результат показу",
        )
        notes = models.TextField(
            blank=True,
            verbose_name="Коментар агента",
        )

        class Meta:
            verbose_name = "Показ об'єкта"
            verbose_name_plural = "Покази об'єктів"
            ordering = ["scheduled_at"]

        def __str__(self) -> str:
            return f"Показ угоди #{self.deal_id} на {self.scheduled_at}"


    class Task(models.Model):
        """
        Завдання для ріелтора/менеджера, пов'язане з певною угодою.
        """

        class Status(models.TextChoices):
            TODO = "todo", "Заплановано"
            IN_PROGRESS = "in_progress", "Виконується"
            DONE = "done", "Виконано"

        user = models.ForeignKey(
            settings.AUTH_USER_MODEL,
            on_delete=models.CASCADE,
            related_name="tasks",
            verbose_name="Виконавець",
        )
        deal = models.ForeignKey(
            Deal,
            on_delete=models.CASCADE,
            null=True,
            blank=True,
            related_name="tasks",
            verbose_name="Пов'язана угода",
        )
        title = models.CharField(
            max_length=255,
            verbose_name="Назва завдання",
        )
        due_date = models.DateTimeField(
            null=True,
            blank=True,
            verbose_name="Термін виконання",
        )
```

```
status = models.CharField(
    max_length=20,
    choices=Status.choices,
    default=Status.TODO,
    verbose_name="Статус",
)
priority = models.PositiveSmallIntegerField(
    default=1,
    verbose_name="Пріоритет",
)

class Meta:
    verbose_name = "Завдання"
    verbose_name_plural = "Завдання"
    ordering = ["status", "due_date"]

def __str__(self) -> str:
    return self.title
```