

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Вебсистема обліку та аналізу нерухомості»

Виконав студент групи ІПЗ-22-3
ШИРЯЄВ Максим Єгорович

Керівник роботи:
ЛОБАНЧИКОВА Надія Миколаївна

Рецензент:
БАКАЛЮК Тетяна Анатоліївна

Житомир – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії програмного
забезпечення

Тетяна ВАКАЛЮК

«28» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ШИРЯЄВ Максим Єгорович**

Керівник роботи: **ЛОБАНЧИКОВА Надія Миколаївна**

Тема роботи: **«Вебсистема обліку та аналізу нерухомості»**,

затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., №92/с**

Вихідні дані для роботи: об'єктом дослідження є процес збору, обробки, збереження та візуалізації даних у сфері нерухомості; предмет дослідження – алгоритми, архітектурні рішення та технологічний стек розробки платформи для обліку та аналізу нерухомості.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Лобанчикова Н.М.	01.03.2026р.	01.03.2026р.
2	Лобанчикова Н.М.	01.04.2026р.	01.04.2026р.
3	Лобанчикова Н.М.	01.05.2026р.	01.05.2026р.

РЕФЕРАТ

Пояснювальна записка до випускної кваліфікаційної роботи бакалавра на тему «Вебсистема обліку та аналізу нерухомості» займає 62 сторінок основного тексту. Матеріал доповнено 27 рисунками та 18 таблицями. Список використаних джерел містить 32 найменування і викладений на 3 сторінках, також наявні 11 сторінок додатків. Загальний обсяг кваліфікаційної роботи становить 72 сторінки.

Головне завдання цієї роботи полягає у проєктуванні та розробці програмного рішення для автоматизованого збору, обробки, збереження та візуалізації даних ринку нерухомості. Предметом дослідження виступають методи, архітектурні підходи та інструменти розробки таких інформаційних систем. Під час виконання роботи застосовувалися системний та порівняльний аналізи для вивчення існуючих рішень і формування вимог, а також підходи об'єктно-орієнтованого проєктування та моделювання баз даних.

Практична цінність результатів полягає в реалізації повноцінного програмного продукту. Розроблена вебсистема спирається на клієнт-серверну архітектуру за принципом модульного моноліта з використанням актуального стека технологій для забезпечення швидкої обробки запитів. Створена платформа пропонує користувачам інтерактивні дашборди та широкий набір інструментів для перегляду, зручної фільтрації і детального аналізу інформації у сфері нерухомості.

КЛЮЧОВІ СЛОВА: ВЕБСИСТЕМА, ОБЛІК НЕРУХОМОСТІ, АНАЛІТИКА ДАНИХ, CRM, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, ДАШБОРД.

					ІПЗ.КР.Б-121-26-ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Вебсистема обліку та аналізу нерухомості	Лім.	Арк.	Аркушів
Розроб.		Ширяєв М.Є.						
Керівник		Лобанчикова Н.М.					3	75
Н. контр.		Вакалюк Т.А.				Житомирська політехніка, група ІПЗ-22-3		
Затвердив		Вакалюк Т.А.						

ABSTRACT

The explanatory note for the bachelor's thesis titled "Real Estate Accounting and Analysis Web System" consists of 62 pages of main text. The material is illustrated with 27 figures and contains 18 tables. The bibliography includes 32 references spanning 3 pages, and there are also 11 pages of appendices. The total volume of the thesis is 72 pages.

The primary objective of this research is to design and develop a software system for the automated collection, processing, storage, and visualization of real estate market data. The subject of the study involves the methods, architectural approaches, and software tools required to build such an information system. The research employs comparative and systems analysis to review existing solutions and define requirements, alongside object-oriented design and database modeling techniques.

The practical significance of the results lies in the creation of a fully functional software product. The developed web system utilizes a client-server architecture based on a modular monolith approach, incorporating a modern web technology stack to ensure rapid request processing. This platform provides users with interactive dashboards and a comprehensive set of tools for viewing, filtering, and conducting in-depth analysis of real estate market information.

KEYWORDS: WEB SYSTEM, REAL ESTATE ACCOUNTING, DATA ANALYTICS, CRM, CLIENT-SERVER ARCHITECTURE, DASHBOARD.

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ВЕБСИСТЕМИ ОБЛІКУ ТА АНАЛІЗУ НЕРУХОМОСТІ.....	10
1.1 Постановка задачі.....	10
1.2 Аналіз аналогів програмного продукту.....	12
1.3 Вибір архітектури.....	17
1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення.....	20
Висновки до першого розділу	23
РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБСИСТЕМИ ДЛЯ ОБЛІКУ НЕРУХОМОСТІ	25
2.1 Визначення варіантів використання	25
2.2 Проектування логічної структури бази даних	28
2.3 Проектування та реалізація алгоритмів роботи системи.....	36
Висновки до другого розділу	39
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБСИСТЕМОЮ ОБЛІКУ ТА АНАЛІЗУ НЕРУХОМОСТІ.....	41
3.1 Порядок встановлення та налаштування параметрів системи.....	41
3.2 Структура інтерфейсу вебсистеми для обліку та аналізу нерухомості..	44
3.3 Тестування роботи програмного продукту	54
Висновок до третього розділу	58
ВИСНОВКИ.....	60
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	61
ДОДАТКИ	64
Додаток А.....	65

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API - Application Programming Interface

CI/CD - Continuous Integration / Continuous Deployment

JSON - JavaScript Object Notation

JWT - JSON Web Token

БД – база даних.

ПЗ – програмне забезпечення.

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. Сучасний етап розвитку інформаційних технологій характеризується цифровізацією всіх сфер суспільного життя, і ринок нерухомості не є винятком. Щодня генеруються величезні масиви даних - нові оголошення про продаж та оренду житлових і комерційних об'єктів, зміни цінових трендів, характеристики будівель та статуси угод. Традиційні підходи до обліку та аналізу такої інформації, які часто базуються на використанні розрізнених джерел, дошок оголошень або універсальних електронних таблиць, є неефективними. Цей процес вимагає значних витрат часу на ручну обробку, підвищує ризик помилок і не дозволяє оперативно отримувати цілісну картину для прийняття стратегічних рішень клієнтами та рієлторами.

В умовах високої конкуренції та цифровізації як покупцям, так і професійним учасникам ринку потрібні швидкі, гнучкі та точні інструменти для роботи з даними. Саме тому в останні роки спостерігається стійкий тренд на впровадження інтегрованих платформ у сфері PropTech (Property Technology). Розробка спеціалізованих вебсистем дозволяє автоматизувати процеси обліку, агрегації та візуалізації даних. Надання рієлторам вбудованих CRM-рішень, а користувачам – зручних інтерактивних мап та аналітичних дашбордів значно зменшує час на пошук та аналіз об'єктів нерухомості.

Але розробка сучасних систем аналізу нерухомості є складною архітектурною задачею. Вона вимагає проєктування надійних баз даних із підтримкою геопросторових індексів (для пошуку на мапі) та гнучких схем для збереження динамічних атрибутів (оскільки характеристики різних типів нерухомості суттєво відрізняються). Побудова системи вимагає впровадження ефективних підходів до структурування коду та побудови клієнт-серверної архітектури. Поєднання сучасного стеку технологій для фронтенд та бекенд розробки забезпечує необхідну продуктивність, інтерактивність та швидкість відгуку системи. З огляду на це, проєктування та розробка спеціалізованої вебсистеми обліку та аналізу нерухомості є актуальним науково-практичним завданням.

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

Об'єкт дослідження – процес проєктування та розробки інформаційних систем для обліку, обробки та аналізу даних у сфері нерухомості.

Предмет дослідження – методи, архітектурні підходи та програмні засоби створення вебсистеми обліку та аналізу нерухомості.

Метою роботи є проєктування та розробка вебсистеми обліку та аналізу нерухомості для збору, управління, агрегації та візуалізації даних про об'єкти нерухомості та забезпечує автоматизацію обліку для ріелторів, надає клієнтам аналітичний інструментарій.

Для досягнення поставленої мети було сформульовано та вирішено низку завдань дослідження.

1. Проаналізувати предметну область, специфіку ринку PropTech та провести огляд існуючих інформаційних систем-аналогів.

2. Обґрунтувати вибір сучасного стеку вебтехнологій та інструментів розробки, визначити функціональні й нефункціональні вимоги до програмного продукту та апаратного забезпечення.

3. Спроєктувати загальну архітектуру системи з використанням клієнт-серверного підходу (REST API), побудувати об'єктно-орієнтовану та компонентну структуру.

4. Спроєктувати базу даних для збереження інформації про нерухомість із підтримкою просторових геоданих та динамічних атрибутів, розробити алгоритми обробки інформації та взаємодії між логічними модулями системи.

5. Реалізувати вебсистему, включаючи розробку серверної частини (API), клієнтського інтерфейсу з інтерактивною мапою, модулем CRM для ріелторів, профілем користувача та панеллю адміністратора.

6. Провести тестування системи, налаштувати механізми безпеки та здійснити розгортання і демонстрацію роботи готового програмного продукту.

Методи дослідження. У роботі використано методи порівняльного та системного аналізу для огляду існуючих рішень і формування вимог до програмного забезпечення, методи об'єктно-орієнтованого проєктування та

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

архітектурного моделювання для розробки структури системи, а також методи проєктування реляційних та геопросторових баз даних.

Практичне значення одержаних результатів. Створена вебсистема обліку та аналізу нерухомості надає користувачам інтуїтивно зрозумілий інструментарій для пошуку, фільтрації та візуалізації пропозицій, що знижує складність аналізу ринку та оптимізує часові ресурсозатрати. Агенціям та ріелторам система пропонує інструменти CRM для управління портфелем об'єктів. Архітектурні рішення, закладені в систему (зокрема, використання геоіндексів та гнучких JSON-структур), дозволяють легко масштабувати платформу або інтегрувати її з іншими зовнішніми сервісами у сфері нерухомості.

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ВЕБСИСТЕМИ ОБЛІКУ ТА АНАЛІЗУ НЕРУХОМОСТІ

1.1 Постановка задачі

У сучасних умовах цифровізації процесів управління нерухомістю виникає об'єктивна потреба у відмові від застарілих та неструктурованих методів ведення обліку, зокрема використання розрізнених електронних таблиць або паперових документів. Впровадження комплексних вебсистем забезпечує якісно новий рівень систематизації, поєднуючи інструменти для детального обліку об'єктів, взаємодії з власниками та орендарями, укладання договорів і контролю фінансових потоків у єдиному централізованому цифровому середовищі. Це формує надійне підґрунтя для автоматизації рутинної операційної діяльності агентств нерухомості, приватних рієлторів та інвесторів.

Головною метою розробки є створення сучасної комплексної вебсистеми управління нерухомістю під назвою «Estateer», яка функціонуватиме як повноцінна спеціалізована система управління взаємовідносинами з клієнтами та гарантуватиме централізоване, безпечне й інтуїтивно зрозуміле оперування всіма супутніми даними.

Для досягнення визначеної мети необхідно виконати низку послідовних взаємопов'язаних завдань. Насамперед вимагається спроектувати загальну архітектуру системи та затвердити оптимальний технологічний стек, здатний забезпечити високу продуктивність, масштабованість і безпеку. За основу платформи необхідно взяти сучасні інструменти з підтримкою серверного рендерингу та суворої типізації коду, а також організувати повністю контейнеризовану інфраструктуру з налаштованими конвеєрами безперервної інтеграції та розгортання для автоматизації процесів тестування.

Наступним кроком є проєктування реляційної бази даних та інтеграція ефективного інструментарію для об'єктно-реляційного відображення. Спроектowana структура повинна містити деталізовану інформацію про об'єкти нерухомості різних категорій, вичерпні дані власників, контактну

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

інформацію та рейтинг орендарів, статуси активності договорів, а також фінансові транзакції та бюджети. При цьому ключовою вимогою є гарантування повної цілісності даних через чітко встановлені міжатабличні зв'язки.

Критично важливим етапом розробки є реалізація функціональних модулів та багаторівневої системи доступу з відповідними адаптивними інтерфейсами. Система повинна забезпечити адміністраторам безперешкодний контроль над глобальними налаштуваннями та обліковими записами. Основні користувачі, якими виступають агенти, повинні отримати розширений інструментарій для моніторингу лістингів, генерації унікальних електронних посилань для отримання заявок від потенційних орендарів, управління статусами договорів та ведення фінансової аналітики. Одночасно з цим власники майна повинні мати доступ до ізольованих особистих кабінетів виключно для моніторингу стану власних об'єктів та перегляду фінансової звітності. У якості додаткового каналу комунікації передбачається налаштування гнучкої системи автоматичних сповіщень через зовнішні месенджери для оперативного інформування про нові фінансові надходження чи заявки.

Завершальні етапи проєкту включають комплексне автоматизоване тестування розробленого коду з метою виявлення та усунення можливих дефектів, перевірки коректності роботи логіки платформи та оцінки її відмовостійкості. Після етапу тестування здійснюватиметься розгортання додатка на хмарних серверних потужностях. Експлуатаційний період передбачає постійну технічну підтримку, регулярне оновлення компонентів безпеки та оптимізацію роботи системи на основі аналізу взаємодії з нею кінцевих користувачів.

Результатом виконання поставлених завдань стане успішне впровадження в експлуатацію веб-платформи «Estateer», яка вирішить проблему хаотичного управління об'єктами нерухомості та надасть

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

спеціалістам галузі єдиний потужний інструмент для ефективної, прозорої та безпечної роботи з майном, клієнтами та фінансами.

1.2 Аналіз аналогів програмного продукту

У сучасному сегменті цифрових рішень для управління нерухомістю функціонує значна кількість спеціалізованих платформ та систем управління взаємовідносинами з клієнтами. Ці програмні продукти надають інструменти для автоматизації рутинних процесів агентств, ведення баз даних об'єктів, контролю фінансових потоків та взаємодії з власниками й орендарями. Для проведення ґрунтовного аналізу найефективніших рішень у цій галузі та виявлення їхніх сильних і слабких сторін було обрано такі сервіси як «LigaPro», «Buildium», «AppFolio» та «EstOffice».

Система «LigaPro» позиціонується як один із провідних українських програмних продуктів, створених для автоматизації щоденної роботи ріелторів та агентств нерухомості. Цей сервіс надає користувачам можливість формувати власну структуровану базу об'єктів та клієнтів, здійснювати автоматичне вивантаження оголошень на популярні агрегатори нерухомості, а також використовувати інструменти для моніторингу ринку через вбудовані алгоритми збору даних з відкритих джерел.

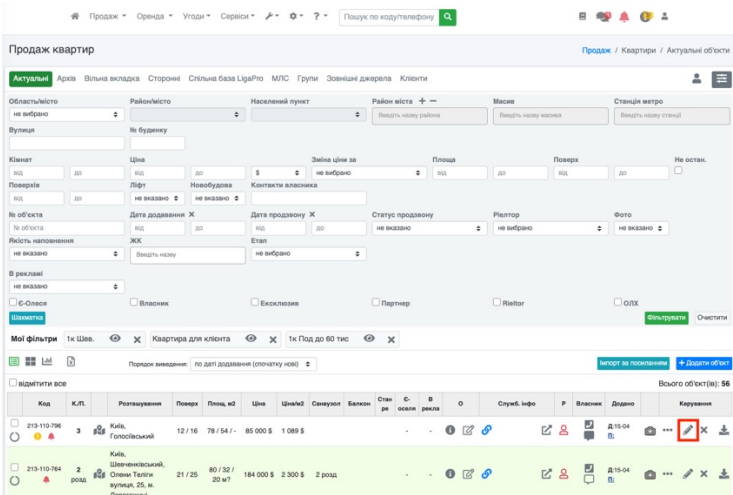


Рис.1.1 Інтерфейс головної панелі управління та бази об'єктів системи «LigaPro»

На представленому зображенні (рис.1.1) демонструється структура робочого простору агента з нерухомості, де зосереджено інструменти для управління лістингами та клієнтською базою. Візуалізація дозволяє об'єктивно оцінити щільність інформаційного наповнення інтерфейсу та логіку організації навігаційних елементів всередині вітчизняної платформи. Ознайомлення з цією схемою є критично важливим для розуміння рівня складності освоєння системи новими користувачами, що своєю чергою підкреслює необхідність розробки значно більш інтуїтивного та чистого дизайну у майбутньому проєкті.

До безперечних переваг «LigaPro» належить глибока адаптація до специфіки місцевого ринку, наявність функцій автоматизованого збору інформації та широкі можливості експорту лістингів на зовнішні ресурси. Основними недоліками цього рішення виступають перевантажений та морально застарілий користувацький інтерфейс, що вимагає тривалого початкового навчання персоналу. Також суттєвим мінусом є відсутність повноцінного спеціалізованого функціоналу для ведення глибокого фінансового обліку за договорами оренди та відсутність інтерактивних особистих кабінетів безпосередньо для власників майна.

Платформа «Buildium» є комплексним міжнародним програмним забезпеченням хмарного типу, яке орієнтоване на професійне управління житловим та комерційним майном. Цей сервіс пропонує потужні модулі для ведення бухгалтерського обліку, відстеження регулярних орендних платежів, організації технічного обслуговування об'єктів та формування деталізованої фінансової звітності. Крім того, система забезпечує функціонування окремих ізольованих порталів як для орендарів, так і для інвесторів.

Наведене зображення (рис. 1.2) ілюструє західний підхід до візуалізації фінансових потоків та структурування звітності для власників нерухомого майна. Схема чітко відображає інтеграцію графічних аналітичних елементів із сухими табличними даними щодо доходів та витрат у розрізі кожного окремого об'єкта.

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						13
Змн.	Арк.	№ докум.	Підпис	Дата		

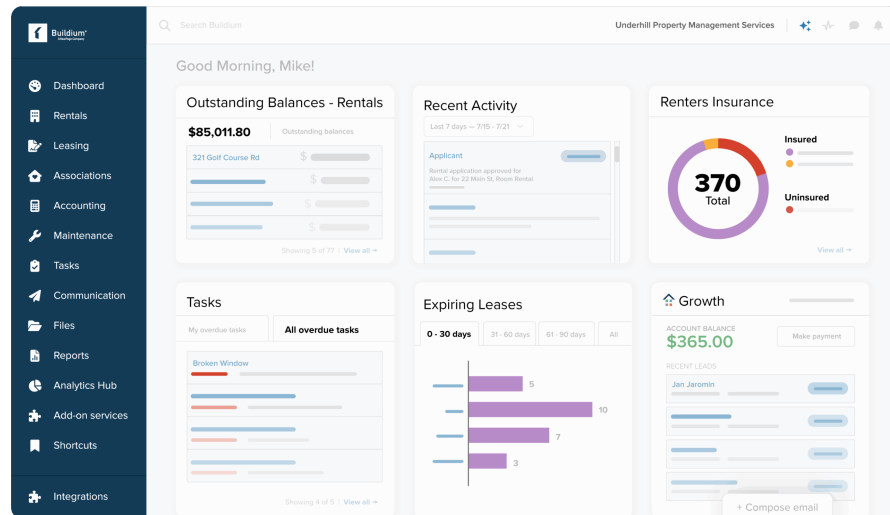


Рис.1.2 Панель фінансового аналізу та портал власника у системі «Buildium»

Детальний розгляд цього модуля є об'єктивно необхідним етапом для проєктування архітектури фінансового обліку в розроблюваній системі, оскільки він демонструє передові світові стандарти надання прозорої фінансової інформації користувачам. До ключових переваг «Buildium» можна віднести вичерпний набір інструментів для проведення фінансового аудиту, високий рівень автоматизації комунікації з резидентами об'єктів та надійну серверну архітектуру. До недоліків належить доволі висока вартість підписки, яка робить продукт економічно нерентабельним для приватних агентів, складна та тривала процедура налаштування системи, а також жорстка орієнтованість на західні стандарти оподаткування та юридичного документообігу.

Програмний комплекс «AppFolio» (рис.1.4) виступає аналогом корпоративного рівня, що забезпечує управління великими портфелями нерухомості із широким залученням технологій штучного інтелекту. Платформа здатна автоматизувати процеси публікації вакантних площ, обробки онлайн-заявок від потенційних орендарів, укладання смарт-договорів та інтелектуальної маршрутизації запитів на ремонтні роботи. Головними перевагами «AppFolio» є потужна цифрова екосистема, суттєва мінімізація ручної операційної праці за рахунок розумних алгоритмів та єдина централізована база даних підвищеної надійності.

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		14

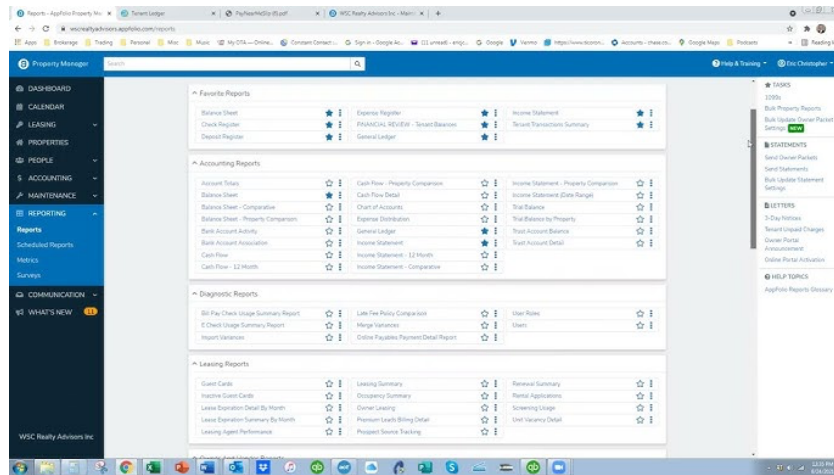


Рис.1.4 Інтерфейс головної панелі управління системи « AppFolio»

Основні недоліки полягають у наявності жорстких вимог до мінімальної кількості об'єктів для підключення, надзвичайно високій ціні впровадження системи та загальній перевантаженості функціями, які є надлишковими для локальних фахівців з нерухомості.



Рис.1.5 Інтерфейс головної панелі управління системи « AppFolio»

Українська платформа «EstOffice» (рис.1.5) розроблена переважно для оптимізації внутрішніх бізнес-процесів агентств нерухомості. Вона поєднує в собі інструменти класичної системи управління клієнтською базою, модулі постановки завдань для співробітників, а також розширену інтеграцію з сервісами цифрової телефонії і популярними месенджерами. Сильною стороною «EstOffice» є стовідсоткова локалізація, зручне управління щоденними завданнями всередині ріелторської команди та можливість швидкого розгортання без залучення технічних спеціалістів. Слабкими

сторонами продукту є його вузький фокус виключно на процесі продажу чи первинної здачі в оренду, ігнорування етапу подальшого довгострокового супроводу об'єкта, повна відсутність персональних кабінетів для власників майна та дуже обмежені можливості побудови прогнозних бюджетів.

Для об'єктивного зіставлення та систематизації проаналізованих даних було сформовано зведену таблицю.

Таблиця 1.1

Порівняння аналогів

Назва платформи	Наявність порталів власника	Модуль фінансового обліку	Адаптивність для агентів	Інтеграція з месенджерами
LigaPro	Відсутня	Базовий	Висока	Часткова
Buildium	Наявна	Розширений	Низька	Відсутня
AppFolio	Наявна	Розширений	Низька	Відсутня
EstOffice	Відсутня	Базовий	Висока	Наявна

Ця таблиця (таблиця 1.1) призначена для наочного порівняння чотирьох розглянутих платформ за тими ключовими критеріями, які мають найбільшу вагу для проєктування архітектури системи «Estate». Серед таких параметрів виділено наявність персональних порталів для клієнтів, потужність інструментів фінансового обліку, рівень адаптованості для невеликих учасників ринку та можливості інтеграції з комунікаційними сервісами. Головний висновок, який можна зробити на основі наведеної таблиці, полягає у тому, що на ринку програмного забезпечення існує помітний розрив між локальними системами, яким бракує управлінських та фінансових інструментів, і потужними західними платформами, які є занадто складними для типового вітчизняного користувача.

На основі проведеного аналізу функціональних можливостей та архітектурних особливостей існуючих програмних рішень сформовано остаточний перелік вимог до платформи «Estate». Розроблювана система повинна органічно поєднувати інтуїтивну простоту локальних сервісів із потужністю міжнародних аналогів. Основними функціональними вимогами

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

визначено реалізацію багаторівневого доступу з ізольованими робочими просторами для агентів, власників та адміністраторів; впровадження механізму генерації унікальних електронних посилань для отримання структурованих заявок на оренду, створення надійного модуля для ведення бюджетів та фіксації транзакцій у розрізі кожного об'єкта; розробку системи автоматичного відстеження статусів договорів оренди. Окремою вимогою є забезпечення гнучкої інтеграції з месенджером «Telegram»[22] для оперативного сповіщення учасників процесу про ключові події. Реалізація зазначеного комплексу вимог дозволить створити збалансований та високоефективний інструмент, що повністю відповідатиме потребам сучасного ринку управління нерухомістю.

1.3 Вибір архітектури

Вибір архітектурного шаблону веборієнтованої системи є фундаментальним етапом проєктування програмного забезпечення, оскільки саме архітектурні рішення визначають логічну організацію системи, способи внутрішньої взаємодії її компонентів, а також потенціал для подальшого масштабування та модернізації. Оптимально спроектована архітектура повинна враховувати специфіку предметної області, суворі вимоги до продуктивності, безпеки збереження даних, адаптивності користувацького інтерфейсу та можливостей надійної інтеграції із зовнішніми сервісами. Вона слугує базисом, який гарантує стабільну роботу платформи в умовах реального часу, мінімізує технічні ризики при внесенні змін та оптимізує процеси тестування і розгортання оновлень. Для обґрунтування кінцевого вибору доцільно проаналізувати найбільш поширені архітектурні моделі у сучасній сфері веброзробки.

Монолітна архітектура історично виступає найдавнішою та найвідомішою моделлю побудови програмного забезпечення. У межах цього підходу всі функціональні компоненти, включаючи користувацький інтерфейс, бізнес-логіку та модулі взаємодії з реляційними чи нереляційними

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

базами даних, об'єднуються в єдиний цілісний застосунок зі спільною кодовою базою. Головною перевагою такої структури є відносна простота початкової розробки, швидке локальне розгортання та легкість налагодження, що робить її привабливою для створення експериментальних прототипів або реалізації невеликих проєктів. Проте зі зростанням обсягу функціональності монолітна модель стрімко втрачає свою ефективність.

Будь-які модифікації окремих підсистем вимагають повної компіляції та перезапуску всієї системи, що підвищує ризик виникнення критичних збоїв. Крім того, виникають суттєві перешкоди при спробах точкового масштабування найбільш навантажених модулів та при організації паралельної роботи кількох незалежних команд програмістів.

Клієнт-серверна архітектура є сучасним галузевим стандартом та найбільш збалансованим підходом до розробки повноцінних вебдодатків. Ця модель передбачає чіткий логічний та фізичний розподіл програмного комплексу на дві взаємодіючі частини - клієнтську та серверну. Клієнтська частина виконується безпосередньо у середовищі користувача та повністю відповідає за візуалізацію динамічного інтерфейсу, первинну валідацію введених даних і навігацію. Серверна частина ізолювано обробляє складну бізнес-логіку, забезпечує криптографічний захист та автентифікацію, управляє збереженням інформації та формує стандартизовані відповіді на запити.

Ключовою перевагою цього підходу є суворе розмежування зон відповідальності, що значно спрощує підтримку чистоти коду та дозволяє незалежно нарощувати обчислювальні потужності бекенду для обслуговування великої кількості одночасних сесій. До певних недоліків можна віднести залежність клієнта від стабільності мережевого з'єднання та безумовну необхідність впровадження надійних протоколів шифрування для захисту конфіденційної інформації під час її передачі відкритими каналами зв'язку.

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

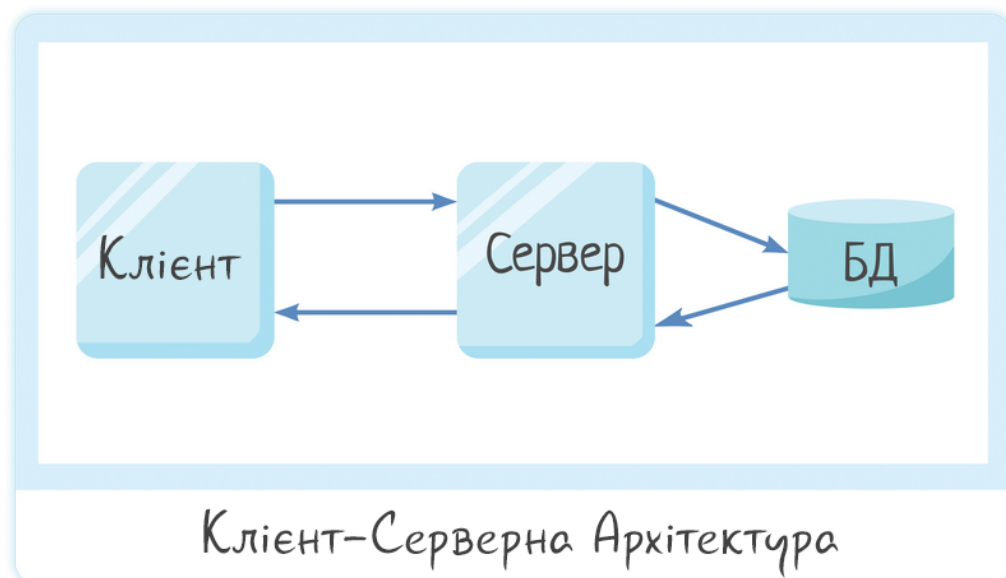


Рис.1.6 Схема взаємодії компонентів у клієнт-серверній архітектурі

На схемі (рис1.6) детально проілюстровано процес двосторонньої маршрутизації даних між фронтенд-додатком, який функціонує у браузері користувача, та бекенд-сервером, що безпосередньо обробляє бізнес-логіку системи управління нерухомістю. Візуалізація чітко демонструє повну ізоляцію бази даних, доступ до якої здійснюється виключно через захищені внутрішні серверні інтерфейси, що є критично важливим для розуміння принципів захисту фінансової інформації та персональних даних орендарів. Ця ілюстрація об'єктивно необхідна у тексті для наочного підтвердження того, як саме розподіляється обчислювальне навантаження між пристроєм клієнта та потужностями сервера під час виконання типових операцій на платформі.

Мікросервісна архітектура являє собою еволюційний крок у розвитку розподілених систем, що характеризується максимальним рівнем децентралізації. У цій парадигмі єдиний проєкт розділяється на десятки дрібних, повністю автономних програмних вузлів, кожен з яких відповідає за вузьку бізнес-функцію, працює з власною базою даних та спілкується з іншими через мережеві протоколи. Такий підхід забезпечує найвищу стійкість до локальних відмов та дозволяє масштабувати виключно ті сервіси, які зазнають пікового навантаження. Однак реалізація мікросервісів супроводжується екстремальним зростанням складності інфраструктури.

З'являється критична потреба у розгортанні систем автоматизованої оркестрації, безперервного розподіленого трасування запитів та складних механізмів синхронізації стану даних, що робить цю архітектуру надлишковою та економічно невиправданою для систем із середнім рівнем навантаження.

Остаточний вибір архітектурного патерну для розроблюваної платформи «Estate» базується на глибокому аналізі її цільового призначення, функціональних вимог та очікуваного вектора розвитку. З огляду на об'єктивну необхідність забезпечення високого рівня гнучкості, гарантування безпеки обробки фінансових транзакцій договорів оренди та створення високореактивного інтерфейсу, найбільш раціональним рішенням стало застосування класичної клієнт-серверної архітектури. Вона гарантує оптимальний баланс між затратами на розробку та широкими можливостями подальшого масштабування.

Такий підхід дозволить надійно сховати логіку обробки заявок і транзакцій у захищеному бекенд-середовищі, одночасно забезпечуючи плавний користувацький досвід в особистих кабінетах агентів та власників на стороні клієнта. Окрім цього, обрана клієнт-серверна модель ідеально інтегрується із затвердженим технологічним стеком, створюючи міцний фундамент для ефективної експлуатації програмного продукту.

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Враховуючи специфіку платформи «Estate», сформульовані технічні завдання та затверджену клієнт-серверну архітектуру, об'єктивно необхідно здійснити вибір комплексу інструментальних засобів, які забезпечать ефективну та безпечну реалізацію всіх запланованих модулів. Оскільки проєкт є комплексною веборієнтованою системою з високими вимогами до продуктивності та безпеки, для його розробки було обрано передовий технологічний стек, орієнтований на сучасні стандарти вебпрограмування.

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

Базовим середовищем виконання для серверної частини системи обрано платформу «Node.js». Це кросплатформене середовище дозволяє виконувати код мовою JavaScript поза межами веббраузера, використовуючи високопродуктивний рушій «V8». Головною перевагою платформи є її асинхронна, подієво-орієнтована архітектура з неблокуючою моделлю введення-виведення. Такий підхід ідеально підходить для створення масштабованих систем управління нерухомістю, оскільки дозволяє бекенду ефективно обробляти велику кількість одночасних з'єднань від агентів та власників без невиправданих витрат системних ресурсів.

Для безпосереднього написання вихідного коду як клієнтської, так і серверної частини системи обрано мову програмування «TypeScript». Це потужна надбудова над стандартом JavaScript, яка впроваджує строгу статичну типізацію на етапі розробки. Використання цієї мови дозволяє суттєво мінімізувати кількість помилок ще до етапу компіляції коду та значно спрощує процеси рефакторингу й підтримки складної архітектури. Впровадження типізації є критично важливим для гарантування безпеки фінансових розрахунків і цілісності даних при передачі інформації між модулями програми.

У якості надійної та продуктивної системи управління реляційними базами даних обрано об'єктно-реляційну систему «PostgreSQL» з розміщенням на спеціалізованому безсерверному хостингу «Neon» [5]. Ця система гарантує повну відповідність транзакцій вимогам ACID (атомарність, узгодженість, ізолюваність, довговічність), що є абсолютною необхідністю при проєктуванні модуля фінансового обліку та бюджетування. Серверна платформа підтримує виконання складних аналітичних запитів та має розвинені механізми забезпечення цілісності даних.

Для оптимізації взаємодії серверної логіки з базою даних замість традиційних SQL-запитів впроваджено сучасний інструмент об'єктно-реляційного відображення «Drizzle ORM». Цей інструментарій гарантує найвищий рівень типобезпеки, працюючи у тісній інтеграції з кодовою базою

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

проєкту, і дозволяє розробникам формувати складні запити до бази даних у зручному декларативному форматі з автоматичною перевіркою коректності типів. Для розробки динамічної та швидкої клієнтської частини системи застосовується потужний фреймворк «Next.js» [1] на базі маршрутизатора додатків. Цей фреймворк побудований поверх популярної бібліотеки для створення користувацьких інтерфейсів «React.js» [2]. Головною особливістю обраного інструменту є гібридний підхід до рендерингу вебсторінок, який включає як рендеринг на стороні сервера для прискорення завантаження критичного контенту, так і клієнтську генерацію для динамічної взаємодії. Вбудовані механізми маршрутизації та обробки серверних дій суттєво спрощують розробку складних багаторівневих інтерфейсів особистих кабінетів для різних ролей користувачів.

Наведена схема пояснює принцип роботи віртуальної об'єктної моделі документа у середовищі обраної бібліотеки. Цей механізм дозволяє системі оптимізувати оновлення візуальних елементів на екрані, перемальовуючи лише ті фрагменти інтерфейсу, які зазнали фактичних змін, наприклад, при оновленні статусу платежу або отриманні нової заявки на оренду. Розуміння цієї концепції є необхідним для усвідомлення того, як саме система забезпечує миттєву реакцію на дії користувача без повного перезавантаження сторінок платформи.

Забезпечення безпечної реєстрації, авторизації та багаторівневого контролю доступу у платформі «Estate» здійснюється за допомогою спеціалізованого сервісу «Clerk» [7]. Це рішення повністю бере на себе відповідальність за безпечне зберігання облікових даних користувачів, обробку входу через електронну пошту чи облікові записи зовнішніх провайдерів, а також управління життєвим циклом захищених сесій агентів, власників та адміністраторів.

Для забезпечення високої швидкості стилізації компонентів інтерфейсу застосовується утилітарний фреймворк «Tailwind CSS» [8], який дозволяє створювати складні адаптивні макети сторінок через застосування коротких

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

семантичних класів безпосередньо у коді розмітки. Додатково використовується набір нестилізованих доступних компонентів «Radix UI»[9] та бібліотека готових рішень «shadcn/ui [10]», які значно прискорюють процес побудови інтерактивних елементів, таких як модальні вікна, випадаючі списки та форми вводу даних.

Розгортання повнофункціональної системи та налаштування конвеєрів безперервної інтеграції здійснюється з використанням платформ контейнеризації та автоматизації процесів тестування й доставки коду на робочі сервери. Такий підхід забезпечує абсолютну ідентичність середовищ розробки та експлуатації, запобігаючи виникненню непередбачуваних помилок при випуску оновлень.

Апаратне та програмне забезпечення користувачів повинно відповідати мінімальним вимогам для забезпечення комфортної та стабільної роботи з вебінтерфейсом платформи. Процесор із тактовою частотою не менше 1,2 ГГц; мінімальний обсяг оперативної пам'яті - 4 ГБ; обсяг вільного простору на накопичувачі - 500 МБ для збереження тимчасових файлів браузера; операційна система рівня не нижче «Windows 10», «macOS 11», «Ubuntu 20.04», «Android 12» або «iOS 16»; актуальна версія сучасного браузера з повноцінною підтримкою стандарту «ECMAScript 6» та механізмів виконання асинхронного коду.

Висновки до першого розділу

Здійснено ґрунтовний аналіз функціональних можливостей, архітектурних особливостей та наявних обмежень провідних існуючих онлайн-платформ у сфері оренди та управління нерухомістю. На основі отриманих результатів дослідження та виявлених недоліків наявних аналогів було сформульовано вичерпний перелік ключових технічних і функціональних вимог до розроблюваної системи.

Для забезпечення надійності, масштабованості та ефективності майбутнього програмного продукту обґрунтовано вибір клієнт-серверної

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		23

архітектури з використанням архітектурного шаблону «MVC» та проєктуванням програмного інтерфейсу за принципами передачі стану представлення. Застосування такого підходу гарантує чіткий логічний розподіл зон відповідальності між різними шарами системи, відокремлюючи візуальний інтерфейс від серверної бізнес-логіки.

Інструментарій для практичної реалізації проєкту складається з комплексу сучасних технологічних рішень, підібраних відповідно до затверджених вимог. Серверна частина реалізовуватиметься на базі середовища «Node.js» із застосуванням мови програмування «TypeScript» для забезпечення суворої типізації, інструменту об'єктно-реляційного відображення «Prisma ORM» та реляційної системи управління базами даних «PostgreSQL» для надійного збереження інформації. Клієнтська частина будуватиметься з використанням фреймворку «Next.js» на основі бібліотеки «React» та інструменту «Tailwind CSS» для створення швидкого адаптивного інтерфейсу. З метою вирішення специфічних функціональних завдань передбачено інтеграцію бібліотеки «React Leaflet» для забезпечення картографічної візуалізації об'єктів нерухомості та сервісу «LiqPay» для безпечної обробки фінансових платежів. Автоматизоване розгортання розробленої системи та оптимізована мережева доставка контенту кінцевим користувачам забезпечуватиметься ресурсами хмарної платформи «Vercel».

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБСИСТЕМИ ДЛЯ ОБЛІКУ НЕРУХОМОСТІ

2.1 Визначення варіантів використання

Для забезпечення ефективної реалізації ключових функціональних можливостей системи управління нерухомістю «Estate» та гарантування її відповідності визначеним технічним умовам, необхідно сформулювати узагальнені сценарії використання. Ці сценарії визначають типову поведінку платформи при взаємодії з різними категоріями користувачів. Систематизація таких сценаріїв дозволяє чітко структурувати функціональність та встановити логічні межі відповідальності кожної підсистеми, що слугуватиме фундаментом для безпосередньої розробки програмних модулів.

Загальні функціональні вимоги до платформи охоплюють низку важливих процесів. Зокрема, система має забезпечувати створення особистих облікових записів та проходження процедури автентифікації для різних типів користувачів. Також передбачається управління інформацією про об'єкти нерухомості, що дозволяє її додавати, редагувати та видаляти, разом із веденням деталізованої бази даних власників нерухомості з чіткою прив'язкою до відповідних об'єктів. Платформа повинна вміти генерувати унікальні електронні посилання для збору заявок на оренду від потенційних клієнтів, а також повноцінно підтримувати формування, підписання та відстеження статусів договорів оренди. Крім того, функціонал має включати ведення фінансового обліку для фіксації транзакцій та планування бюджетів у розрізі кожного об'єкта. Наостанок, система повинна дозволяти формування рейтингів орендарів із можливістю залишення текстових відгуків та підтримувати інтеграцію із зовнішніми месенджерами для зручного отримання автоматичних системних сповіщень.

Для систематизації інформації основні функціональні та нефункціональні вимоги згруповано у відповідних таблицях (Таблиця 2.1 – Таблиця 2.2).

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 2.1

Основні функціональні вимоги платформи

Визначення	Опис
Автентифікація та авторизація	Система повинна забезпечувати безпечний процес реєстрації та входу користувачів із використанням електронної пошти або зовнішніх провайдерів автентифікації з подальшим розподілом прав доступу на основі ролей.
Управління даними	Платформа зобов'язана гарантувати надійне збереження введеної інформації, надаючи користувачам інструменти для її перегляду, редагування та видалення у межах встановлених повноважень.

Таблиця 2.2

Основні нефункціональні вимоги платформи

Визначення	Опис
Сприйняття та зручність	Час, необхідний для засвоєння базових принципів роботи з інтерфейсом для нового користувача, не повинен перевищувати 20 хвилин. Інтерфейс має бути адаптивним та містити зрозумілі навігаційні елементи.
Продуктивність	Час реакції системи на стандартні запити користувача не повинен перевищувати 2 секунд. Для операцій, що вимагають генерації складних фінансових звітів, допустимий час очікування становить до 10 секунд.

Надійність та доступність	Система повинна забезпечувати безперебійну роботу на рівні 99,5% часу. Заплановані технічні роботи мають проводитися у години мінімальної активності користувачів.
Масштабованість	Архітектура бекенду повинна бути готовою до горизонтального масштабування шляхом розгортання додаткових серверних контейнерів у разі стрімкого зростання кількості активних сесій.

Аналіз функціональних потреб став підґрунтям для визначення ролей у системі та варіантів використання платформи. У системі «Estate» передбачено три базові ролі із суворо розмежованими рівнями доступу - адміністратор, агент з нерухомості та власник об'єкта.

Адміністратор виступає як суперкористувач із необмеженим доступом до всіх модулів та даних платформи. Головною відповідальністю цієї ролі є глобальне управління системою, моніторинг її стану, управління обліковими записами агентів, включаючи можливість їх блокування або зміни рівнів доступу для забезпечення внутрішньої безпеки.

Агент з нерухомості (або ріелтор) є основним активним користувачем платформи, для якого призначена переважна більшість робочих інструментів. Агент несе повну відповідальність за ведення об'єктів, закріплених за його обліковим записом. Ця роль наділена повноваженнями створювати нові лістинги, керувати медіафайлами об'єктів, формувати унікальні посилання для заявок на оренду та обробляти отримані звернення від потенційних орендарів. Агент має виключне право укладати договори оренди у системі, змінювати їхні статуси, а також вносити дані щодо фінансових транзакцій (доходів та витрат) і формувати бюджети по кожному об'єкту. Крім того, саме агент здійснює оцінку орендарів після завершення дії договору.

Власник об'єкта (інвестор) є користувачем із обмеженим, переважно споглядальним доступом до системи. Власники отримують персональні

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

облікові записи, через які вони можуть відстежувати поточний стан виключно тих об'єктів нерухомості, які юридично їм належать. Інтерфейс власника дозволяє переглядати ключові фінансові показники, сформовані звіти про доходи та витрати, а також базову інформацію про поточних орендарів. При цьому власник позбавлений можливості редагувати дані об'єктів, змінювати умови договорів чи втручатися у процес прийняття заявок на оренду.

2.2 Проектування логічної структури бази даних

Для забезпечення надійного збереження, швидкої обробки та ефективного управління великими масивами інформації у системі «Estate» розроблено оптимізовану реляційну структуру бази даних. Архітектура сховища даних проектувалася з урахуванням необхідності підтримки складної бізнес-логіки управління нерухомістю, гарантування цілісності фінансових транзакцій та забезпечення високої швидкості виконання запитів від клієнтських додатків. Логічна структура бази даних складається з дев'яти основних таблиць, кожна з яких відповідає за ізольований домен предметної області.

Для систематизації інформації про власників нерухомості у системі передбачено окрему таблицю. Її призначення полягає в акумуляції ідентифікаційних та контактних даних інвесторів, чиї об'єкти обслуговуються на платформі.

Таблиця 2.3

Опис таблиці власників нерухомості

Поле	Тип даних	Опис
owner_id	uuid	Унікальний ідентифікатор власника
clerk_user_id	varchar	Ідентифікатор із системи автентифікації
full_name	varchar	Повне ім'я власника

phone_number	varchar	Контактний номер телефону
email	varchar	Електронна пошта
notes	text	Додаткові нотатки агента

Наведена таблиця створена для формування централізованого реєстру інвесторів та їх подальшої інтеграції у систему багаторівневого доступу. Вона відображає базові параметри власників, такі як імена, канали зв'язку та унікальні ідентифікаційні ключі зовнішньої системи авторизації. Головним висновком з аналізу цієї структури є те, що вона забезпечує надійне розмежування профілів власників, що є необхідною умовою для створення ізольованих особистих кабінетів та коректної прив'язки майна до конкретних осіб.

Фундаментальною сутністю всієї платформи є таблиця об'єктів нерухомості, яка зберігає вичерпну інформацію про кожен зареєстрований у системі актив.

Таблиця 2.4

Опис таблиці об'єктів нерухомості

Поле	Тип даних	Опис
property_id	uuid	Унікальний ідентифікатор об'єкта
owner_id	uuid	Посилання на власника нерухомості
agent_id	varchar	Посилання на відповідального агента
title	varchar	Комерційна назва об'єкта
address	varchar	Точна фізична адреса
property_type	varchar	Категорія приміщення
status	varchar	Поточний стан доступності
area	numeric	Загальна площа об'єкта

продовж. табл. 2.4

price	numeric	Встановлена вартість
description	text	Текстовий опис характеристик
media_urls	jsonb	Посилання на збережені медіафайли

Зазначена таблиця створена як головний інформаційний хаб системи, навколо якого будується вся подальша логіка оренди та фінансового обліку. Вона відображає та деталізує вичерпний спектр параметрів майна, починаючи від базових фізичних характеристик і закінчуючи статусом доступності, мультимедійними даними та ієрархічною прив'язкою до агента і власника. З даної таблиці можна зробити висновок, що платформа підтримує універсальний підхід до зберігання даних, дозволяючи однаково ефективно оперувати як житловими квартирами, так і комерційними або земельними активами.

Для управління життєвим циклом клієнтів агенції спроектовано таблицю обліку орендарів, що фіксує поточних та колишніх резидентів.

Таблиця 2.5

Опис таблиці орендарів

Поле	Тип даних	Опис
tenant_id	uuid	Унікальний ідентифікатор орендаря
property_id	uuid	Посилання на орендований об'єкт
full_name	varchar	Повне ім'я орендаря
contact_info	varchar	Контактні дані
start_date	date	Дата фактичного заселення
end_date	date	Дата очікуваного виселення
monthly_rent	numeric	Щомісячна орендна плата
is_active	boolean	Статус актуальності проживання

Таблиця орендарів створена з метою ведення структурованої бази даних клієнтів та відстеження їхньої активності. Вона порівнює та відображає часові проміжки проживання, контактні дані та закріплені за конкретними особами фінансові зобов'язання у вигляді суми щомісячного платежу. Головний висновок полягає у тому, що наявність булевого статусу активності дозволяє системі автоматично відокремлювати діючих клієнтів від архівних, зберігаючи при цьому повну історію взаємодії для формування аналітичних звітів. Юридична площина взаємовідносин між сторонами фіксується у таблиці договорів, яка є невід'ємною частиною модуля документального супроводу.

Таблиця 2.6

Опис таблиці договорів

Поле	Тип даних	Опис
contract_id	uuid	Унікальний номер документа
tenant_id	uuid	Посилання на сторону орендаря
property_id	uuid	Посилання на предмет оренди
valid_from	date	Дата набрання чинності
valid_until	date	Дата завершення дії
rent_amount	numeric	Зафіксована сума платежу
status	varchar	Юридичний стан документа
file_url	varchar	Посилання на відскановану копію
notes	text	Додаткові умови договору

Дана таблиця створена для суворого контролю за документообігом та автоматизації процесів пролонгації угод. Вона відображає ключові параметри будь-якого договору - терміни його дії, фінансові умови, поточний статус активності та містить пряме посилання на цифрову копію оригіналу. З цієї структури можна зробити висновок, що платформа здатна автоматично аналізувати дати завершення угод та своєчасно ініціювати зміну їхніх статусів, попереджаючи агентів про необхідність переукладання документів. Фінансова

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						31
Змн.	Арк.	№ докум.	Підпис	Дата		

стабільність та звітність забезпечуються через таблицю транзакцій, що діє як головна бухгалтерська книга платформи.

Таблиця 2.7

Опис таблиці транзакцій

Поле	Тип даних	Опис
transaction_id	uuid	Унікальний ідентифікатор операції
property_id	uuid	Посилання на об'єкт нерухомості
tenant_id	uuid	Опціональне посилання на орендаря
type	varchar	Категорія руху коштів
amount	numeric	Сума фінансової операції
transaction_date	date	Дата фактичного проведення
description	text	Призначення платежу

Ця таблиця створена для безперервної фіксації абсолютно всіх грошових потоків, що проходять через платформу. Вона відображає та систематизує параметри доходів і витрат, їхній обсяг, дати проведення та обов'язкову прив'язку до конкретних об'єктів нерухомості. Головним висновком є те, що саме ця сутність виступає джерелом даних для генерації розгорнутих фінансових звітів для власників, гарантуючи абсолютну прозорість економічної ефективності кожного окремого активу. Процес первинного відбору клієнтів оптимізовано за допомогою таблиці заявок на оренду, яка збирає анкети через унікальні токени.

Таблиця 2.8

Опис таблиці заявок на оренду

Поле	Тип даних	Опис
application_id	uuid	Унікальний ідентифікатор анкети
property_id	uuid	Посилання на бажаний об'єкт
token	varchar	Унікальне посилання для доступу
applicant_name	varchar	Ім'я потенційного клієнта
monthly_income	numeric	Задекларований рівень доходу

Поле	Тип даних	Опис
people_count	integer	Запланована кількість мешканців
has_pets	boolean	Наявність домашніх тварин
status	varchar	Поточний етап розгляду

Таблиця заявок створена для цифровізації та стандартизації процесу отримання звернень від зацікавлених осіб. Вона відображає критично важливі для прийняття рішення параметри потенційних орендарів - їхню платоспроможність, склад сім'ї, наявність тварин та стадії обробки анкети агентом. Висновок з аналізу таблиці підтверджує, що наявність системи токенів дозволяє безпечно збирати персональні дані із зовнішніх джерел без необхідності реєстрації заявників у системі. Для формування внутрішньої екосистеми довіри розроблено таблицю рейтингів, де зберігаються результати оцінювання клієнтів.

Таблиця 2.9

Опис таблиці рейтингів орендарів

Поле	Тип даних	Опис
rating_id	uuid	Унікальний ідентифікатор відгуку
tenant_id	uuid	Посилання на оцінювану особу
payment_score	integer	Оцінка своєчасності розрахунків
condition_score	integer	Оцінка дбайливості ставлення
overall_rating	numeric	Розрахований середній бал
comments	text	Розгорнутий текстовий коментар

Ця таблиця створена для накопичення репутаційної історії клієнтів після або під час їхнього проживання. Вона відображає кількісні оцінки дисципліни платежів, якості утримання майна, а також загальний рейтинг, доповнений суб'єктивними нотатками агентів. Головний висновок з її використання

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		33

полягає у можливості формування єдиної внутрішньої бази надійних та проблемних орендарів, що суттєво знижує фінансові ризики при укладанні майбутніх угод. Система автоматичного інформування базується на таблиці налаштувань сповіщень, яка керує інтеграцією з зовнішнім месенджером.

Таблиця 2.10

Опис таблиці налаштувань сповіщень

Поле	Тип даних	Опис
setting_id	uuid	Ідентифікатор конфігурації
user_id	varchar	Посилання на користувача системи
chat_id	varchar	Ідентифікатор чату месенджера
notify_payments	boolean	Дозвіл на сповіщення про оплату
notify_applications	boolean	Дозвіл на інформування про заявки
notify_reports	boolean	Дозвіл на надсилання звітів

Таблиця налаштувань створена для забезпечення гнучкої персоналізації каналів комунікації між платформою та користувачем. Вона відображає параметри інтеграції стороннього сервісу та масив логічних перемикачів, що регулюють типи повідомлень, які бажає отримувати клієнт. Висновок з аналізу цієї сутності свідчить про високий рівень орієнтованості системи на користувача, оскільки вона запобігає інформаційному перевантаженню, дозволяючи агентам та власникам самостійно визначати пріоритетність подій.

Модуль довгострокового економічного планування спирається на таблицю фінансових бюджетів, яка дозволяє прогнозувати доходи.

Таблиця 2.11

Опис таблиці фінансових бюджетів

Поле	Тип даних	Опис
budget_id	uuid	Унікальний ідентифікатор плану
user_id	varchar	Посилання на власника плану
year	integer	Звітний рік

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						34
Змн.	Арк.	№ докум.	Підпис	Дата		

Поле	Тип даних	Опис
month	integer	Звітний місяць
planned_revenue	numeric	Очікувана сума надходжень

Таблиця бюджетів створена виключно для забезпечення функціоналу фінансового планування та побудови прогнозної аналітики. Вона порівнює та відображає очікувані суми доходів у розрізі конкретних календарних місяців та років для кожного облікового запису. Головним висновком є те, що інтеграція цих даних із таблицею фактичних транзакцій дозволяє системі автоматично генерувати графіки виконання планів та виявляти касові розриви.

На основі детально визначених таблиць та їхніх атрибутів побудовано комплексну діаграму структури бази даних. Вона наочно відображає концептуальні взаємозв'язки між основними сутностями розробленої системи управління нерухомістю.

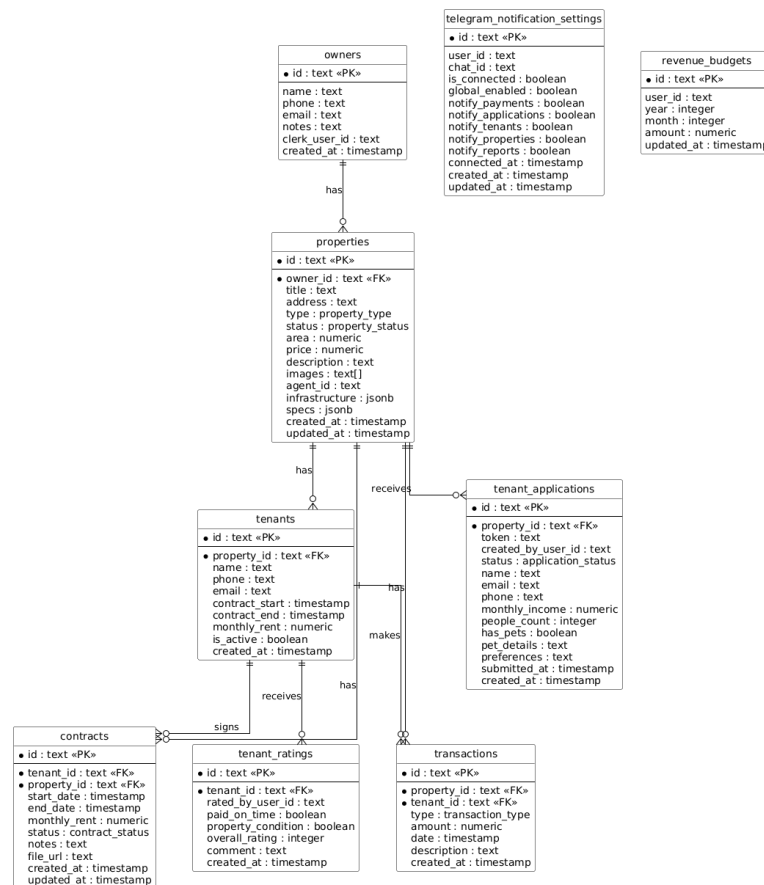


Рис.2.1 Діаграма структури бази даних

На представленій схемі (рис.2.1) детально візуалізовано архітектуру реляційної бази даних платформи «Estateer», включно з усіма дев'ятьма ключовими таблицями та логічними зв'язками між ними. Ілюстрація наочно демонструє механізми забезпечення цілісності інформації, зокрема ієрархічну прив'язку транзакцій, договорів, заявок та орендарів до конкретних об'єктів нерухомості за класичним принципом зв'язків «один до багатьох». Ця візуалізація є об'єктивно необхідною у тексті документа, оскільки вона дозволяє комплексно усвідомити логіку маршрутизації даних всередині бекенду та принципи функціонування інструменту об'єктно-реляційного відображення при виконанні запитів.

2.3 Проектування та реалізація алгоритмів роботи системи

З огляду на функціональне призначення та архітектурні особливості комплексної вебплатформи управління нерухомістю «Estateer», система характеризується інтуїтивно зрозумілим користувацьким інтерфейсом та логічним розподілом доступу до основних дій. Це дозволяє користувачам ефективно взаємодіяти з платформою без необхідності розуміння складних внутрішніх алгоритмів обробки даних. Основна алгоритмічна логіка системи побудована навколо життєвого циклу об'єкта нерухомості та базується на суворому рольовому розмежуванні прав доступу, що гарантує інформаційну безпеку та операційну ефективність процесів.

Для систематизації інформації щодо розподілу повноважень у системі було сформовано зведену таблицю. Вона створена для чіткого розмежування функціональних можливостей кожної з визначених ролей платформи. Таблиця відображає та порівнює набір дозволених дій для потенційних орендарів, власників майна, агентів з нерухомості та системних адміністраторів. Головним висновком, який можна зробити з цієї таблиці, є те, що платформа забезпечує повну ізоляцію процесів операційного управління, делегованих агентам, від функцій пасивного фінансового моніторингу, доступних власникам, при збереженні абсолютного контролю над системою з боку адміністраторів.

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						36
Змн.	Арк.	№ докум.	Підпис	Дата		

Основний функціонал користувачів залежно від визначеної ролі

Роль користувача	Повноваження та доступний функціонал
Власник об'єкта	Користувач проходить процедуру автентифікації та отримує доступ до особистого кабінету у режимі перегляду, має можливість моніторити поточний стан виключно закріплених за ним об'єктів нерухомості, аналізувати сформовані фінансові звіти, переглядати прогнози бюджету та базову інформацію про активних орендарів без права внесення змін до бази даних платформи.
Агент з нерухомості	Користувач є основною операційною ланкою платформи після проходження автентифікації, має повний доступ до управління закріпленими за ним об'єктами, створення нових лістингів, генерації посилань для отримання заявок на оренду, укладання електронних договорів з фіксацією їхніх статусів, ведення фінансового обліку доходів та витрат, планування бюджетів та оцінювання орендарів після завершення дії угод.
Адміністратор	Користувач має статус суперкористувача системи з необмеженим доступом до всіх без винятку даних платформи, володіє повним інструментарієм агента з нерухомості, а також додатково здійснює управління обліковими записами інших користувачів, має право змінювати їхні системні ролі, блокувати доступ до платформи та керувати глобальними налаштуваннями безпеки.

Найбільш розгалуженим та функціонально насиченим є алгоритм роботи агента з нерухомості, оскільки саме ця роль відповідає за виконання переважної більшості операційних завдань на платформі. Логіка його роботи охоплює безперервний цикл управління об'єктами, що включає створення нових карток майна, завантаження мультимедійних даних, генерацію електронних посилань для збору заявок, а також подальший контроль фінансових надходжень і взаємодію з орендарями.

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

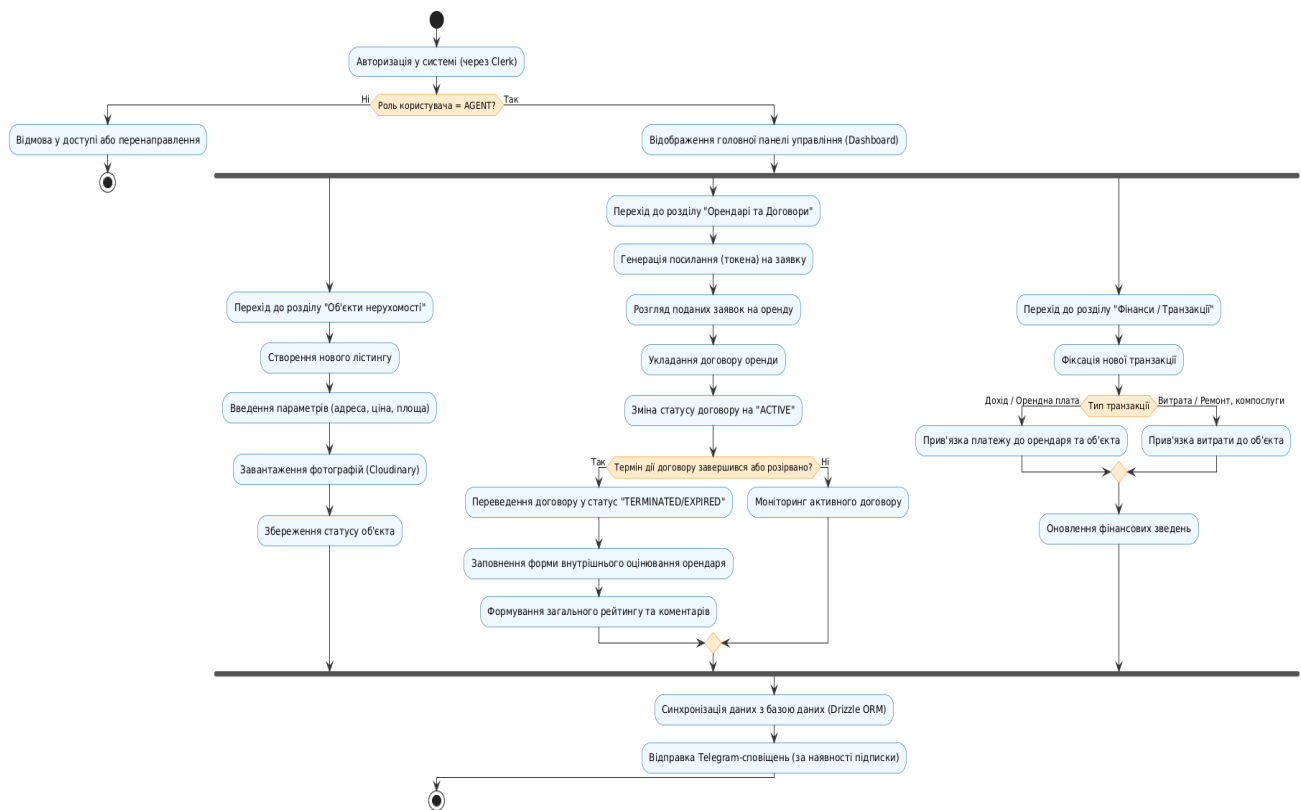


Рис.2.3 Діаграма активності агента з нерухомості

На запропонованій блок-схемі детально візуалізовано повний цикл робочих процесів головного операційного користувача платформи. Схема відображає послідовність кроків від успішної автентифікації агента у системі до етапів створення лістингів, фіксації щомісячних транзакцій орендарів та заповнення форм внутрішнього оцінювання після розірвання договорів. Наявність цієї ілюстрації є критично необхідною для розуміння того, як система обробляє фінансові та управлінські запити, забезпечуючи абсолютну зв'язність даних між модулями об'єктів, договорів та бухгалтерії.

Алгоритм взаємодії власника майна з платформою є значно лінійнішим і спрямований виключно на отримання достовірної аналітичної інформації. Після проходження процедури авторизації система автоматично ідентифікує закріплені за користувачем об'єкти нерухомості та формує для них актуальні фінансові зведення на основі даних, попередньо внесених відповідальними агентами.

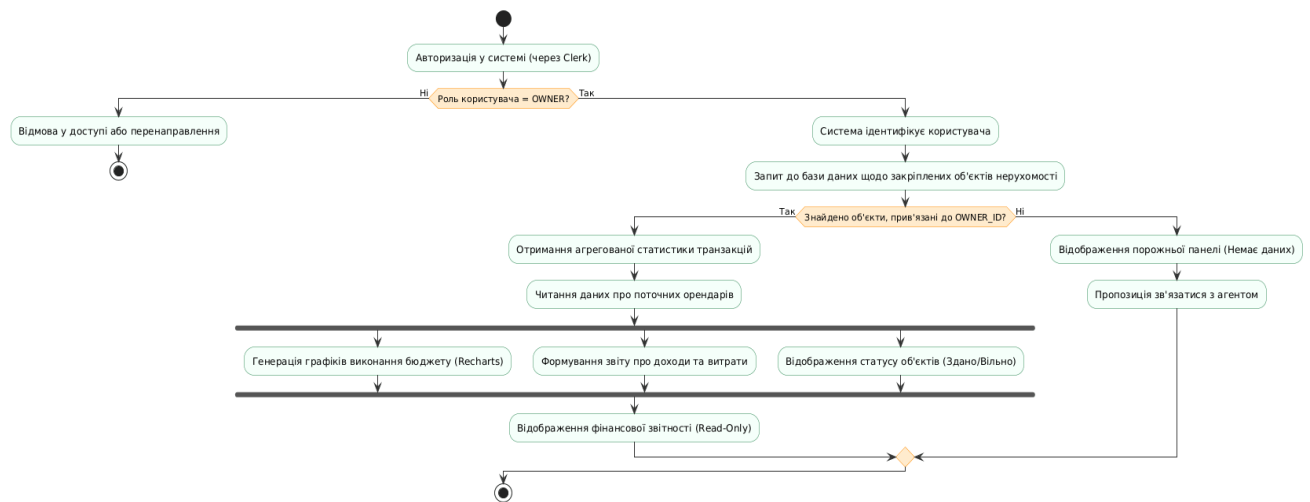


Рис.2.4 Діаграма активності власника нерухомості

Представлена діаграма ілюструє пасивну модель взаємодії інвестора з інформаційною системою. Вона наочно демонструє процес звернення клієнтського додатку до бази даних для отримання агрегованої статистики щодо доходів і витрат, а також алгоритм генерації графіків виконання бюджету без права внесення змін до вихідних параметрів. Ця візуалізація об'єктивно необхідна у тексті для підтвердження успішної реалізації принципу ізольованого доступу та демонстрації механізму формування фінансової звітності у режимі реального часу.

Окремої уваги заслуговує алгоритм роботи системного адміністратора, який несе відповідальність за підтримку стабільної працездатності платформи та управління доступом. Його сценарії використання фокусуються на моніторингу загального стану системи, управлінні обліковими записами агентів, перевірці їхньої активності та контролі глобальних налаштувань безпеки.

Висновки до другого розділу

Під час проєктування комплексної вебплатформи управління нерухомістю «Estate» було здійснено ґрунтовний аналіз основних сценаріїв взаємодії різних категорій користувачів із системою, а також чітко сформульовано вичерпні вимоги до її функціональних та нефункціональних

характеристик. На основі проведеного аналізу розроблено об'єктно-орієнтовану модель системи, яка концептуально охоплює ключові компоненти клієнтської та серверної частин, забезпечуючи їхню безперебійну взаємодію у межах обраної архітектури.

Для надійного збереження та ефективної обробки інформації спроектовано оптимізовану структуру реляційної бази даних, розгорнутої на базі системи «PostgreSQL». Розроблена структура включає ізольовані таблиці для обліку профілів інвесторів, детальних характеристик об'єктів нерухомості, електронних договорів оренди, заявок від потенційних клієнтів та фінансових транзакцій. Сукупність цих елементів гарантує абсолютну цілісність даних та відмовостійкість модуля фінансового планування.

У процесі проєктування також визначено базову алгоритмічну логіку взаємодії користувачів із платформою, формалізовану через відповідні діаграми активності. Вони наочно відображають типові сценарії операційної роботи агентів з нерухомості, процеси моніторингу аналітики власниками майна та процедури контролю системи з боку адміністраторів.

Окрему увагу в межах розділу приділено детальному розгляду внутрішньої механіки ключових функціональних модулів розроблюваного програмного продукту. Було послідовно описано логіку створення нових лістингів майна, процеси безпечного завантаження мультимедійних файлів у хмарне сховище, маршрутизацію даних між клієнтом та сервером, а також механізми суворої валідації введеної інформації на клієнтській стороні до моменту її безпосереднього запису у базу даних.

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБСИСТЕМОЮ ОБЛІКУ ТА АНАЛІЗУ НЕРУХОМОСТІ

3.1 Порядок встановлення та налаштування параметрів системи

Порядок розгортання, конфігурування та точного налаштування програмних параметрів вебплатформи управління нерухомістю «Estater» є фундаментальним етапом усього життєвого циклу розробки та впровадження програмного забезпечення. Від коректності та оптимізації налаштувань усіх інфраструктурних вузлів безпосередньо залежить загальна стабільність функціонування системи у виробничому середовищі, стійкість до відмов, рівень захищеності конфіденційних даних користувачів, а також швидкість обробки клієнтських запитів за умов інтенсивного пікового навантаження. Процес автоматизованого впровадження архітектурних компонентів передбачає обов'язкове проектування та налаштування конвеєра безперервної інтеграції та безперервної доставки, що дозволяє мінімізувати людський фактор під час оновлення кодової бази додатка. Окрему увагу в межах цього етапу приділено точній конфігурації змінних оточення, які використовуються для безпечного підключення до сторонніх мікросервісів, віддалених баз даних та зовнішніх API-інтерфейсів без ризику компрометації секретних ключів та токенів у репозиторіях вихідного коду.

Основні обчислювальні потужності платформи «Estater», що охоплюють як клієнтську, так і серверну частини вебдодатка, делеговано передовому хмарному провайдеру «Vercel». Вибір цієї інфраструктурної платформи обумовлений її високою оптимізацією під сучасні вебфреймворки та здатністю забезпечувати автоматичне горизонтальне масштабування ресурсів у режимі реального часу. Використання хмарної безсерверної архітектури повністю усуває потребу в традиційному ручному адмініструванні операційних систем, конфігуруванні вебсерверів та виділенні фіксованих віртуальних машин. Замість цього «Vercel» надає глобально розподілену мережу периферійних серверів, яка здійснює кешування та доставку контенту з мінімальними затримками залежно від географічного розташування кінцевого користувача

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

додатка.

Архітектура розгортання відображає фізичний та логічний розподіл компонентів системи між різними апаратними та програмними вузлами, визначаючи межі відповідальності кожного елемента інфраструктури. Для наочної візуалізації топології мережі, ієрархії підсистем та механізмів міжкомпонентної взаємодії між клієнтськими пристроями, сервером додатків, платіжними шлюзами та хмарними базами даних було розроблено відповідну структурно-технологічну схему.

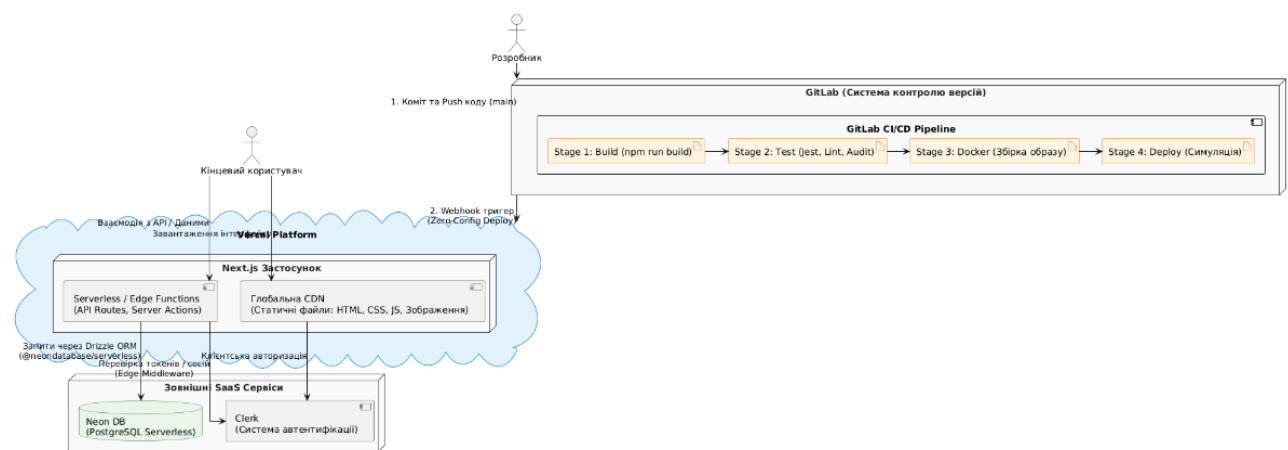


Рис.3.1 Діаграма розгортання платформи «Estateer»

На представленій діаграмі розгортання (рис. 3.1) детально проілюстровано архітектурну топологію інфраструктури вебплатформи з акцентом на використання повністю безсерверних хмарних технологій нового покоління. Схема чітко відображає маршрутизацію мережевих запитів від веббраузера користувача через захищене з'єднання за протоколом HTTPS до безсерверних функцій платформи «Vercel», які виступають у ролі динамічного бекенду додатка. Крім того, візуалізація демонструє інтеграцію із ізольованими зовнішніми вузлами, показуючи захищені канали зв'язку між основним ядром додатка, хмарною реляційною базою даних «Neon» [5], спеціалізованим об'єктним сховищем медіафайлів «Cloudinary», платіжним шлюзом та незалежним сервісом автентифікації.

Клієнтський рівень архітектури представлений робочою станцією або

мобільним пристроєм користувача, з якого здійснюється безпосередній доступ до системи через сучасний веббраузер. Браузер виконує завантажений оптимізований клієнтський код на базі бібліотеки «React» [2], обробляючи інтерфейсні події, керуючи станом компонентів та ініціюючи асинхронні HTTP-запити до серверної частини. Замість розгортання традиційного монолітного сервера, що постійно функціонує в очікуванні підключень, логіка бекенду реалізована через механізм серверних дій фреймворку «Next.js» [1]. Ці дії автоматично ізолюються та компілюються у незалежні безсерверні функції, що дозволяє платформі миттєво реагувати на непередбачувані піки навантаження. Ресурси виділяються виключно в момент фактичного виконання конкретного запиту користувача, а після завершення операції автоматично вивільнюються, що оптимізує споживання хмарних потужностей та підвищує відмовостійкість системи.

Обробка, структурування та довгострокове збереження бізнес-даних системи здійснюється на віддаленому сервері баз даних, який розміщений на спеціалізованій хмарній платформі «Neon» [5]. Платформа надає безсерверний варіант СУБД PostgreSQL, що підтримує динамічне масштабування обчислювальних вузлів від нуля до заданого максимуму залежно від інтенсивності потоку запитів. Взаємодія між безсерверними функціями додатка та реляційною базою даних «PostgreSQL» [6] виконується за допомогою сучасного інструменту об'єктно-реляційного відображення «Drizzle ORM» [4]. Це гарантує сувору типізацію на рівні мови TypeScript, забезпечує генерацію безпечних SQL-запитів на етапі компіляції та мінімізує затримки з'єднання завдяки використанню оптимізованих безсерверних драйверів.

Для суттєвого розширення функціональних можливостей, забезпечення модульності та підвищення надійності система активно взаємодіє з низкою зовнішніх спеціалізованих сервісів через захищені API-інтерфейси. Зокрема, збереження великих обсягів неструктурованих даних, таких як фотографії об'єктів нерухомості у високій роздільній здатності, реалізовано через

інтеграцію з хмарним сервісом «Cloudinary». Медіафайли завантажуються безпосередньо з клієнтського інтерфейсу у хмарне сховище, після чого сервер платформи отримує та зберігає в базі даних лише оптимізовані посилання. Сервіс автоматично здійснює стиснення зображень та їх дистрибуцію через мережу доставки контенту, що значно прискорює рендеринг інтерфейсу.. Паралельно з цим, управління життєвим циклом захищених сесій та авторизація користувачів забезпечується зовнішнім провайдером «Clerk» [7]. Інтеграція з цим сервісом знімає з основного сервера навантаження, пов'язане з хешуванням паролів і генерацією токенів, гарантуючи відповідність системи найвищим міжнародним стандартам безпеки персональних даних.

3.2 Структура інтерфейсу вебсистеми для обліку та аналізу нерухомості

Після успішного проектування архітектури бази даних, вибору технологічного стеку та формалізації бізнес-логіки на серверному рівні, критичним етапом розробки стало створення ефективного клієнтського середовища. Якість реалізації візуальної частини безпосередньо впливає на швидкість адаптації нових користувачів та загальну продуктивність їхньої щоденної операційної діяльності.

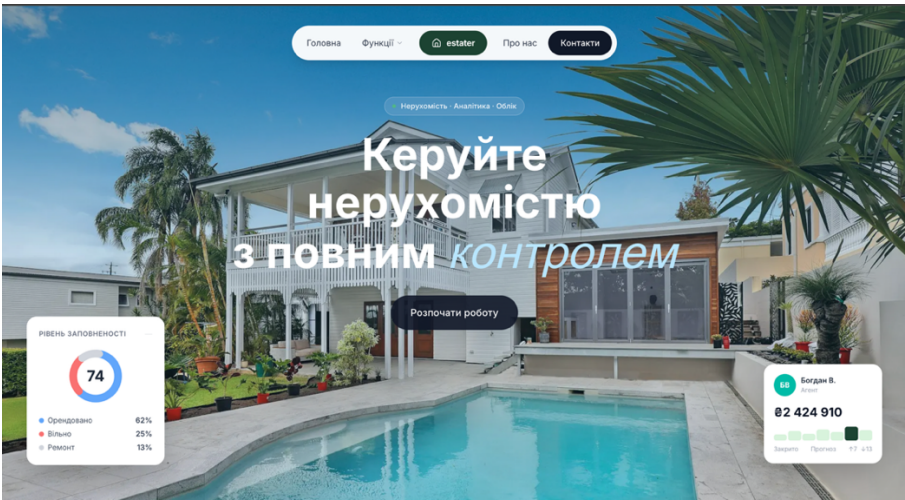


Рис. 3.2 Головна сторінка системи управління нерухомістю «estater»

У верхній частині інтерфейсу розташовано навігаційне меню для швидкого доступу до ключових розділів платформи (Головна, Функції, Про

нас, Контакти). Центральним візуальним елементом є типографічний блок із головним гаслом та акцентною кнопкою «Розпочати роботу» (Call to Action), що ініціює процес реєстрації або входу.

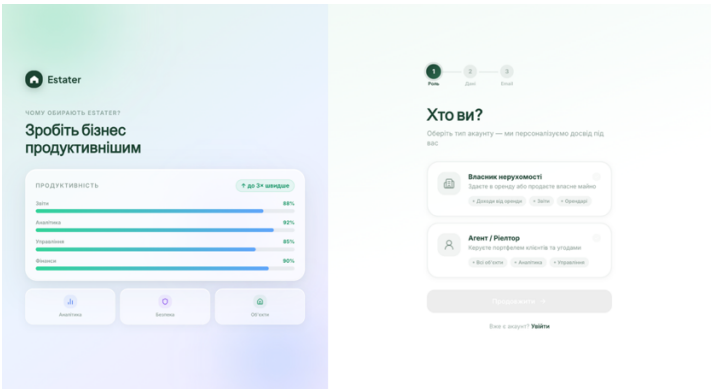


Рис. 3.3 Сторінка реєстрації та вибору ролі користувача в

Інтерфейс вікна реєстрації та первинного налаштування профілю (рис.3.3) візуально розділений на дві функціональні частини. Ліва панель виконує інформаційно-презентаційну функцію, акцентуючи увагу на перевагах використання платформи. Тут розміщено логотип системи, мотиваційне гасло та інфографічний віджет, що демонструє прогнозоване зростання продуктивності за ключовими напрямками (звітність, аналітика, управління, фінанси). Права частина робочої області безпосередньо відповідає за процес створення облікового запису.

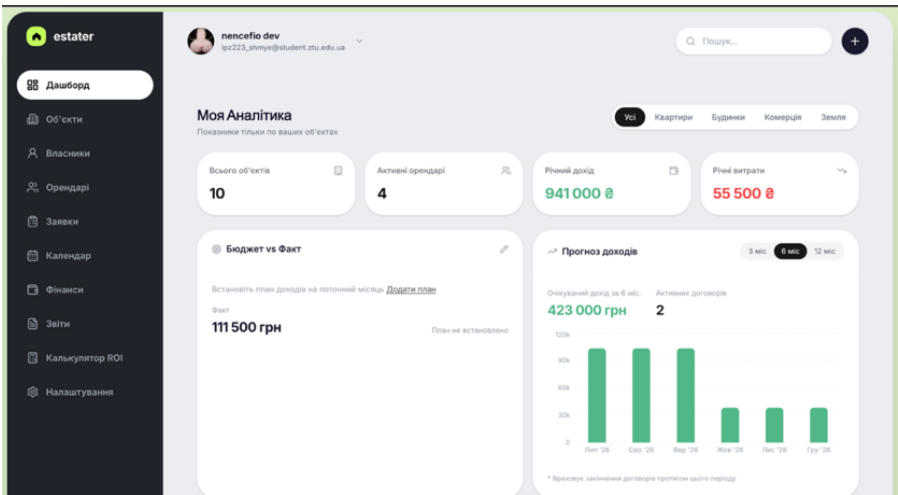


Рис. 3.4 Головна панель керування (дашбордин) та зведена аналітика платформи

Головна панель керування (рис.3.4) системи «estater» слугує центральним аналітичним вузлом для комплексного моніторингу ключових показників управління нерухомістю. Архітектура інтерфейсу логічно розділена на три основні функціональні блоки. Бічне навігаційне меню, розташоване ліворуч на контрастному темному тлі, забезпечує маршрутизацію між усіма робочими модулями платформи, такими як управління об'єктами, власниками, орендарями, фінансами, звітністю, а також містить інструмент розрахунку ROI (окупності інвестицій) та системні налаштування.

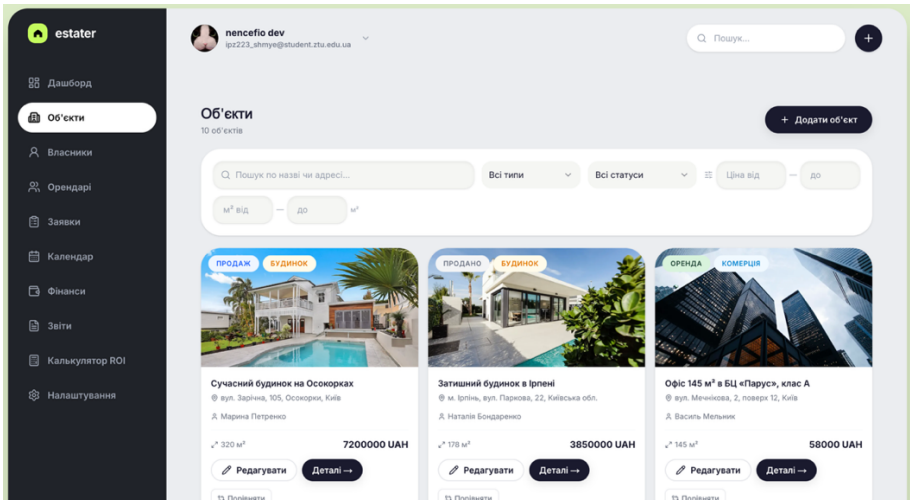


Рис. 3.5 Сторінка управління об'єктами нерухомості та розширена система фільтрації

Розділ «Об'єкти» (рис.3.5) є основним робочим простором для управління портфелем нерухомості в системі «estater». У верхній частині робочої області розташовано заголовок із лічильником загальної кількості доступних записів та акцентною кнопкою «Додати об'єкт», що ініціює процес створення нової сутності в базі даних. Безпосередньо під заголовком розміщено комплексну панель пошуку та фільтрації. Цей інструментарій дозволяє користувачеві здійснювати швидкий текстовий пошук за назвою чи адресою, а також застосовувати параметричні фільтри за допомогою випадаючих списків (вибір типу нерухомості та її поточного статусу) і полів для введення числових діапазонів (обмеження за ціною та загальною площею в квадратних метрах).

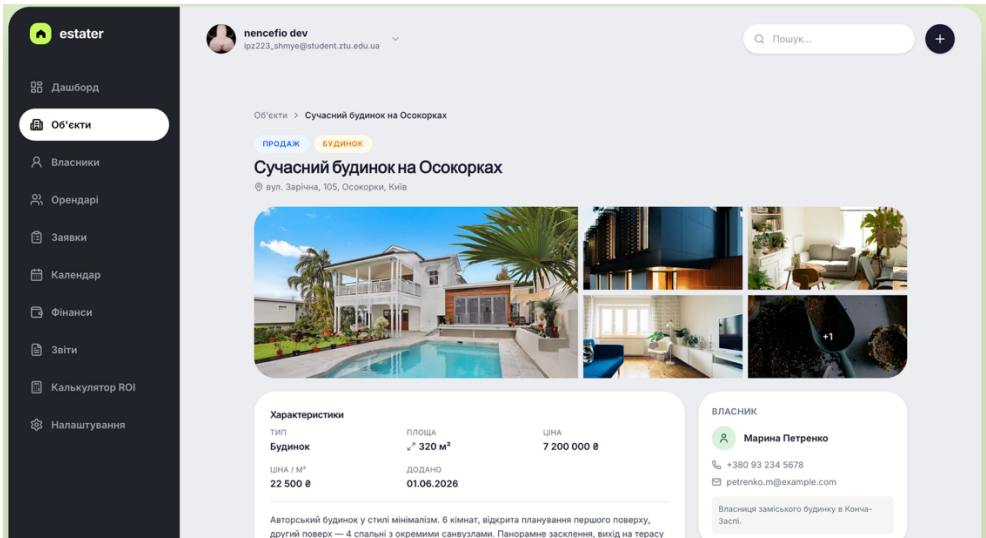


Рис. 3.6 Сторінка детальної інформації про об'єкт нерухомості та картка власника

Сторінка детальної інформації про об'єкт (рис.3.6) слугує розширеним профілем конкретної одиниці нерухомості в системі. У верхній частині робочої області реалізовано навігаційний ланцюжок (breadcrumbs), що дозволяє користувачеві легко відстежувати своє поточне місцезнаходження в ієрархії платформи. Під ним розташовано заголовок профілю, який містить кольорові ідентифікаційні теги поточного статусу та типу майна, офіційну назву об'єкта, а також його точну фізичну адресу з відповідним географічним маркером.

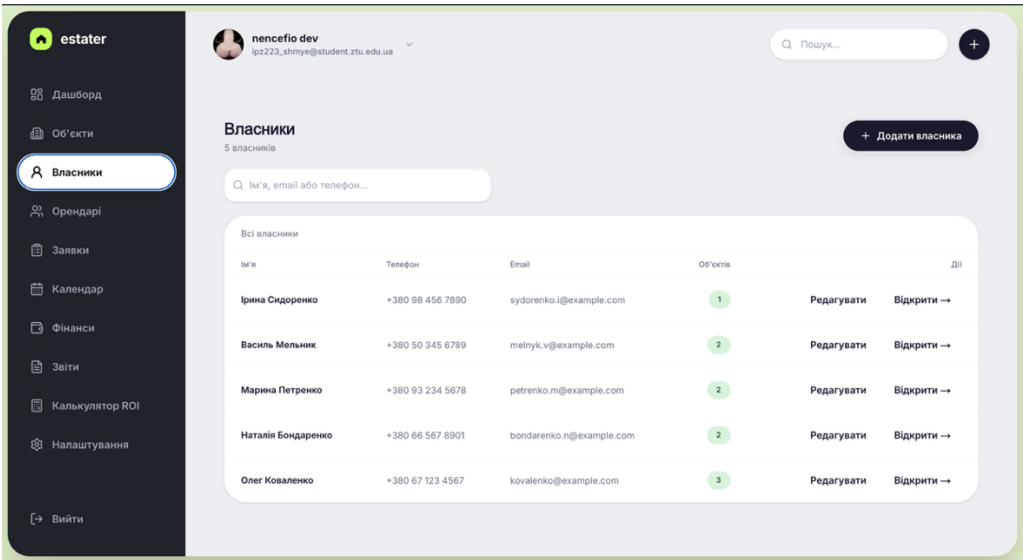


Рис. 3.7 Сторінка управління базою власників нерухомості

Розділ «Власники» (рис 3.7) призначений для централізованого управління контактною базою клієнтів, які володіють об'єктами нерухомості в межах системи «estater». У верхній частині робочої області розташовано заголовок із лічильником загальної кількості зареєстрованих записів та акцентною кнопкою «Додати власника», що ініціює процес створення нового клієнтського профілю. Безпосередньо під заголовком розміщено поле локального пошуку, яке дозволяє оперативно знаходити потрібну особу за ключовими ідентифікаційними даними - ім'ям, контактним номером телефону або адресою електронної пошти.

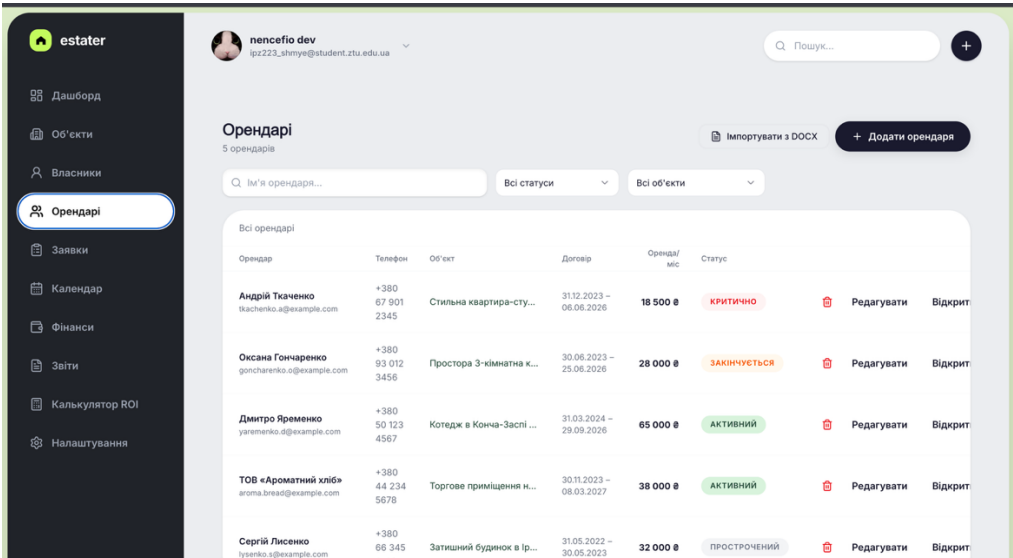


Рис. 3.8 Сторінка управління базою орендарів та моніторингу договорів

Розділ «Орендарі» (рис 3.8) є спеціалізованим модулем для адміністрування клієнтської бази користувачів, які орендують об'єкти нерухомості. Кожен рядок містить комплексну інформацію про клієнта - ідентифікаційні та контактні дані (ім'я, електронна пошта, номер телефону), назву орендованого об'єкта, часові рамки дії договору та фіксовану щомісячну вартість оренди. Важливим інструментом моніторингу є колонка з кольоровими індикаторами статусу угод (наприклад, «КРИТИЧНО», «ЗАКІНЧУЄТЬСЯ», «АКТИВНИЙ», «ПРОСТРОЧЕНИЙ»), що допомагає адміністратору швидко виявляти пріоритетні для реагування договори.

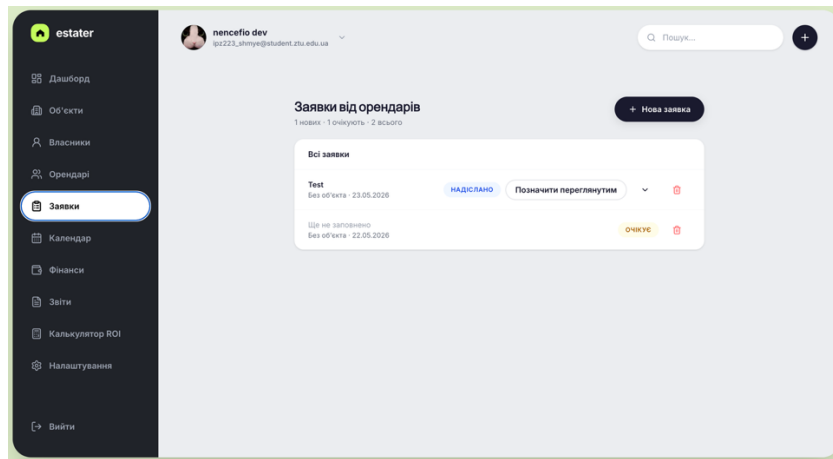


Рис. 3.9 Сторінка управління заявками від орендарів

Розділ «Заявки» (рис 3.9) слугує функціональним модулем для централізованої обробки та адміністрування вхідних звернень від орендарів. У цьому робочому просторі користувач має змогу як самостійно створювати нові запити за допомогою кнопки «Нова заявка», так і керувати поточними - відстежувати їхній стан, змінювати статус обробки (наприклад, «Позначити переглянутим») або безповоротно видаляти неактуальні записи.

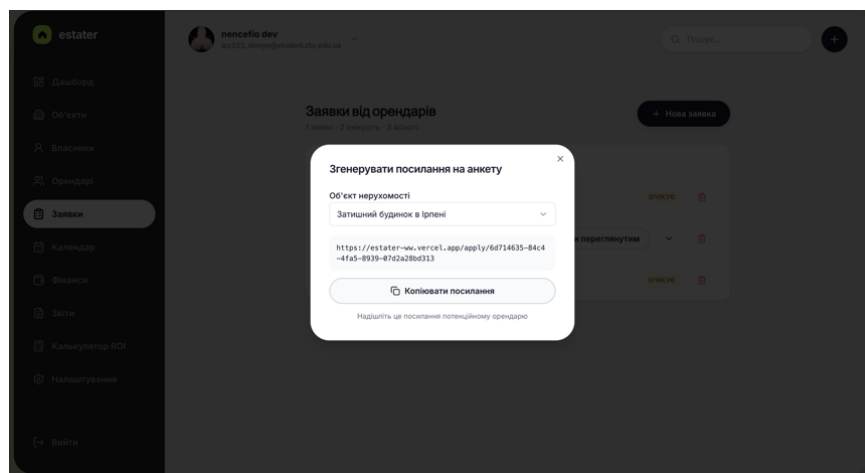


Рис. 3.10 Модальне вікно генерації посилання на анкету для потенційних орендарів

Модальне вікно «Згенерувати посилання на анкету» (рис 3.10) надає користувачеві інструмент для автоматичного створення унікальної URL-адреси з метою збору вхідних даних від потенційних клієнтів. Обравши конкретний об'єкт нерухомості з випадваючого списку

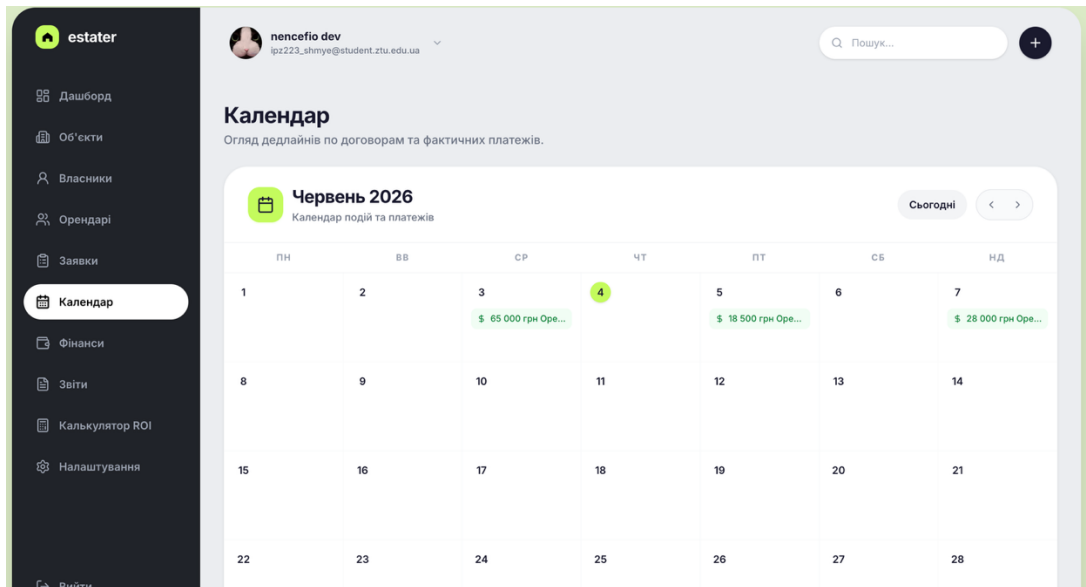


Рис. 3.11 Сторінка інтерактивного календаря для моніторингу платежів та подій

Розділ «Календар» (рис 3.11) являє собою інтерактивний планувальник для візуального контролю фінансових надходжень та дедлайнів за поточними договорами оренди. Робоча область виконана у форматі класичної сітки місяця, де користувач може зручно перемикатися між періодами за допомогою кнопок навігації або миттєво повертатися до поточної дати за допомогою кнопки «Сьогодні». На сітці календаря кольоровими маркерами автоматично відображаються заплановані транзакції (суми очікуваних орендних платежів), що дозволяє адміністратору наочно відстежувати графік виплат та контролювати своєчасність фінансових розрахунків із клієнтами.

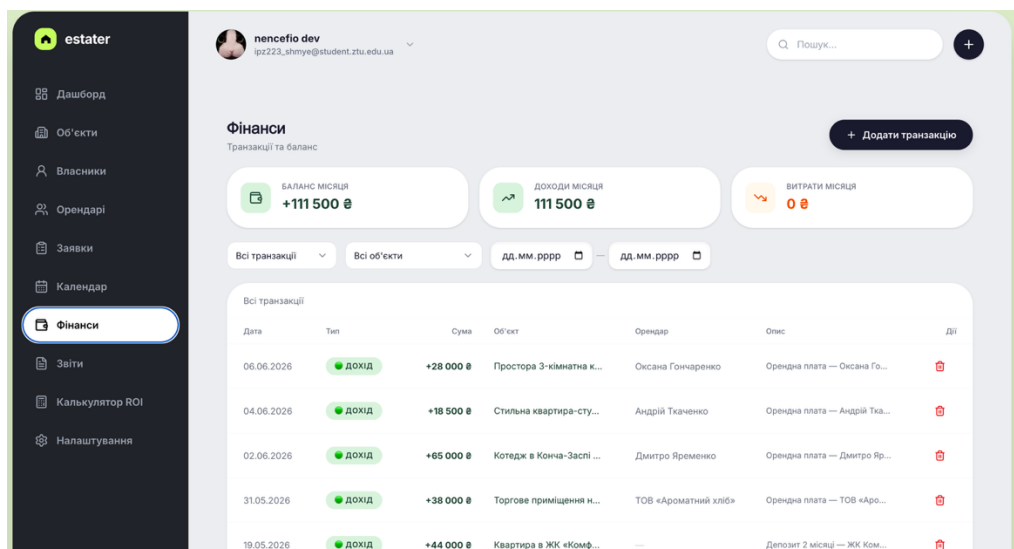


Рис. 3.12 Сторінка фінансового обліку та управління транзакціями

Розділ «Фінанси» (рис 3.12) призначений для комплексного обліку грошових потоків, де користувач може миттєво відстежувати загальний баланс, доходи та витрати за поточний місяць за допомогою зведених інформаційних карток. Робоча область дозволяє вручну фіксувати нові фінансові операції через кнопку «Додати транзакцію», а також гнучко фільтрувати історію платежів за їхнім типом, конкретним об'єктом нерухомості або заданим часовим діапазоном. У деталізованій таблиці транзакцій адміністратор має змогу аналізувати інформацію про кожен платіж (суму, прив'язку до орендаря та призначення), а також оперативно видаляти помилкові записи з бази даних.

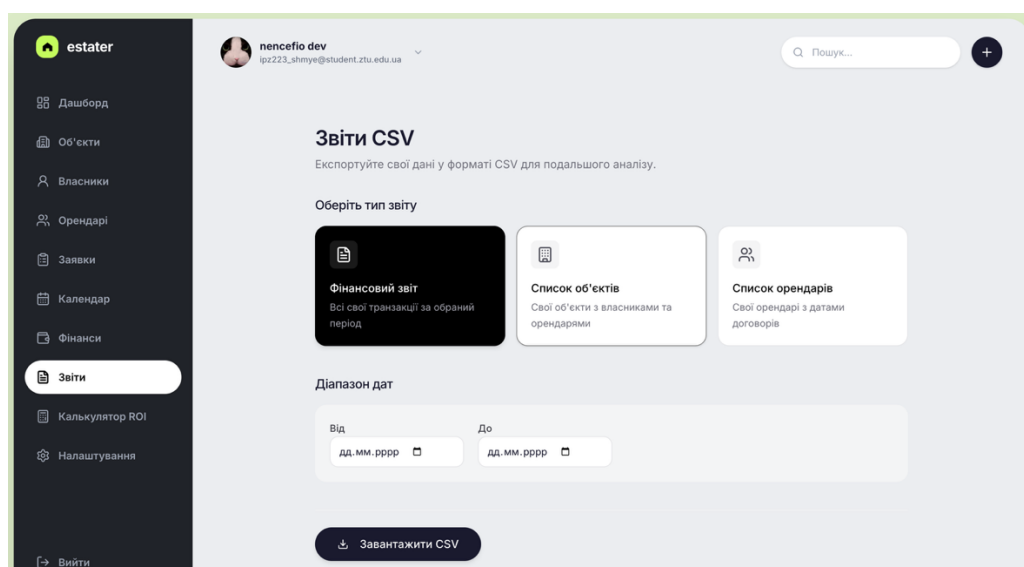


Рис. 3.13 Сторінка генерації та експорту звітів у форматі CSV

Розділ «Звіти» (рис 3.13) слугує спеціалізованим інструментом для експорту масивів системних даних у форматі CSV з метою їх подальшого зовнішнього аналізу. Користувач має можливість гнучко налаштувати вивантаження, обравши потрібну категорію інформації (фінансові транзакції, перелік об'єктів або базу орендарів) та обмеживши вибірку конкретним часовим діапазоном через вбудований календар. Після конфігурації параметрів адміністратор може миттєво згенерувати та зберегти готовий файл на свій локальний пристрій за допомогою кнопки «Завантажити CSV».

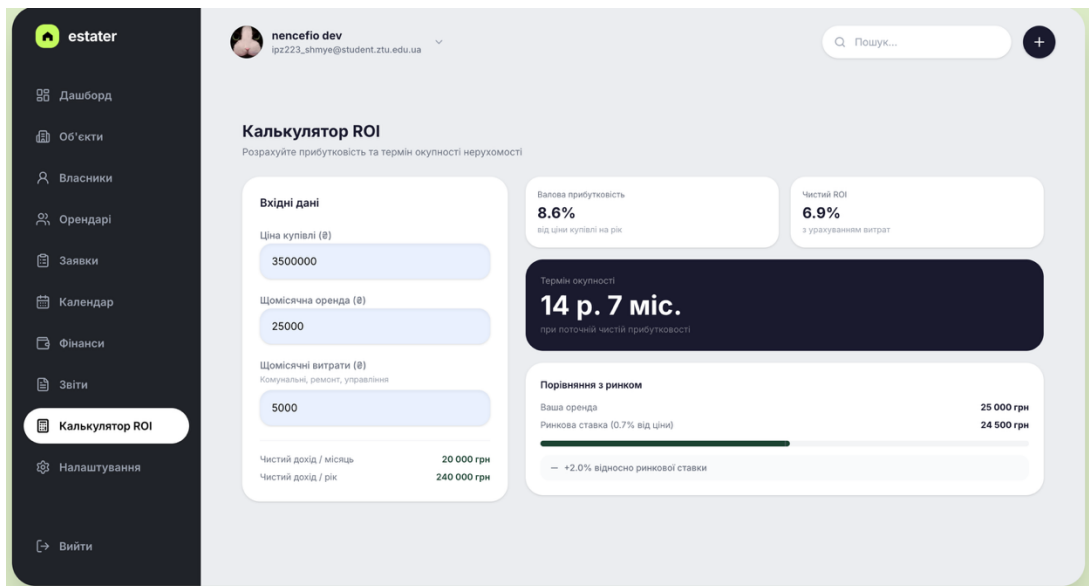


Рис. 3.14 Сторінка інвестиційного калькулятора ROI

Розділ «Калькулятор ROI» (рис 3.14) надає користувачеві спеціалізований інструмент для швидкої оцінки інвестиційної привабливості об'єктів нерухомості. Шляхом введення базових фінансових показників (ціни купівлі, планованої орендної ставки та супутніх щомісячних витрат) у відповідні поля, адміністратор автоматично отримує детальний розрахунок ключових метрик - валової прибутковості, чистого показника ROI та точного терміну окупності. Додатково система генерує аналітичний віджет, що дозволяє користувачеві візуально порівняти встановлену орендну плату із середньоринковими ставками.

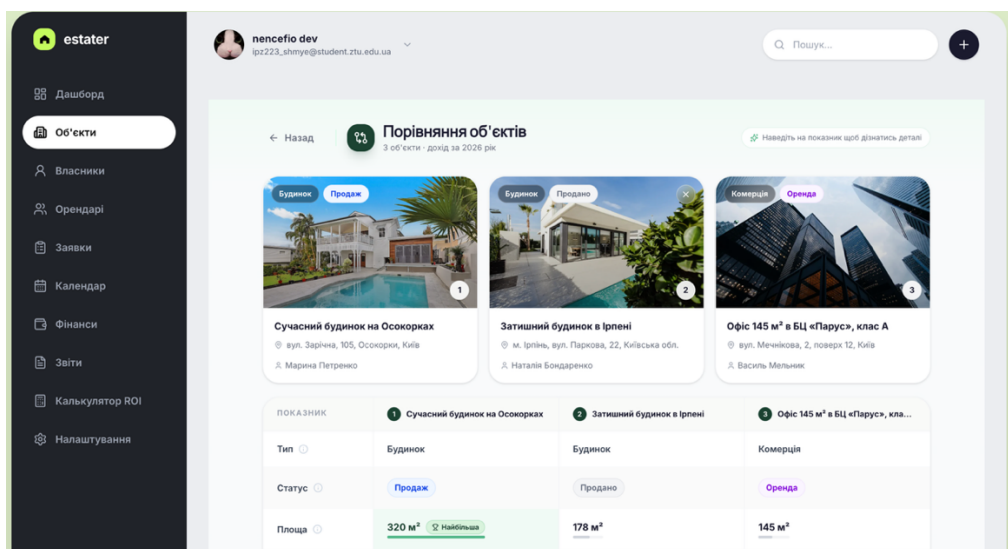


Рис. 3.15 Сторінка паралельного порівняння характеристик об'єктів нерухомості

Сторінка «Порівняння об'єктів» (рис 3.15) надає користувачеві зручний аналітичний інструмент для паралельного зіставлення характеристик одразу кількох обраних одиниць нерухомості. Завдяки структурованому табличному формату адміністратор може візуально аналізувати ключові метрики (такі як поточний статус, тип та площа), а також отримувати розширену інформацію за допомогою інтерактивних підказок при наведенні курсору. Цей функціонал дозволяє швидко виявляти відмінності між пропозиціями для прийняття обґрунтованих рішень, маючи при цьому змогу оперативно видаляти об'єкти з поточної вибірки.

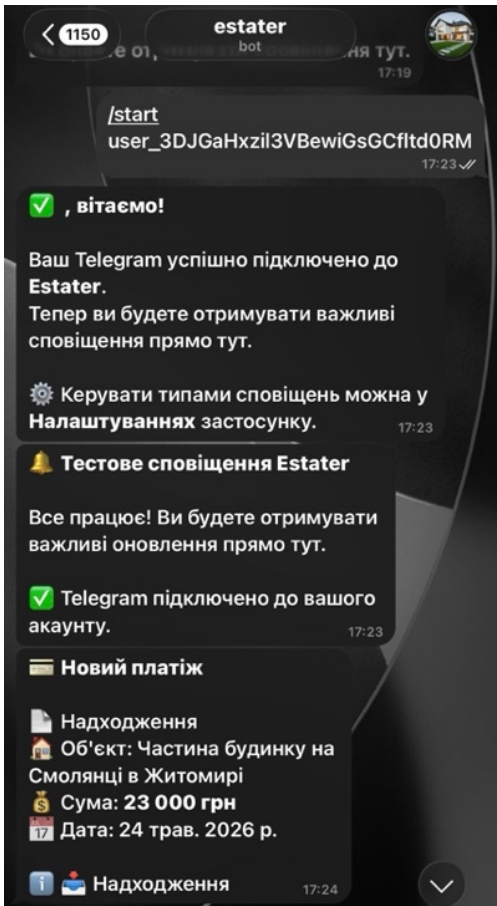


Рис. 3.16 Інтерфейс Telegram-бота для отримання системних сповіщень

Інтерфейс Telegram-бота «estater» (рис 3.16) демонструє можливості інтеграції платформи з месенджером для налаштування миттєвих системних сповіщень. Завдяки цій функції користувач отримує автоматичні push-повідомлення про ключові події, зокрема деталізовані звіти про нові фінансові надходження за конкретними об'єктами нерухомості. Це дозволяє

адміністратору оперативно моніторити фінансову активність у режимі реального часу, керуючи типами розсилок безпосередньо через налаштування основного застосунку.

3.3 Тестування роботи програмного продукту

Тестування системи охоплює три рівні - модульні тести утиліт і доменної логіки на Jest [14], модульні тести клієнтської частини на Jest [14] та React Testing Library і інтеграційні тести серверної частини з реальною базою даних PostgreSQL [6] через Drizzle ORM [4]. Кожен рівень вирішує свою задачу і навмисно не дублює інші. Серверна частина має 42 модульних тести, всі проходять успішно. Клієнтська 12 тестових файлів, що охоплюють компоненти, хуки і сторінки.

```
PASS src/lib/__tests__/utils.test.ts
formatCurrency
  ✓ formats zero as UAH currency (13 ms)
  ✓ formats a positive integer amount
  ✓ formats a large amount without decimal digits
  ✓ rounds fractional amounts to whole numbers
formatDate
  ✓ formats a Date object as DD.MM.YYYY (3 ms)
  ✓ formats an ISO date string (1 ms)
  ✓ returns a string in DD.MM.YYYY pattern
getDaysUntil
  ✓ returns a positive number for a future date
  ✓ returns a negative number for a past date
  ✓ returns approximately 10 for a date 10 days from now (1 ms)
getContractStatus
  ✓ returns "ok" when contractEnd is null
  ✓ returns "ok" for a contract ending in 60 days
  ✓ returns "warning" for a contract ending in 20 days (1 ms)
  ✓ returns "critical" for a contract ending in 5 days
  ✓ returns "expired" for a contract that ended yesterday

Test Suites: 1 passed, 1 total
Tests: 15 passed, 15 total
Snapshots: 0 total
Time: 0.44 s, estimated 1 s
Ran all test suites matching /src\\lib|src\\db/i.
```

Рис. 3.17 Результати виконання модульних тестів серверної частини

Доменні тести перевіряють інваріанти сутностей та форматування без жодної залежності від бази чи HTTP. Наприклад, статус контракту обчислюється коректно на основі дати завершення, грошові суми форматовуються згідно з вимогами.

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

Тести доменного рівня

Клас	Тест	Що перевіряється
Utils	formatCurrency_FormatsZeroAsUAH	Нульова сума форматується у гривнях
Utils	formatCurrency_RoundsFractionalAmounts	Округлення дробових сум до цілих чисел
Utils	formatDate_FormatsISODateString	Коректне перетворення ISO дат в формат DD.MM.YYYY
Utils	getDaysUntil_ReturnsNegativeForPastDate	Минулі дати повертають від'ємну кількість днів
Utils	getContractStatus_CriticalForEndingIn5Days	Контракт, що закінчується через 5 днів, має статус 'CRITICAL'
ContractInvariants	CreateContract_WhenEndDateBeforeStart_ShouldThrow	Дата кінця контракту не може бути раніше дати початку
PropertyInvariants	CreateProperty_WhenPriceIsNegative_ShouldThrow	Від'ємна ціна об'єкта недопустима
TenantInvariants	SetTenantActive_WithoutProperty_ShouldThrow	Орендар без прив'язаного об'єкта не може бути активним

Репозиторії і сервіси підмінюються через Mocks (`jest.mock`), тому тест не потребує бази і виконується за мілісекунди. Перевіряється, що обробник створення об'єкту не зберігає дані якщо валідація схеми Zod [13] провалилась, що аналітика повертає правильні фінансові показники та прибутки.

Таблиця 3.6

Тестові класи обробників команд

Клас	Кількість тестів	Що охоплює
PropertyActionsTests	12	Створення, редагування, видалення об'єктів нерухомості
TenantActionsTests	10	Реєстрація орендарів, призначення на об'єкт, завершення оренди
FinanceActionsTests	9	Запис транзакцій, генерація місячних звітів та аналітики
TelegramNotifyTests	11	Формування та відправка повідомлень у Telegram-бот

Тести клієнтської частини написані на Jest [14] з React Testing Library. Ключовий функціонал `Middleware` - неавтентифікований користувач (Clerk [7]) перенаправляється на сторінку логіну. Сторінка Dashboard перевіряє маршрутизацію - власник бачить список об'єктів, агент заявки орендарів (рис. 3.22).

```
> estater@0.1.0 test:coverage
> jest --coverage

PASS src/lib/_tests_/utils.test.ts
  formatCurrency
    ✓ formats zero as UAH currency (12 ms)
    ✓ formats a positive integer amount (1 ms)
    ✓ formats a large amount without decimal digits
    ✓ rounds fractional amounts to whole numbers
  formatDate
    ✓ formats a Date object as DD.MM.YYYY (3 ms)
    ✓ formats an ISO date string
    ✓ returns a string in DD.MM.YYYY pattern (1 ms)
  getDaysUntil
    ✓ returns a positive number for a future date
    ✓ returns a negative number for a past date
    ✓ returns approximately 10 for a date 10 days from now
  getContractStatus
    ✓ returns "ok" when contractEnd is null
    ✓ returns "ok" for a contract ending in 60 days
    ✓ returns "warning" for a contract ending in 20 days
    ✓ returns "critical" for a contract ending in 5 days
    ✓ returns "expired" for a contract that ended yesterday
```

Рис. 3.18 Результати виконання тестів клієнтської частини

Тестові файли клієнтської частини

Файл	Що перевіряється
Middleware.test.ts	Захист маршрутів - анонімний, автентифікований користувач
DashboardPage.test.tsx	Відображення аналітики та віджетів фінансів
PropertiesList.test.tsx	Рендеринг списку об'єктів, фільтрація за статусом ('FOR_SALE', 'RENTED')
TenantApplicationForm.test.tsx	Валідація форми заявки на оренду, обробка помилок
FinancialChart.test.tsx	Відображення графіка доходів/витрат з Recharts
useProperties.test.ts	Хук отримання та кешування списку об'єктів

Інтеграційні тести останній рівень. При кожному запуску піднімається тестова база PostgreSQL [6] і застосовуються всі міграції Drizzle [4]. Перевіряється збереження і читання агрегатів, включно з JSONB-колонками де лежать колекції інфраструктури ('infrastructure') та специфікації ('specs') (рис. 3.23).

```

◇ injected env (14) from .env.local // tip: ⚙ suppress logs { quiet: true }
🧹 Очищення старих даних...
👤 Додавання власників...
🏠 Додавання об'єктів...
👤 Додавання орендарів...
🕒 Додавання транзакцій (6 місяців)...

✅ Seed завершено!
Власники: 5
Об'єкти: 10
Орендарі: 5 (4 активних)
Транзакції: 33
Загальний дохід: ₾941 000
Загальні витрати: ₾55 500

```

Рис. 3.19 Результати виконання інтеграційних тестів та підключення до бази даних

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		57

Зведені дані щодо обраних інструментів та результатів покриття коду зібрано нижче (табл. 3.8).

Таблиця 3.8

Загальне покриття тестами

Рівень	Інструмент	Кількість тестів	Результат
Доменна логіка	Jest	29	Пройдено
Серверні обробники	Jest + Mocks	42	Пройдено
Компоненти клієнта	Jest + Testing Library	43	Пройдено
Репозиторії	Jest + Drizzle ORM	20	Пройдено

Загалом для перевірки працездатності проєкту написано 134 тести. Такий комбінований підхід дозволив протестувати систему з різних боків, не покладаючись на один інструмент. Серверна частина перевірялася фреймворком Jest. Щоб тестувати обробники команд ізольовано від бази даних, зовнішні залежності підмінялися за допомогою вбудованих mock-механізмів. На клієнтській стороні застосовано зв'язку Jest та React Testing Library. Цей набір перевірів правильність відмальовування компонентів інтерфейсу та надійність захисту приватних роутів (завдяки Clerk [7]). Окрему увагу було приділено інтеграційному рівню репозиторіїв. Завдяки Drizzle [4] ORM та тестовій БД вдалося перевірити SQL-запити на реальній базі даних. Оскільки під час прогону жоден тест не видав помилки, поточну версію системи можна вважати стабільною та готовою до експлуатації.

Висновок до третього розділу

У третьому розділі визначено ключові особливості програмної реалізації та детально описані архітектурні рішення, необхідні для забезпечення стабільного функціонування розробленої веб-платформи оренди житла «Estateer». Було проведено ґрунтовний опис розробленої інтерфейсної частини застосунку, а також продемонстровано базові принципи взаємодії різних категорій користувачів із системою. Зокрема, розглянуто процеси перегляду детальної інформації про об'єкти нерухомості, здійснення розширеного

пошуку доступного житла, застосування багатокритеріальної фільтрації за характеристиками та ціновими пропозиціями, а також процедуру оформлення та підтвердження бронювання. Додатково висвітлено спеціалізований інструментарій орендодавця для публікації та управління власними оголошеннями.

Крім того, сформовано чіткий порядок розгортання клієнт-серверної інфраструктури розробленого продукту. У тексті вказано логічну послідовність дій та технічні особливості налаштування клієнтського середовища, підключення безсерверної хмарної бази даних, а також інтеграції сторонніх сервісів для забезпечення безпечної автентифікації. Це дозволяє гарантувати високу масштабованість проєкту та його готовність до роботи в умовах реального веб-середовища.

За результатами проведеного комплексного тестування, зокрема модульного перевіряння компонентів та навантажувального аналізу, вдалося підтвердити, що створений веб-застосунок має високий рівень працездатності. Система стійко витримує конкурентні запити користувачів під час імітації пікових навантажень, а внутрішня бізнес-логіка обробки даних, узгодження статусів бронювань та збереження інформації функціонує абсолютно коректно і без збоїв.

Таким чином, реалізована функціональність повною мірою відповідає поставленим цілям і завданням дослідження, забезпечуючи зручний та ефективний інструментарій для взаємодії орендодавців, орендарів та адміністрації ресурсу. Протестовані програмні модулі демонструють безперебійність, стабільність і високу точність виконання операцій, що робить цей програмний продукт надійним, конкурентоспроможним та повністю придатним для впровадження у практичну діяльність на сучасному ринку нерухомості.

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Результатом виконання кваліфікаційної роботи є створена вебсистема обліку та аналізу нерухомості Estater. Платформа призначена для автоматизації роботи агенцій, незалежних ріелторів та приватних інвесторів. Вона переводить у цифровий формат увесь цикл управління нерухомістю - від базового ведення реєстру об'єктів та фінансів до глибокої візуальної аналітики ринкових трендів за допомогою інтерактивних мап.

Огляд існуючого програмного забезпечення виявив суттєві обмеження наявних на ринку рішень. Популярні масові платформи виявилися закритими екосистемами з жорстко зафіксованим інтерфейсом, які не надають професійним учасникам ринку повноцінних вбудованих інструментів управління портфелем. Жоден із розглянутих сервісів не мав гнучкого механізму, який би дозволяв користувачеві самостійно конструювати аналітичні дашборди під специфічні бізнес-потреби. Це стало головним аргументом для розробки власного рішення.

Архітектура програмного продукту базується на клієнт-серверній моделі з використанням підходу модульного моноліта. Щоб поєднати надійність суворого фінансового обліку з гнучкістю кастомних звітів, було застосовано гібридний підхід до бази даних. Клієнтська частина бере на себе складну логіку візуального редактора, дозволяючи формувати звіти без написання коду.

Головна практична цінність створеної системи полягає у ліквідації хаосу в даних, а саме у відмові від ведення нерухомості в електронних таблицях та розрізнених записниках. Тепер ріелтор контролює всі об'єкти, договори та фінансові транзакції в єдиному місці, а клієнти отримують зручний доступ до інтерактивної аналітики. Стабільність платформи доведено на практиці - для перевірки коду написано 134 тести, які успішно покривають логіку клієнтських компонентів, серверних обробників та інтеграційний рівень взаємодії з базою даних.

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Vercel Inc. Next.js App Router Official Documentation. 2026. URL: <https://nextjs.org/docs> (дата звернення: 05.06.2026).
2. Meta Platforms, Inc. React 19 Documentation. 2026. URL: <https://react.dev/> (дата звернення: 05.06.2026).
3. Microsoft Corporation. TypeScript Documentation. 2026. URL: <https://www.typescriptlang.org/docs/> (дата звернення: 05.06.2026).
4. Drizzle Team. Drizzle ORM Official Guide. 2026. URL: <https://orm.drizzle.team/docs/overview> (дата звернення: 05.06.2026).
5. Neon Inc. Neon Serverless Postgres Documentation. 2026. URL: <https://neon.tech/docs> (дата звернення: 05.06.2026).
6. The PostgreSQL Global Development Group. PostgreSQL 16 Official Documentation. 2026. URL: <https://www.postgresql.org/docs/> (дата звернення: 05.06.2026).
7. Clerk, Inc. Clerk Authentication & User Management. 2026. URL: <https://clerk.com/docs> (дата звернення: 05.06.2026).
8. Tailwind Labs Inc. Tailwind CSS Official Documentation. 2026. URL: <https://tailwindcss.com/docs> (дата звернення: 05.06.2026).
9. WorkOS. Radix UI Primitives (shadcn/ui base). 2026. URL: <https://www.radix-ui.com/docs> (дата звернення: 05.06.2026).
- 10.shadcn. shadcn/ui Installation and Components. 2026. URL: <https://ui.shadcn.com/docs> (дата звернення: 05.06.2026).
- 11.Recharts. Recharts API Reference. 2026. URL: <https://recharts.org/en-US/guide> (дата звернення: 05.06.2026).
- 12.React Hook Form. React Hook Form Documentation. 2026. URL: <https://react-hook-form.com/> (дата звернення: 05.06.2026).
- 13.Zod. Zod: TypeScript-first schema validation. 2026. URL: <https://zod.dev/> (дата звернення: 05.06.2026).
- 14.Meta Platforms, Inc. Jest Testing Framework. 2026. URL: <https://jestjs.io/docs/getting-started> (дата звернення: 05.06.2026).

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

- 15.Svix Inc. Svix Webhooks Integration Guide. 2026. URL: <https://docs.svix.com/>
(дата звернення: 05.06.2026).
- 16.Docker Inc. Docker Official Documentation. 2026. URL: <https://docs.docker.com/> (дата звернення: 05.06.2026).
- 17.GitLab Inc. GitLab CI/CD Pipeline Documentation. 2026. URL: <https://docs.gitlab.com/ee/ci/> (дата звернення: 05.06.2026).
- 18.GitHub, Inc. GitHub Actions Documentation. 2026. URL: <https://docs.github.com/en/actions> (дата звернення: 05.06.2026).
- 19.Three.js Authors. Three.js Documentation (3D Graphics). 2026. URL: <https://threejs.org/docs/> (дата звернення: 05.06.2026).
- 20.Lucide Contributors. Lucide Icons Reference. 2026. URL: <https://lucide.dev/guide/> (дата звернення: 05.06.2026).
- 21.Cloudinary Ltd. Cloudinary Image Management API. 2026. URL: <https://cloudinary.com/documentation> (дата звернення: 05.06.2026).
- 22.Telegram FZ-LLC. Telegram Bot API Specification. 2026. URL: <https://core.telegram.org/bots/api> (дата звернення: 05.06.2026).
- 23.Riva M. Real-World Next.js: Build scalable, high-performance, and modern web applications using Next.js. Packt Publishing, 2022. 366 p.
- 24.Vanderkam D. Effective TypeScript: 83 Specific Ways to Improve Your TypeScript. 2nd edition. O'Reilly Media, 2024. 401 p.
- 25.Freeman A. Pro React 18 with TypeScript. Apress, 2022. 750 p.
- 26.Canton R. Full-Stack Web Development with Next.js. Packt Publishing, 2023. 410 p.
- 27.Wieruch R. The Road to Next: The comprehensive Next.js Tutorial. Independently published, 2024. 350 p.
- 28.Kumar A. Mastering PostgreSQL 15. 4th edition. Packt Publishing, 2023. 532 p.
- 29.Borrie J. TypeScript Cookbook: Real World Type-Level Programming. O'Reilly Media, 2023. 350 p.
- 30.Zammetti F. Modern Full-Stack Development: Using TypeScript, React, Node.js, Webpack, and Docker. 2nd edition. Apress, 2022. 450 p.

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		62

31. Goldberg J. Learning TypeScript: Enhance Your Web Development Skills Using Type-Safe JavaScript. O'Reilly Media, 2022. 320 p.
32. Poulton N. Docker Deep Dive: Zero to Docker in a single book. Independently published, 2023. 300 p

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг програми: фрагмент

```

import { db } from '@db'
import { contracts, tenants, properties } from '@db/schema'
import { and, eq, gte, lt, lte, ne } from 'drizzle-orm'
export type ContractStatus = 'ACTIVE' | 'EXPIRING_SOON' | 'EXPIRED' |
'TERMINATED'
export function computeContractStatus(endDate: Date | string | null | undefined):
ContractStatus {
  if (!endDate) return 'ACTIVE'
  const diff = new Date(endDate).getTime() - Date.now()
  const days = Math.ceil(diff / (1000 * 60 * 60 * 24))
  if (days < 0) return 'EXPIRED'
  if (days <= 30) return 'EXPIRING_SOON'
  return 'ACTIVE'
}
export async function syncContractStatuses(): Promise<void> {
  const now = new Date()
  const in30Days = new Date(now.getTime() + 30 * 24 * 60 * 60 * 1000)
  const nowExpired = await db.query.contracts.findMany({
    where: and(
      lt(contracts.endDate, now),
      ne(contracts.status, 'EXPIRED'),
      ne(contracts.status, 'TERMINATED'),
    ),
  })
  for (const c of nowExpired) {
    await db
      .update(contracts)

```

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        .set({ status: 'EXPIRED', updatedAt: new Date() })
        .where(eq(contracts.id, c.id))
const stillRunning = await db.query.contracts.findFirst({
  where: and(
    eq(contracts.tenantId, c.tenantId),
    ne(contracts.status, 'EXPIRED'),
    ne(contracts.status, 'TERMINATED'),
    ne(contracts.id, c.id),
  ),
})
if (!stillRunning) {
  await db.update(tenants).set({ isActive: false }).where(eq(tenants.id,
c.tenantId))
}
const activeForProperty = await db.query.contracts.findFirst({
  where: and(
    eq(contracts.propertyId, c.propertyId),
    ne(contracts.status, 'EXPIRED'),
    ne(contracts.status, 'TERMINATED'),
    ne(contracts.id, c.id),
  ),
})
if (!activeForProperty) {
  await db
    .update(properties)
    .set({ status: 'FOR_RENT', updatedAt: new Date() })
    .where(eq(properties.id, c.propertyId))
}
}

```

```

await db
  .update(contracts)
  .set({ status: 'EXPIRING_SOON', updatedAt: new Date() })
  .where(
    and(
      eq(contracts.status, 'ACTIVE'),
      gte(contracts.endDate, now),
      lte(contracts.endDate, in30Days),
    ),
  )
} import { auth } from "@clerk/nextjs/server";
import { db } from "@db";
import { properties, transactions } from "@db/schema";
import { and, eq, gte, lte, inArray } from "drizzle-orm";
import type { SQL } from "drizzle-orm";
import { getOwnerId, getRole } from "@lib/auth-utils";
function escapeCsvField(field: unknown): string {
  if (field === null || field === undefined) return "";
  const str = String(field);
  if (str.includes("") || str.includes(",") || str.includes("\n")) {
    return `"${str.replace(/"/g, "")}"`;
  }
  return `"${str}"`;
}
export async function GET(request: Request) {
  const { userId, sessionClaims } = await auth();
  if (!userId) return new Response("Unauthorized", { status: 401 });
  const role = await getRole(userId, sessionClaims);
  const { searchParams } = new URL(request.url);

```

```

const type = searchParams.get("type");
const dateFrom = searchParams.get("dateFrom");
const dateTo = searchParams.get("dateTo");
if (!type || ![ "financial", "objects", "tenants" ].includes(type)) {
  return new Response("Invalid report type", { status: 400 });
}
let allowedPropertyIds: string[] | null = null;
if (role === "AGENT") {
  const rows = await db.select({ id: properties.id
}).from(properties).where(eq(properties.agentId, userId));
  allowedPropertyIds = rows.map((r) => r.id);
  if (allowedPropertyIds.length === 0) return new Response("", { headers: {
"Content-Type": "text/csv" } });
} else if (role === "OWNER") {
  const ownerId = await getOwnerId(userId);
  if (!ownerId) return new Response("", { headers: { "Content-Type": "text/csv" }
});
  const rows = await db.select({ id: properties.id
}).from(properties).where(eq(properties.ownerId, ownerId));
  allowedPropertyIds = rows.map((r) => r.id);
  if (allowedPropertyIds.length === 0) return new Response("", { headers: {
"Content-Type": "text/csv" } });
}
let csvContent = "";
if (type === "financial") {
  const conditions: SQL[] = [];
  if (allowedPropertyIds !== null) {
    conditions.push(inArray(transactions.propertyId, allowedPropertyIds));
  }
  if (dateFrom) conditions.push(gte(transactions.date, new Date(dateFrom)));

```

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

```

if (dateTo) {
  const to = new Date(dateTo);
  to.setHours(23, 59, 59, 999);
  conditions.push(lte(transactions.date, to));
}
const data = await db.query.transactions.findMany({
  where: conditions.length ? and(...conditions) : undefined,
  with: {
    property: { columns: { title: true } },
    tenant: { columns: { name: true } },
  },
});
const headers = ["Date", "Type", "Amount", "Property", "Tenant",
"Description"];
csvContent += headers.map(escapeCsvField).join(",") + "\n";
for (const tx of data) {
  const row = [
    tx.date.toISOString().split("T")[0],
    tx.type === "INCOME" ? "Дохід" : "Витрата",
    tx.amount,
    tx.property.title,
    tx.tenant?.name ?? "",
    tx.description ?? "",
  ];
  csvContent += row.map(escapeCsvField).join(",") + "\n";
}
const bom = "\uFEFF";
return new Response(bom + csvContent, {
  headers: {

```

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						69
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    "Content-Type": "text/csv; charset=utf-8",
    "Content-Disposition": `attachment; filename="report_${type}_${new
Date().toISOString().split("T")[0]}.csv"`,
  },
});
} type NotificationType = 'test' | 'payment' | 'application' | 'tenant' | 'property' |
'report'
interface NotificationPayload {
  title?: string
  amount?: number
  currency?: string
  address?: string
  tenantName?: string
  date?: string
  details?: string
}
export function buildNotificationMessage(
  type: NotificationType,
  payload: NotificationPayload = {},
): string {
  const { title, amount, currency = 'грн', address, tenantName, date, details } =
payload
  switch (type) {
    case 'test':
      return (
        '🔔 <b>Тестове сповіщення Estater</b>\n\n' +
        'Все працює! Ви будете отримувати важливі оновлення прямо тут.\n\n' +
        '✅ Telegram підключено до вашого акаунту.'
      )
  }
}

```

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		70

```

case 'payment':
  return (
    `🏠 <b>Новий платіж</b>\n\n` +
    (title ? `📄 ${title}\n` : "") +
    (tenantName ? `👤 Орендар: ${tenantName}\n` : "") +
    (address ? `🏠 Об'єкт: ${address}\n` : "") +
    (amount !== undefined ? `💰 Сума: <b>${amount.toLocaleString('uk-UA')}${currency}</b>\n` : "") +
    (date ? `📅 17 Дата: ${date}\n` : "") +
    (details ? `📋 ${details}\n` : "")
  )
case 'application':
  return (
    `📄 <b>Нова заявка на оренду</b>\n\n` +
    (tenantName ? `👤 Заявник: ${tenantName}\n` : "") +
    (address ? `🏠 Об'єкт: ${address}\n` : "") +
    (date ? `📅 17 Подано: ${date}\n` : "") +
    (details ? `📋 ${details}\n` : "")
  )
case 'tenant':
  return (
    `👤 <b>Зміна орендаря</b>\n\n` +
    (title ? `${title}\n` : "") +
    (tenantName ? `👤 Орендар: ${tenantName}\n` : "") +
    (address ? `🏠 Об'єкт: ${address}\n` : "") +
    (date ? `📅 17 Дата: ${date}\n` : "") +
    (details ? `📋 ${details}\n` : "")
  )

```

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		71

case 'property':

return (

` <b>Оновлення об'єкта</b>\n\n` +

(title ? `\${title}\n` : ") +

(address ? ` Адреса: \${address}\n` : ") +

(date ? ` Дата: \${date}\n` : ") +

(details ? `\n\n\${details}` : ")

)

case 'report':

return (

` <b>Звіт Estater</b>\n\n` +

(title ? `\${title}\n` : ") +

(date ? ` Період: \${date}\n` : ") +

(details ? `\n\n\${details}` : ")

)

default:

return ` Нове сповіщення від Estater` + (details ? `\n\n\${details}` : ")

}

}

					ІПЗ.КР.Б-121-26-ПЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		